

Interex '94

Conference & Expo

Denver, Colorado
September 18-22, 1994



Proceedings



HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

Interex '94

Conference & Expo



Proceedings

Denver, Colorado

September 18-22, 1994

Sponsored by



**The International
Association of
Hewlett-Packard
Computing
Professionals**

Index by Paper Number

1000		The Subject is Objects
	Orly Larson - Hewlett-Packard Co.	
1002		Delivering Client-Server Functions Today
	Riz Shakir - SSA	
1003		Client-Server RAD: A Case Study
	Ben Lloyd - Hewlett-Packard Co.	
1004		A Comparison of Distributed Application Technologies
	Gerald Duggan - Hewlett-Packard Co.	
1005		Using High-Performance Real-Time I/O with HP-UX
	Robert Cleary - KineticSystems Corporation	
1006		Migrating COBOL to HP-UX
	Alan Meyer - Hewlett-Packard Co.	
1007		A Practical Guide for Integrating Imaging into your Existing Business Applications
	John Carney - WANG	
1008		Future CASE: Enabling End Users to Customize Applications
	Ken Robinson - Auto-trol Technology Corp.	
1009		HP SourceReader UX - HP 3000 Debugging, Development, and Source Understanding Tool of the Future
	Gary Robillard - Hewlett-Packard Co.	
1010		The Client-Server Headache: Performance, and How to Get Relief
	Dan Gillis - DISC	
1011		How to Effectively Select Software Development Tools
	Padma Sreenivasan - Hewlett-Packard Co.	
1014		How to Design GUI's Effectively Using GUI's Standards and Specs
	Padma Sreenivasan - Hewlett-Packard Co.	
1015		Getting the Most out of HP-UX
	Andrew Phillips - Hewlett-Packard Co.	
1016		Visual Basic for the MIS Professional: How to Develop High Performance Client-Server Applications
	Terry O'Brien - DISC	
2000		The HP Information Systems Marketplace Today: What's Hot and What's Not
	Diane Amos - Amos & Associates, Inc.	
2001		Re-Examining Re-Engineering
	Feyzi Fatehi - Hewlett-Packard Co.	
2002		Network Outsourcing: A Key to Corporate Reengineering
	Ernest Eugster - Random Access	
2003		Legal Issues in EDI
	Christopher Seline - Formosa Transnational Attorneys at Law	
2004		Source Code Escrow: A Critical Business and Legal Issue
	Bea Strickland - SourceFile	
2005		Telephone Responsibilities: Has Your MIS Taken Over Yet?
	Rick Hupe - Telenomics Inc.	
2006		Imaging and Document Management
	David Sborov - NSD, Inc.	
2009		Future Strategies for DP Managers
	Roger Lawson - Proactive Systems Ltd.	
2010		Turning Multimedia Data Into Information
	Mike Kennedy - Excalibur Technologies	
2011		Leveraging Support Alliances
	Bashinder Grewal, Kumar Gollabinnie - Hewlett-Packard Co.	
2012		Rightsizing to Client-Server Applications
	John Woolsoncroft - Concepts Dynamic, Inc.	
2013		Thinking About an Image Management System: Think Again and Again
	Stephen Fontana - Software Systems Technology	
2014		Data Center Consolidation at Hewlett-Packard
	John Podkomorski - Hewlett-Packard Co.	
3000		Indexing: When, When Not, and How
	Mark Gardner - Bradmark Technologies, Inc.	

Index by Paper Number

3001		Criteria for RDBMS Selection
	James Hepler - Hewlett-Packard Co.	
3002		An Overview of IMAGE/SQL
	Alvina Nishimoto, Jim Sartain - Hewlett-Packard Co.	
3003		Introduction to IMAGE/SQL
	Denys Beauchemin - HICOMP AMERICA, Inc.	
3004		O.K., It's Installed - Now What? IMAGE/SQL Future
	Liz Norman - Hewlett-Packard Co.	
3005		IMAGE/SQL and Indexing
	Larry Boyd - Bradmark Technologies, Inc.	
3007		Does Your HP "Talk" Back
	Bruce Frank - Frank Solutions	
3008		The New Relational Database: IMAGE/SQL
	Bradley Tashenberg - Bradmark Technologies, Inc.	
3009		Evolving Your HP IM Investment
	Mary Ann Gustafson - Hewlett-Packard Co.	
3013		Working with Allbase/SQL PC API
	Roy Losch - Hewlett-Packard Co.	
3014		The Database Administrator (DBA): A Ruthless Juggler of Reality, Algorithms and Data Structures
	Alfredo Rego - Adager	
3015		Benefits of Data Warehousing
	Terry O'Brien - DISC	
3016		The ODBMS Role in Client-Server Computing
	Leon Guzenda - Objectivity, Inc.	
3017		IMAGE/SQL At Work
	Martin LaHiff - Hewlett-Packard Co.	
3018		Using Replicate and NetBase for Your ALLBASE DBEs
	Melanie Lallier - Quest Software, Inc.	
3019		Taking Steps to Client-Server with ODBC
	M.B. Foster - M.B. Foster Associates	
4000		The Changing Face of PC-to-UNIX Connectivity
	Matt Hulett - Walker Richer & Quinn, Inc.	
4001		The Smooth Path to Client-Server Computing
	Rolf Brandt - INFOSOFT GmbH	
4002		Integrating UNIX Workstations, Personal Computers and Macs
	Brad Feazell - Dril-Quip, Inc.	
4003		Integrating PC LANS and UNIX Servers
	Ann Hewitt - Hewlett-Packard Co.	
4004		Integrating PCs into the UNIX Environment
	Matt Hulett - Walker Richer & Quinn, Inc.	
4006		NetWare MPE, HP-UX Printing
	Cathy Gunn - Hewlett-Packard Co.	
4007		Understanding and Using X Windows
	Matt Hulett - ConAm Corporation	
4008		Client-Server with Your HP 3000: A Case Study
	David Wright - Tufts Health Plans, Inc.	
4009		Processing Multiple Inputs with the X Toolkit Intrinsic
	Donald Parker - E-Systems, Inc.	
4010		Using Windows NT as a Desktop in an Internetworked, Client-Server Environment
	Robert Jones - Hewlett-Packard Co.	
4011		Client-Server: Plug & Play or Crash & Burn
	Ralph Bagen - Aircast Inc.	
4012		Integrating Windows NT Clients into an HP Server Environment: A Case Study
	Robert Jones - Hewlett-Packard Co.	
4013		Integrating HP OpenMail and HP 9000 Fax
	Tony Jones - Hewlett-Packard Co.	

Index by Paper Number

- 4014 Evaluating and Implementing Office Automation Solutions
Eric Kowalski, Ken Smith - Higley & Company, Inc.
- 4015 Using GIS for Business Applications
Tony Burns - ESRI
- 4016 HP-UX to PC Porting Tools and Techniques
Arthur Walasek - Hewlett-Packard Co.
- 5000 Enterprise Collaboration Through Multimedia
Charlie Fernandez - Hewlett-Packard Co.
- 5001 Networking and ATM
Richard Nelson - Vanguard Technologies, LLC
- 5002 Using PC, HP 3000, HP 9000, and IBM Machines in Distributed Client-Server Applications
Michael Wang - Hewlett-Packard Co.
- 5004 The Role of Enterprise Network Services in the Emerging Global Area Network
Bill Johnson - Banyan Systems
- 5005 Automating FTP Sessions to an MVS Host
Rich Case - Nielsen Media Research
- 5007 A Global HP PC Common Operating Environment Implementation
Brandt Faatz - Hewlett-Packard Co.
- 5008 An Introduction to TCP/IP Network Security
Douglas Conorich - Raxco, Inc.
- 6000 The OSF/Motif Style Guide and You: Developing and Certifying OSF/Motif Applications
Natasha Flaherty - Oracle Corporation
- 6001 Evolving ODBMS Standards
Andrew Wade - Objectivity, Inc.
- 6002 ABCs of ODBC
Dirk Huizenga - DISC
- 6003 Porting Qedit from MPE to HP-UX
David Greer - Robelle Consulting Ltd.
- 6004 On-Line Analytical Processing (OLAP): Spoken Here
Gary Farner - Information Resources, Inc.
- 6005 Supporting Your Mission Critical Environment
Deborah Nelson - Hewlett-Packard Co.
- 6006 Retrofitting Your HP 3000 Applications to the MS-Windows Standard
Ross Hopmans, Joe Grimm - Brant Technologies & MiniSoft Inc.
- 6007 Tuning the HP-UX High Performance File System
Glen Johnson - CSI Computer Solutions, Inc.
- 6008 Open Systems Air Traffic Control
Robert Chew - EG & G
- 6009 ODBC: Visual Basic Application Development with HP/ALLBASE
Jeff Zalkind - Microsoft
- 6010 Client-Server and Rightsizing, Open Systems Challenges, and Solutions
Marlene Nesson - Information Builders, Inc.
- 6011 System Administration of DCE
Jeffery Hamilton - CellularOne
- 6012 Meeting ANSI Standards for Software Configuration Management
James Lofink - Operations Control Systems
- 6013 The New Computer Graphics Metafile Standard
Lofton Henderson - Henderson Software, Inc.
- 6014 HP-UX 9.X Operating System Update
Paul Perlmutter - Hewlett-Packard Co.
- 7000 Taking the Mystery Out of RTE-A System Generations
Bill Chu - Hewlett-Packard Co.
- 7001 HP-RT 2.0: A Mile High Above the Rest
Eric Mostardi, Rita Sandhu - Hewlett-Packard Co.
- 7002 Porting from HP-UX to HP-RT
Frances Huang - Hewlett-Packard Co.

Index by Paper Number

- 7003 RTE to HP-UX Migration Tools: An Update
Becky Carroll - Hewlett-Packard Co.
- 7004 Use of HP Optical Disk Libraries with HP 1000 Computers
Clifford Buffett - Warner Lambert
- 7005 Floating-point Management in the HP-RT Kernel
Ren Wang, George Anzinger - Hewlett-Packard Co.
- 7006 Client-Server Process Control Over a WAN
Kevin Wong - East Bay Municipal Utility District
- 7007 Turbo Your IMAGE
Donald Clapp
- 8000 Systems Management
Steven Cole - Northern Telecom
- 8001 Closing the Holes in HP-UX
Joe Pennell - Paychex, Inc.
- 8002 Building a Tool for Remote UNIX Administration
Bobby Siu - Hewlett-Packard Co.
- 8003 High Availability, Mirroring, RAID: A How-to System
Raymond Riedel - Hewlett-Packard Co.
- 8004 The HP 3000 Workload Manager
Susan Campbell, Doug Perry - Hewlett-Packard Co.
- 8005 Managing a POSIX 3000 System
Jeff Vance - Hewlett-Packard Co.
- 8006 UNIX Boot Disasters: Planning Your Alternatives
Dennis McClure - Hewlett-Packard Co.
- 8007 Effective Network Management on MPE-V
Ross Martin, George Clark - Hewlett-Packard Co.
- 8008 A System Manager's Look at MPE/iX 5.0
Kevin Cooper - Hewlett-Packard
- 8009 De-Centralizing Printing with the Associate Command
Royden Somerville - University of Notre Dame
- 8010 MPE/iX Debug and Dump Analysis
Michael Hornsby - Beechglen Development, Inc.
- 8011 Life with a "Cluster" of HP 3000s
Stephen Thomas - Graco Children's Products, Inc.
- 8012 Help! My PC's Down and I Can't Get It Up!!
Peggy Beall
- 8015 Prepare Your MPE/iX System Before Update to 5.0
Al Dulancy - Hewlett-Packard Co.
- 8016 Open Systems Backup: How to Maximize Performance
Andrew Ibbotson, Uwe Hinrichs - HICOMP Storage Technologies GmbH
- 8017 Standalone Clusters: A Model of Server- Client Workstation Clustering Using NFS
Michael Williams - Hewlett-Packard Co.
- 8020 A Comparison of Hardware and Software Data Compression
Husni Sayed - IEM, Inc.
- 8022 Operational Management and Control in Distributed Environments
Timothy Early - Open Systems Integration, Inc.
- 8023 Security in a Distributed Environment
Michel Kabay - Jinbu Corporation

Index by Author

- Flaherty, Natasha The OSF/Motif Style Guide and You: Developing and Certifying
6000, Oracle Corporation OSF/Motif Applications
- Fontana, Stephen Thinking About an Image Management System: Think Again and Again
2013, Software Systems Technology
- Foster, M.B. Taking Steps to Client-Server with ODBC
3019, M.B Foster Associates
- Frank, Bruce Does Your HP "Talk" Back
3007, Frank Solutions
- Gardner, Mark Indexing: When, When Not, and How
3000, Bradmark Technologies, Inc.
- Gebhardt, John The New Computer Graphics Metafile Standard
6013, InterCAP Graphics Systems
- Gillis, Dan The Client-Server Headache: Performance, and How to Get Relief
1010, DISC
- Greer, David Porting Qedit from MPE to HP-UX
6003, Robelle Consulting Ltd.
- Grewal, Bashinder, Gollabinnie, Kumar Leveraging Support Alliances
2011, Hewlett-Packard Co.
- Gunn, Cathy NetWare MPE, HP-UX Printing
4006, Hewlett-Packard Co.
- Gustafson, Mary Ann Evolving Your HP IM Investment
3009, Hewlett-Packard Co.
- Guzenda, Leon The ODBMS Role in Client-Server Computing
3016, Objectivity, Inc.
- Hamilton, Jeffery System Administration of DCE
6011, CellularOne
- Hepler, James Criteria for RDBMS Selection
3001, Hewlett-Packard Co.
- Hewitt, Ann Integrating PC LANS and UNIX Servers
4003, Hewlett-Packard Co.
- Hopmans, Ross, Grimm, Joe Retrofitting Your HP 3000 Applications to the MS-Windows
6006, Brant Technologies & MiniSoft Inc. Standard
- Hornsby, Michael MPE/iX Debug and Dump Analysis
8010, Beechglen Development, Inc.
- Huang, Frances Porting from HP-UX to HP-RT
7002, Hewlett-Packard Co.
- Huizenga, Dirk ABCs of ODBC
6002, DISC
- Hulett, Matt Integrating PCs into the UNIX Environment
4004, Walker Richer & Quinn, Inc.
- Hulett, Matt The Changing Face of PC-to-UNIX Connectivity
4000, Walker Richer & Quinn, Inc.
- Hupe, Rick Telephone Responsibilities: Has Your MIS Taken Over Yet?
2005, Telnomics Inc.
- Ibbotson, Andrew, Hinrichs, Uwe Open Systems Backup: How to Maximize Performance
8016, HICOMP Storage Technologies GmbH
- Johnson, Bill The Role of Enterprise Network Services in the Emerging Global
5004, Banyan Systems Area Network
- Johnson, Glen Tuning the HP-UX High Performance File System
6007, CSI Computer Solutions, Inc.
- Jones, Robert Integrating Windows NT Clients into an HP Server Environment:
4012, Hewlett-Packard Co. A Case Study
- Jones, Robert Using Windows NT as a Desktop in an Internetworked, Client-Server
4010, Hewlett-Packard Co. Environment
- Jones, Tony Integrating HP OpenMail and HP 9000 Fax
4013, Hewlett-Packard Co.

Index by Author

- | | |
|--|--|
| Kabay, Michel
8023, Jinbu Corporation | Security in a Distributed Environment |
| Kennedy, Mike
2010, Excalibur Technologies | Turning Multimedia Data Into Information |
| Kowalski, Eric, Smith, Ken
4014, Higley & Company, Inc. | Evaluating and Implementing Office Automation Solutions |
| Kramer, John
5901, Computer Data Systems, Inc. | Speaking the Language of TCP/IP and LANs |
| LaHiff, Martin
3017, Hewlett-Packard Co. | IMAGE/SQL At Work |
| Lallier, Melanie
3018, Qest Software, Inc. | Using Replicate and NetBase for Your ALLBASE DBEs |
| Larson, Orly
1000, Hewlett-Packard Co. | The Subject is Objects |
| Lawson, Roger
2009, Proactive Systems Ltd. | Future Strategies for DP Managers |
| Lloyd, Ben
1003, Hewlett-Packard Co. | Client-Server RAD: A Case Study |
| Lofink, James
6012, Operations Control Systems | Meeting ANSI Standards for Software Configuration Management |
| Losch, Roy
3013, Hewlett-Packard Co. | Working with Allbase/SQL PC API |
| Martin, Ross, Clark, George
8007, Hewlett-Packard Co. | Effective Network Management on MPE-V |
| McClure, Dennis
8006, Hewlett-Packard Co. | UNIX Boot Disasters: Planning Your Alternatives |
| Meyer, Alan
1006, Hewlett-Packard Co. | Migrating COBOL to HP-UX |
| Mostardi, Eric, Sandhu, Rita
7001, Hewlett-Packard Co. | HP-RT 2.0: A Mile High Above the Rest |
| Nelson, Deborah
6005, Hewlett-Packard Co. | Supporting Your Mission Critical Environment |
| Nelson, Richard
5001, Vanguard Technologies, LLC | Networking and ATM |
| Nesson, Marlene
6010, Information Builders, Inc. | Client-Server and Rightsizing, Open Systems Challenges, and Solutions |
| Nishimoto, Alvina, Sartain, Jim
3002, Hewlett-Packard Co. | An Overview of IMAGE/SQL |
| Norman, Liz
3004, Hewlett-Packard Co. | O.K., It's Installed - Now What? IMAGE/SQL Future |
| O'Brien, Terry
1016, DISC | Visual Basic for the MIS Professional (How to Develop High Performance Client-Server Applications) |
| O'Brien, Terry
3015, DISC | Benefits of DataWarehousing |
| Parker, Donald
4009, E-Systems, Inc. | Processing Multiple Inputs with the X Toolkit Intrinsics |
| Pennell, Joe
8001, Paychex, Inc. | Closing the Holes in HP-UX |
| Perlmutter, Paul
6014, Hewlett-Packard Co. | HP-UX 9.X Operating System Update |
| Phillips, Andrew
1015, Hewlett-Packard Co. | Getting the Most out of HP-UX |
| Podkomorski, John
2014, Hewlett-Packard Co. | Data Center Consolidation at Hewlett-Packard |
| Rego, Alfredo
3014, Adager | The Database Administrator (DBA): A Ruthless Juggler of Reality, Algorithms and Data Structures |

APPLICATION DEVELOPMENT

- 1000 The Subject is Objects
- 1002 Orly Larson - Hewlett-Packard Co. Delivering Client-Server Functions Today
- 1003 Riz Shakir - SSA Client-Server RAD: A Case Study
- 1004 Ben Lloyd - Hewlett-Packard Co. A Comparison of Distributed Application Technologies
- 1005 Gerald Duggan - Hewlett-Packard Co. Using High-Performance Real-Time I/O with HP-UX
- 1006 Robert Cleary - KineticSystems Corporation Migrating COBOL to HP-UX
- 1007 Alan Meyer - Hewlett-Packard Co. A Practical Guide for Integrating Imaging into your Existing Business Applications
- 1008 John Carney - WANG Future CASE: Enabling End Users to Customize Applications
- 1009 Ken Robinson - Auto-trol Technology Corp. HP SourceReader UX - HP 3000 Debugging, Development, and Source Understanding Tool of the Future
- 1010 Gary Robillard - Hewlett-Packard Co. The Client-Server Headache: Performance, and How to Get Relief
- 1011 Dan Gillis - DISC How to Effectively Select Software Development Tools
- 1014 Padma Sreenivasan - Hewlett-Packard Co. How to Design GUI's Effectively Using GUI's Standards and Specs
- 1015 Padma Sreenivasan - Hewlett-Packard Co. Getting the Most out of HP-UX
- 1016 Andrew Phillips - Hewlett-Packard Co. Visual Basic for the MIS Professional (How to Develop High Performance Client-Server Applications)
- Terry O'Brien - DISC

BUSINESS MANAGEMENT

- 2000 The HP Information Systems Marketplace Today: What's Hot and What's Not
- 2001 Diane Amos - Amos & Associates, Inc. Re-Examining Re-Engineering
- 2002 Feyzi Fatehi - Hewlett-Packard Co. Network Outsourcing: A Key to Corporate Reengineering
- 2003 Ernest Eugster - Random Access Legal Issues in EDI
- 2004 Christopher Seline - Formosa Transnational Attorneys at Law Source Code Escrow: A Critical Business and Legal Issue
- 2005 Bea Strickland - SourceFile Telephone Responsibilities: Has Your MIS Taken Over Yet?
- 2006 Rick Hupe - Telenomics Inc. Imaging and Document Management
- 2009 David Sborov - NSD, Inc. Future Strategies for DP Managers
- 2010 Roger Lawson - Proactive Systems Ltd. Turning Multimedia Data Into Information
- 2011 Mike Kennedy - Excalibur Technologies Leveraging Support Alliances
- 2012 Bashinder Grewal, Kumar Gollabinnie - Hewlett-Packard Co. Rightsizing to Client-Server Applications
- 2013 John Woolsoncroft - Concepts Dynamic, Inc. Thinking About an Image Management System: Think Again and Again
- Stephen Fontana - Software Systems Technology

Index by Category

- 2014 Data Center Consolidation at Hewlett-Packard
John Podkomorski - Hewlett-Packard Co.

DATABASE TECHNOLOGY

- 3000 Indexing: When, When Not, and How
Mark Gardner - Bradmark Technologies, Inc.
- 3001 Criteria for RDBMS Selection
James Hepler - Hewlett-Packard Co.
- 3002 An Overview of IMAGE/SQL
Alvina Nishimoto, Jim Sartain - Hewlett-Packard Co.
- 3003 Introduction to IMAGE/SQL
Denys Beauchemin - HICOMP AMERICA, Inc.
- 3004 O.K., It's Installed - Now What? IMAGE/SQL Future
Liz Norman - Hewlett-Packard Co.
- 3005 IMAGE/SQL and Indexing
Larry Boyd - Bradmark Technologies, Inc.
- 3007 Does Your HP "Talk" Back
Bruce Frank - Frank Solutions
- 3008 The New Relational Database: IMAGE/SQL
Bradley Tashenberg - Bradmark Technologies, Inc.
- 3009 Evolving Your HP IM Investment
Mary Ann Gustafson - Hewlett-Packard Co.
- 3013 Working with Allbase/SQL PC API
Roy Losch - Hewlett-Packard Co.
- 3014 The Database Administrator (DBA): A Ruthless Juggler of Reality, Algorithms
and Data Structures
Alfredo Rego - Adager
- 3015 Benefits of Data Warehousing
Terry O'Brien - DISC
- 3016 The ODBMS Role in Client-Server Computing
Leon Guzenda - Objectivity, Inc.
- 3017 IMAGE/SQL At Work
Martin LaHiff - Hewlett-Packard Co.
- 3018 Using Replicate and NetBase for Your ALLBASE DBEs
Melanie Lallier - Quest Software, Inc.
- 3019 Taking Steps to Client-Server with ODBC
M.B. Foster - M.B. Foster Associates

DESKTOP APPLICATIONS & INTEGRATION

- 4000 The Changing Face of PC-to-UNIX Connectivity
Matt Hulett - Walker Richer & Quinn, Inc.
- 4001 The Smooth Path to Client-Server Computing
Rolf Brandt - INFOSOFT GmbH
- 4002 Integrating UNIX Workstations, Personal Computers and Macs
Brad Feazell - Dril-Quip, Inc.
- 4003 Integrating PC LANS and UNIX Servers
Ann Hewitt - Hewlett-Packard Co.
- 4004 Integrating PCs into the UNIX Environment
Matt Hulett - Walker Richer & Quinn, Inc.
- 4006 NetWare MPE, HP-UX Printing
Cathy Gunn - Hewlett-Packard Co.
- 4007 Understanding and Using X Windows
Matt Silberstein - ConAm Corporation
- 4008 Client-Server with your HP 3000: A Case Study
David Wright - Tufts Health Plans, Inc.

Index by Category

- 4009 Donald Parker - E-Systems, Inc. Processing Multiple Inputs with the X Toolkit Intrinsic
- 4010 Using Windows NT as a Desktop in an Internetworked, Client-Server Environment
Robert Jones - Hewlett-Packard Co.
- 4011 Client-Server: Plug & Play or Crash & Burn
Ralph Bagen - Aircast Inc.
- 4012 Integrating Windows NT Clients into an HP Server Environment: A Case Study
Robert Jones - Hewlett-Packard Co.
- 4013 Integrating HP OpenMail and HP 9000 Fax
Tony Jones - Hewlett-Packard Co.
- 4014 Evaluating and Implementing Office Automation Solutions
Eric Kowalski, Ken Smith - Higley & Company, Inc.
- 4015 Using GIS for Business Applications
Tony Burns - ESRI
- 4016 HP-UX to PC Porting Tools and Techniques
Arthur Walasek - Hewlett-Packard Co.

NETWORKING AND INTEROPERABILITY

- 5000 Enterprise Collaboration Through Multimedia
Charlie Fernandez - Hewlett-Packard Co.
- 5001 Networking and ATM
Richard Nelson - Vanguard Technologies, LLC
- 5002 Using PC, HP 3000, HP 9000, and IBM Machines in Distributed Client-Server
Michael Wang - Hewlett-Packard Co. Applications
- 5004 The Role of Enterprise Network Services in the Emerging Global Area Network
Bill Johnson - Banyan Systems
- 5005 Automating FTP Sessions to an MVS Host
Rich Case - Nielsen Media Research
- 5007 A Global HP PC Common Operating Environment Implementation
Brandt Faatz - Hewlett-Packard Co.
- 5008 An Introduction to TCP/IP Network Security
Douglas Conorich - Raxco, Inc.
- 6000 The OSF/Motif Style Guide and You: Developing and Certifying OSF/Motif
Natasha Flaherty - Oracle Corporation Applications

OPEN SYSTEMS AND STANDARDS

- 6001 Evolving ODBMS Standards
Andrew Wade - Objectivity, Inc.
- 6002 ABCs of ODBC
Dirk Huizenga - DISC
- 6003 Porting Qedit from MPE to HP-UX
David Greer - Robelle Consulting Ltd.
- 6004 On-Line Analytical Processing (OLAP): Spoken Here
Gary Farner - Information Resources, Inc.
- 6005 Supporting Your Mission Critical Environment
Deborah Nelson - Hewlett-Packard Co.
- 6006 Retrofitting Your HP 3000 Applications to the MS-Windows Standard
Ross Hopmans, Joe Grimm - Brant Technologies & MiniSoft Inc.
- 6007 Tuning the HP-UX High Performance File System
Glen Johnson - CSI Computer Solutions, Inc.
- 6008 Open Systems Air Traffic Control
Robert Chew - EG & G
- 6009 ODBC: Visual Basic Application Development with HP/ALLBASE
Jeff Zalkind - Microsoft

Index by Category

- 6010 Client-Server and Rightsizing, Open Systems Challenges, and Solutions
Marlene Neason - Information Builders, Inc.
- 6011 System Administration of DCE
Jeffery Hamilton - CellularOne
- 6012 Meeting ANSI Standards for Software Configuration Management
James Lofink - Operations Control Systems
- 6013 The New Computer Graphics Metafile Standard
Loflon Henderson - Henderson Software, Inc.
- 6014 HP-UX 9.X Operating System Update
Paul Perlmutter - Hewlett-Packard Co.

REAL-TIME

- 7000 Taking the Mystery Out of RTE-A System Generations
Bill Chu - Hewlett-Packard Co.
- 7001 HP-RT 2.0: A Mile High Above the Rest
Eric Mostardi, Rita Sandhu - Hewlett-Packard Co.
- 7002 Porting from HP-UX to HP-RT
Frances Huang - Hewlett-Packard Co.
- 7003 RTE to HP-UX Migration Tools: An Update
Becky Carroll - Hewlett-Packard Co.
- 7004 Use of HP Optical Disk Libraries with HP 1000 Computers
Clifford Buffett - Warner Lambert
- 7005 Floating-point Management in the HP-RT Kernel
Ren Wang, George Anzinger - Hewlett-Packard Co.
- 7006 Client-Server Process Control Over a WAN
Kevin Wong - East Bay Municipal Utility District
- 7007 Turbo Your IMAGE
Donald Clapp

SYSTEMS MANAGEMENT

- 8000 Systems Management
Steven Cole - Northern Telecom
- 8001 Closing the Holes in HP-UX
Joe Pennell - Paychex, Inc.
- 8002 Building a Tool for Remote UNIX Administration
Bobby Siu - Hewlett-Packard Co.
- 8003 High Availability, Mirroring, RAID: A How-to System
Raymond Riedel - Hewlett-Packard Co.
- 8004 The HP 3000 Workload Manager
Susan Campbell, Doug Perry - Hewlett-Packard Co.
- 8005 Managing a POSIX 3000 System
Jeff Vance - Hewlett-Packard Co.
- 8006 UNIX Boot Disasters: Planning Your Alternatives
Dennis McClure - Hewlett-Packard Co.
- 8007 Effective Network Management on MPE-V
Ross Martin, George Clark - Hewlett-Packard Co.
- 8008 A System Manager's Look at MPE/iX 5.0
Kevin Cooper - Hewlett-Packard
- 8009 De-Centralizing Printing with the Associate Command
Royden Somerville - University of Notre Dame
- 8010 MPE/iX Debug and Dump Analysis
Michael Hornsby - Beechglens Development, Inc.
- 8011 Life with a "Cluster" of HP 3000s
Stephen Thomas - Graco Children's Products, Inc.

Index by Category

- 8012 Help! My PC's Down and I Can't Get It Up!!
Peggy Beall
- 8015 Prepare Your MPE/iX System Before Update to 5.0
Al Dulaney - Hewlett-Packard Co.
- 8016 Open Systems Backup: How to Maximize Performance
Andrew Ibbotson, Uwe Hinrichs - HICOMP Storage Technologies GmbH
- 8017 Standalone Clusters: A Model of Server- Client Workstation Clustering using NFS
Michael Williams - Hewlett-Packard Co.
- 8020 A Comparison of Hardware and Software Data Compression
Husni Sayed - IEM, Inc.
- 8022 Operational Management and Control in Distributed Environments
Timothy Early - Open Systems Integration, Inc.
- 8023 Security in a Distributed Environment
Michel Kabay - Jinbu Corporation

1000
THE SUBJECT IS OBJECTS

Orly Larson
Hewlett-Packard Company
19091 Pruneridge Avenue
Cupertino, California 95014
(408)447-1197

ABSTRACT

"Object orientation" (OO) is a new buzzword that promises to become the next "eureka" in information systems development. MIS management and their software developers are under increasing pressure to improve productivity, increase software quality and reduce implementation time. Traditional software development methods are not always able to meet these increasingly heavy demands. Object orientation is getting attention as a viable alternative.

This paper reports on the growing body of knowledge about object-oriented technologies. It begins by reviewing some of the critical challenges facing today's enterprises, followed by the definitions, basic mechanisms and key concepts associated with object-oriented systems. Next, it explores various types of applications that benefit from this technology. The potential benefits and the potential concerns will be addressed, followed by the impact object-oriented technologies may have on data administration and systems development in the 90's.

INTRODUCTION

Each decade one or two key advances emerge and change the practice of software development. Object-oriented systems and methods are rapidly entering the mainstream of software engineering and systems development. Leading consultants are heralding object-oriented approaches as one of the most important trends to affect businesses in the 90's.

Software is currently lagging behind hardware capabilities and the lag is increasing. There is general agreement that conventional software tools and techniques are rapidly becoming inadequate as software systems grow larger and increasingly more complex. Also, a consensus is building that the new paradigm

THE SUBJECT IS OBJECTS

1000-1

of object orientation may help control complexity and harness the expanding system environment into more useful and exciting applications.

Applications will need to satisfy more sophisticated requirements, use more complex data structures and architectures, and be delivered to an increasingly broad base of users. Software developers will have to increase their capacity to build, extend, and maintain complex, large-scale systems including their existing legacy systems. This requires that software be more flexible and easier to use.

Many of today's software development processes are out-of-date, with programmers still functioning like craftsmen. They build unique, noninterchangeable components and assemble them by hand, and then they struggle over time to understand the code generated by their predecessors and to extend and refine that software. As powerful computers pervade the lives of more and more people, the inability to deliver and maintain equally powerful software is an increasingly visible problem.

WHAT IS OBJECT ORIENTATION?

There is no single precise rule for describing or identifying object orientation. Rather, a collection of concepts together describes this new paradigm for software construction. In this new paradigm objects and classes are the building blocks, while methods, messages, and inheritance produce the primary mechanisms. Historically, creating a software program involved creating processes that act on a separate set of data. Object orientation changes the focus of the programming process from procedures to objects. Objects are self-contained modules that include both the data and the procedures that act on that data. The procedures contained within the object take on a new name, methods. Objects are activated by messages. Objects that have a common use are grouped together in a class, and new classes can be created that inherit the procedures and data from classes already built. This inheritance enables the programmer to reuse existing classes and to program only the differences. This provides for a new level of abstraction, with prebuilt libraries of classes and even prebuilt application specific class libraries or frameworks. Object orientation is important for the software development challenges of the 90's. This paradigm will improve the software development process and will cause new and better applications to evolve. It's promises will be delivered incrementally and across a broad range of technologies and will permeate the next generation of software architectures.

BASIC MECHANISMS

THE SUBJECT IS OBJECTS

1000-2

OBJECTS

Webster's New Collegiate Dictionary defines an object as "Something that is capable of being seen, touched or otherwise sensed." Grady Booch, in his book, "Object-Oriented Design with Applications", defines an object as "Something you can do things to. An object has state, behavior, and identity; the structure and behavior of similar objects are defined in their common class". David Taylor, in his book "Object-Oriented Technology: A Managers Guide" defines an object as "A software packet containing a collection of related data and methods for operating on that data".

Within objects reside the data of conventional computing languages, such as numbers, arrays, strings and records, as well as any functions, instructions, or subroutines that operate on them.

MESSAGES

Objects have the ability to act. Action occurs when an object receives a message, that is, a request, asking the object to behave in some way. When object-oriented programs execute, objects are receiving, interpreting, and responding to messages from other objects.

METHODS

Procedures called methods reside in the object and determine how the object acts when it receives a message. Methods may also send messages to other objects requesting action or information.

CLASS

Many different objects may act in very similar ways. A class is a description of a set of nearly identical objects. It is a category or collection of objects that share a common structure and a common behavior but contain different data.

INSTANCE

An instance is a term used to refer to an object that is a member of a class. Instance and object are used interchangeably.

INHERITANCE

Inheritance is the mechanism for automatically sharing methods and data among classes, subclasses, and objects. A powerful mechanism whereby classes can make use of the methods and variables defined in all classes above them on their branch of the hierarchy. Inheritance allows programmers to program only what is different from previously defined classes.

KEY CONCEPTS

ENCAPSULATION

Encapsulation is the process of hiding all of the details of an object such as its data (instance variables) and procedures (methods). This is also referred to as "information hiding".

ABSTRACTION

Abstraction is the process of creating a "superclass" by extracting common qualities or general characteristics from more specific classes or objects. Each level of abstraction makes the job of programming easier because it makes more reusable code available.

PERSISTENCE

Persistence refers to the permanence of an object, that is, the amount of time for which it is allocated space and remains accessible in the computer's memory. The object may continue to exist even after its creator ceases to exist. Objects stored permanently are termed persistent.

POLYMORPHISM

Objects act in response to the messages they receive. The same message can result in completely different actions when received by different objects. This phenomenon is referred to as polymorphism.

OBJECT-ORIENTED APPLICATIONS

Object-oriented applications will inspire users to think differently about the nature of computing. Programs in an object-oriented environment will be transparent. Object-oriented frameworks will facilitate simulating and constructing user-specific solutions. Objects will be shared in networking environments to

distribute information within a work group or to parcel out tasks for distributed processing.

Object orientation is favored for applications that are characterized by complex processes and complex data manipulation. Applications in the following categories are classic candidates for enhancement through object orientation:

- ◆ Computer Aided Software Engineering (CASE)
- ◆ Computer Aided Instruction (CAI)
- ◆ Computer Integrated Manufacturing (CIM)
- ◆ Computer Aided Publishing (CAP)
- ◆ CAD/CAM/CAE Systems
- ◆ Document Management Systems
- ◆ Executive Information Systems
- ◆ Geographic Information Systems
- ◆ Graphics, Handling ICONS
- ◆ Health Care
- ◆ Image Storage Management
- ◆ Knowledge Based Systems
- ◆ Multimedia
- ◆ Manufacturing Production Control
- ◆ Manufacturing Requirements Planning
- ◆ Military Command and Control Decision Support
- ◆ Network Management
- ◆ Real Estate Systems
- ◆ Configuration and Version Management
- ◆ Telecommunications Routing Systems
- ◆ Visual Programming

Object-oriented applications will most likely gain in both presence and popularity.

POTENTIAL BENEFITS

Before managers can make informed decisions about adopting a new technology, the advantages of this technology must be translated into measurable benefits. Object-oriented programming improves not only the software development process but also the flexibility and utility of the resulting software. The design process becomes more intuitive as elements of the software correspond to elements in the application's real world domain. The programming process itself encourages teamwork, code reuse, and code polishing.

Reusability is the key to increasing productivity in the face of increasing complexity. The key breakthrough in object technology is the ability to build large programs from lots of small, prefabricated ones. In addition to the increased productivity that results from reusability, using object-oriented technology can result in greater reliability because it reduces the risk of human error. Program structures remain intact, and change propagates naturally through the hierarchy of classes.

Flexibility is also a trademark of object orientation. Programmers are freed from the constraints of preestablished data types, allowing extensions of application functionality and bridging of heterogeneous applications.

Adaptability of object-oriented programs may well turn out to be the most crucial advantage of object-oriented technology. No matter how perfectly crafted, a program is useless if it doesn't meet current needs and the needs of users are changing at an ever increasing rate.

Faster development of applications is another benefit and is a result of all the programming effort that is reused from existing objects and all the design work that went into an existing model of a process.

Increased scalability is another significant benefit of object orientation. Given its improved modularization, it is especially well suited to developing large-scale systems.

Large systems are easier to build and maintain when you build them out of subsystems that can be developed and tested independently.

POTENTIAL CONCERNS

While object-oriented technology promises many benefits, there are some valid concerns about its ability to deliver those benefits. Most of these concerns have to do with temporary limitations and should disappear as the technology and its market mature.

The maturity of the technology itself is a concern to many potential developers. It is not yet a completely stable technology and many companies are not comfortable being "pioneers".

Standards are still evolving and the lack of accepted standards raises concerns about the difficulty of moving programs from one development environment to another and mixing and matching objects and classes from different vendors. Standards are on the way. The Object Management Group (OMG) was formed by a consortium of the major vendors of several object-oriented products. The purpose of this group is to promote the adoption of standards and the interchangeability of objects. In addition, the American National Standards Institute (ANSI) has an ODBMS committee, but no standards have been officially approved.

There is also a shortage of tools for application development. These tools include programs to assist in the design of objects and the management of libraries of reusable objects.

Performance of object-oriented applications is a concern and the speed of object-oriented systems will improve as the technology matures.

The object-oriented approach has a tremendous amount of potential and companies should explore this new technology, and check out the benefits for themselves.

EVOLUTION OR REVOLUTION?

Many organizations have invested heavily in existing non-relational and relational database management technology. The majority of these companies do not want to replace their existing databases and applications. The need to integrate these "legacy" databases with each other and with new systems is an important factor in the future evolution of data management.

There are clearly risks associated with getting into this technology too soon. But there are risks associated with waiting as well. The companies that begin the transition now will enjoy an important competitive advantage while the others strive to catch up.

The most prudent strategy is to avoid the extremes of ignoring the strategy or committing vital systems to it. Instead, companies can make a modest investment in a pilot program to gain first-hand experience with object-oriented development. This approach allows a company to reach its own conclusions about the value of the technology, and places the company well down the experience curve if it converts in the future.

THE FUTURE

Object-oriented technology will provide the clarity and flexibility essential to the successful development of complex systems. Today's applications do not offer the consistency and flexibility needed to make the computing environment more productive for users. Object orientation will provide environments in which users can communicate among applications and navigate easily over distributed, heterogeneous architectures.

In the near future object orientation will deliver the most benefits to three categories of programmers: power users, general business programmers and system developers. The most dramatic near-term benefits will be for system developers who both require and embrace this development approach and evolving tools to implement the increasingly complex and potentially innovative software of the 1990's. Over time, object-oriented technology will begin to have increasing impact on general business programmers and power users. Carefully designed object libraries will become available to support less sophisticated programmers who want to assemble applications quickly from prefabricated objects.

The vision of the future extends beyond the arrival of object-oriented system components development tools and standards. In the future, users will have the power and flexibility to design their own applications just by snapping together the necessary objects. With objects, building applications will be a process of tailoring and linking reusable modules. Object-oriented software architectures will mature in the 1990's. The transition to these new architectures is underway, marked by the arrival of object-oriented languages, databases, interfaces, operating systems and development environments. New types of data, distributed processing, multimedia applications, and end user computing are driving forces in the implementation of the object-oriented software environment.

There has been a great deal of progress implementing object-oriented systems. Today's graphical user interfaces have acquainted users with object manipulation. Among object-oriented languages, C++ has become the de facto standard. Object-oriented extensions are also being implemented in most popular commercial languages. Object-oriented development environments such as Hewlett-Packard's Softbench provide examples of object-oriented programming and applications. Operating systems are also being extended to support interoperability among object-based applications. Over the next decade, the difference between the old and the new will become increasingly obvious to both programmers and users. When the object-oriented future is fully delivered, this

natural, intuitive paradigm will be strongly embraced and will provide benefits to programmers and users alike.

OBJECT TECHNOLOGY FROM HEWLETT-PACKARD

In 1991, Hewlett-Packard announced HP OpenODB, the most advanced, commercial object-oriented database management system for large, multi-user environments. It is designed to enable new, complex business applications to be developed and maintained at a fraction of current costs.

HP OpenODB is based upon the Iris ODBMS prototype developed by HP Labs, starting in 1984. Using Iris, HP has worked extensively with customers and universities in evaluating ODBMS requirements. HP OpenODB is targeted at complex, commercial applications such as Geographic Information Systems (GIS), telecommunications systems and heterogenous information integration such as Executive Information Systems and Computer Integrated Manufacturing, whose needs may not be met by current database products such as TurboIMAGE and ALLBASE/SQL.

HP OpenODB uses a relational database as its storage manager and presents an object-oriented model to developers. It is a hybrid approach that allows integration of existing legacy systems including applications written in C, COBOL, FORTRAN, PASCAL, ADA, etc. This architecture provides a robust storage environment with the DBMS capabilities commercial users have come to expect. HP is currently using ALLBASE/SQL as the storage manager for HP OpenODB for performance and stability of data; however, the architecture allows HP OpenODB to be ported to other relational DBMS's for portability to non HP hardware. This combination is unique in the industry.

A Key benefit of HP OpenODB is that users don't have to abandon their current software or data to work with it. It includes an object-oriented structured query language (OSQL) and external functions that will retrieve information regardless of its format or whether it is stored inside or outside of HP OpenODB.

Object orientation is another phase in the evolution of computing and is an important step towards the vision of "Information At Your Fingertips". Developers must take advantage of the many benefits of this technology while at the same time deal with the hurdles that this technology poses. Object-oriented development environments must play a large role in reducing the learning curve and make object-oriented programming a highly productive process. Object-oriented products are here today and the commercialization of

object-oriented technology is increasing rapidly. Object based architectures lend themselves to the creation of a much richer information environment. Digitized voice, music, video clips and animation will begin to populate our information systems. Object database systems are currently viable for commercial projects and will be widely adopted by the mid to late 1990's.

REFERENCES

Booch, Grady. *Object-Oriented Design With Applications*. Redwood City, CA: The Benjamin/Cummings Publishing Company, Inc, 1990.

DeCastilhos, Margie. *Getting Started With Object-Oriented Databases*. INTERACT Magazine (Hewlett-Packard User Group Publication). pages 102-113, March 1992.

Hewlett-Packard. 1990. *CASE and Objects-A Primer (An Introduction to Object-Oriented Concepts for Software Engineering)*, Palo Alto, CA: HP Literature # 5952-2624.

Taylor, David A., *Object-Oriented Technology: A Manager's Guide*. Reading, MA. Addison-Wesley, 1991. (A very good overview of object-oriented technologies for managers)

Winblad, Ann L., et al. *Object-Oriented Software*. Reading, MA. Addison-Wesley 1990.

PAPER NUMBER 1002
DELIVERING CLIENT/SERVER FUNCTIONS TODAY

RIZ SHAKIR
SYSTEM SOFTWARE ASSOCIATES
500 WEST MADISON
CHICAGO, IL 60661
312/258-6000

In the past, the physical topology of most application systems focused computing logic on a single processor to minimize either costs or implementation complexity. Recent technology shifts now make it practical to implement many business functions much closer to the point of action (the client) with an attendant increase in responsiveness and quality.

I. ORIGINS

Within the last decade, a class of distributed-processing architectures has emerged from peer-to-peer origins to become known as client/server computing. The emergence of the intelligent workstation IWS as a cost-effective business platform has been the main catalyst of this technological trend (Figure 1).

An intelligent workstation can be simply defined as a microcomputer that includes a monitor, keyboard, programmable processor, memory and optionally, disk storage and pointing device, such as a pen or mouse. A workstation is *intelligent* when it can run programs that choose how to display or access data. In contrast, we call nonprogrammable terminals (NPTs) *dumb* because we cannot educate them by way of programming. A terminal is merely an input/output device.

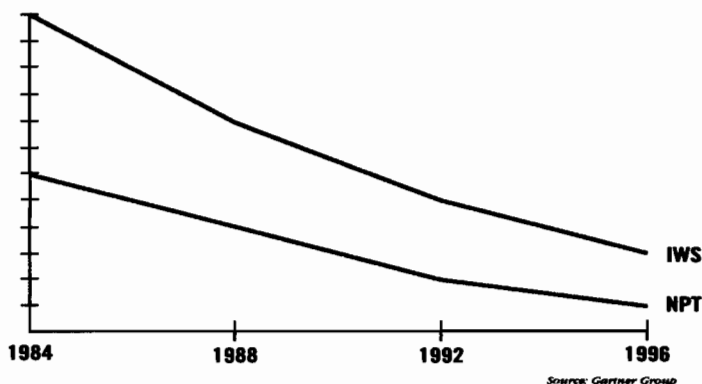


Figure 1: Hypothetical Unit Cost Curves

Meanwhile, business trends brought workstations closer to the source or need for data—offices, warehouses, stores and so on. For example, counting inventory or creating financial statements are business activities often involving data input to or output from a computer by the employee actually performing the task. There is no longer a need for the handoff of forms, calculator tapes, computer reports or other data between the point of action and where in the organization computers are used.

Stand-alone workstations provide inexpensive solutions for point-of-action computing. Shop floor and point-of-sale applications, to name two, are available on workstations. You can implement individual workstations without the traditional communications backbone that has characterized a distributed system. However, the stand-alone workstation does not address the need for consolidating, summarizing, accruing and reporting of corporate-wide enterprise data.

Intelligent workstations initially became popular for new desktop applications because of their ease of use. As the requirements for connectivity to enterprise data increased, the communications infrastructure was formed to bring data from host systems to workstations for ancillary processing. The increased cost of centralized computing, combined with the increasing performance of workstations, prompted *off-loading* of main business processes from traditional business platforms to the workstation (Figure 2). However, a workstation alone cannot provide the security, bulk processing and data reliability required for industrial-strength applications. These remain a key competence of larger systems.

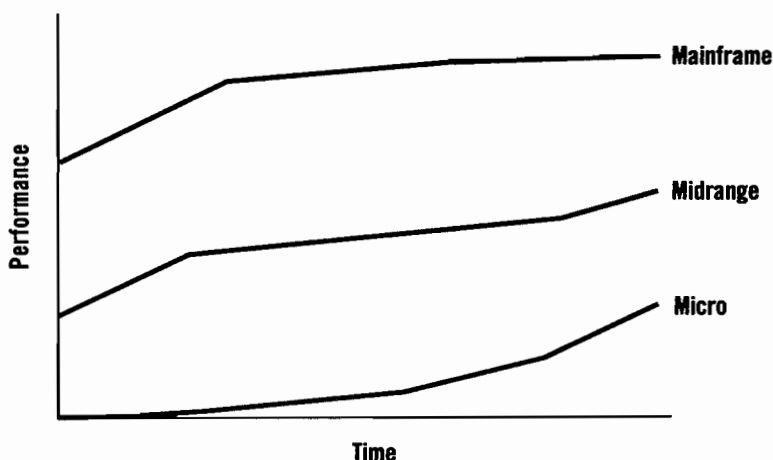


Figure 2: Relative Performance is Narrowing

Point-of-action computing is most commonly located on an intelligent workstation (Figure 3). A client/server architecture does not specifically require implementation of client components on workstation processors and server components on midrange or mainframe processors. The most important aspect of client/server computing is the separation of processing to provide point-of-action applications while protecting the integrity of corporate-wide enterprise data. Therefore, it is possible to implement a client process on a midrange processor and the server process, where most appropriate.

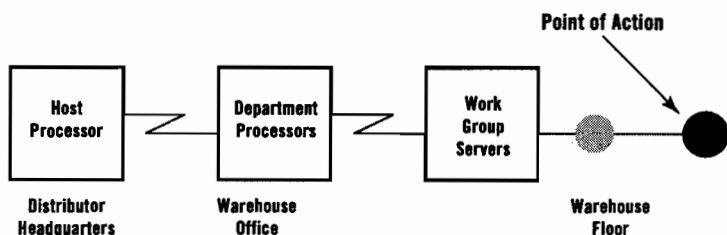


Figure 3: Point-of-Action Computing

To provide a solution appropriate to business, we must exploit the advantages of workstations without risk to production data and expected results. Client/server architectures have evolved to divide database, presentation and application processes among the computing components that bring the most value to each business solution.

II. CLIENT/SERVER ARCHITECTURE

Some traditional programming architectures use a monolithic style, in which a single executable program comprises all processing. The term polyolithic is becoming popular to describe a programming style that separates programs into pieces. For example, properly structuring a polyolithic program into separate modules for user dialog versus application processing really creates two programs. The next step toward client/server is to run the two programs on separate computers: the client program requests and receives services from the server program via passage of data through a communications protocol.

In this simple example (Figure 4), substituting "Program A sends a message to Program B and Program B sends message to Program A" for "Program A calls Program B and Program B returns to Program A" creates the beginnings of a client/server architecture. The *client* program requests some-

thing via message from the *server* program; likewise, the *server* responds to the *client*. We wish it were always so easy.

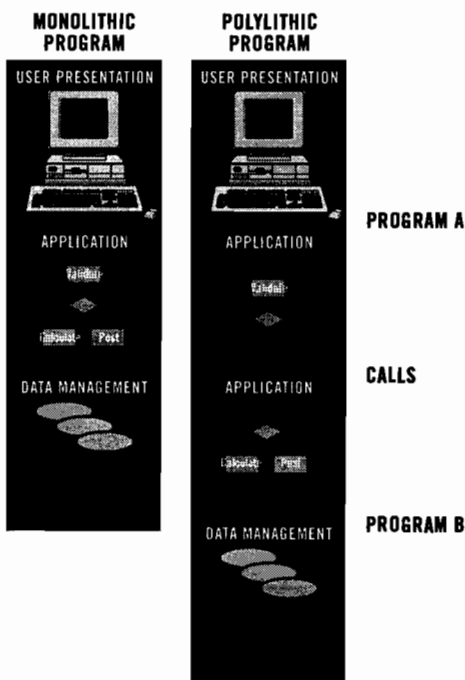


Figure 4: Two Popular Programming Styles

Unfortunately, because we can divide this simple polyolithic model in so many different ways, no one single definition of client/server will satisfy all requirements. Many vendors and products solve this problem by implementing just a single model of client/server. But a monolithic definition of client/server suffers from the same drawback as the definition of monolithic programs: rigidity. Not every business task fits the same client/server model. Therefore, the strategy is to adopt multiple client/server models for your applications because each variation suits one application function better than another.

For example, a user who spends the whole day making queries about customers' payment histories may benefit from having a client program presenting a GUI, with all processing and data residing on the server. On the other hand, creating a budget consists of invoking many functions, not all of which benefit from the same treatment. Last year's actual expenditures must be retrieved, next year's growth factors imputed and so on. The operational database contains the actual figures but need not be accessed again until the new budget is ready to be implemented. One logical implementation

would be to create one server program to retrieve existing enterprise data and another server program to post the new budget. Client programs would perform all other tasks, using data local to the workstation. Among these client programs would be the user's choice of spreadsheet, word processor or graphics software.

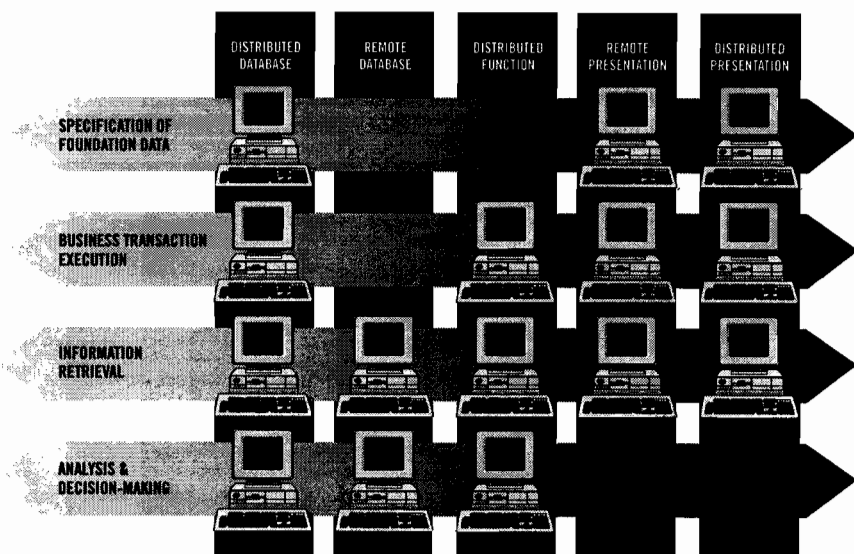
Popular Client/Server Models

Approaches to client/server development can best be discussed within the context of the different models of client/server processing. The Gartner Group has popularized a set of generic models to categorize client/server implementations. For these models, Gartner assumes the user interacts with a graphical display managed by a presentation program (client) on an intelligent workstation. They describe the general manner in which you can divide an application process between two computers. Notwithstanding, actual deployment of client/server products will exploit emerging best of breed technologies, including distributed computing environment (DCE), object linking and embedding (OLE), open database connectivity (ODBC), distributed system object manager (DSOM) and so on.

Figure 5 shows one way how you can categorize its products to exploit the client/server models. Variations of client/server designs will not fall solely within the five Gartner models. The models are generalizations to illustrate the main distinctions of client/server implementations and provide terminology we can share in common. Figure 5 shows some application designs will clearly fit within one model or another. Some will fit between models and yet others will be implemented with some elements of several models. This is because client programs request different degrees of processing from a server program or database manager.

Client/server designs are neither limited to providing graphics, nor to requiring two computers. The following points underscore the difficulty of agreeing on a definition of client/server computing:

- The simplest description of client/server is one process requesting service from another by way of a message.
- Although GUI is popular and ergonomic, users sometimes prefer client processes with character interfaces.
- If you add a third computer (or tier), the middle process is really both a client and a server.



Source: Gartner Group

Figure 5: Client/Server Model

One common characteristic of all client/server definitions is that the program which presents the user interface operates as close to the user as possible. The graphical user interface is the largest contributor to ease of use and, for this reason, the workstation will predominate point-of-action computing. However, for definition purposes, the client/server presentation may be graphical- or character-based but distinguishes itself by the fact that the client program and the server program pass a message instead of sharing memory. Workstations may not always be the point-of-action computer. Nonprogrammable terminals and electronic interfaces, such as Electronic Data Interchange (EDI), will continue to have a significant place in many operations.

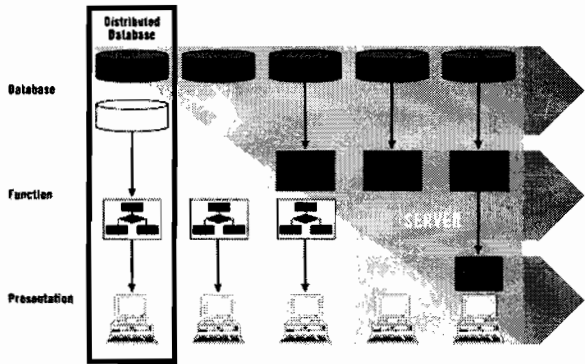
III. BASIC CLIENT/SERVER MODELS

Distributed Database

The Distributed Database model (Figure 6a) contains the most layers and often comes to mind when discussing more than two tiers of computers within a client/server architecture. Distributed database is a client/server model that also has widespread interest in distributed or peer-to-peer networking.

With this model, you can store data that is less critical to timely enterprise processing closer to the point of use. The effectiveness of this model depends largely on the underlying database technology. You should be able to choose on an entity-by-entity basis to implement data locally on the workstation or remotely on the network using popular database management systems.

*Figure 6a:
Distributed
Database*



For instance, as you define your data model you can select SQL/400™ for your AS/400, Informix for your RS/6000 and HP 9000, SQL-Server for your LAN server, or DBASE™-emulation for the drive on your Windows workstation.

For example, you could design an application to allow you to check out data from the server and sever the workstation connection. You can then complete the task at hand and reconnect. The workstation need not be bound to the network connection behind your desk. There are many other ways to apply this model, such as dial-in, wireless and portable computing.

In addition, there are ancillary programs on the workstation, such as spreadsheets, calendars and electronic mail, that enable the business process to be extended. These programs are most easily integrated with databases local to the workstation (via hard disk or LAN server). The local database enables integration with these programs or disconnection of the user from the network to take work off-site.

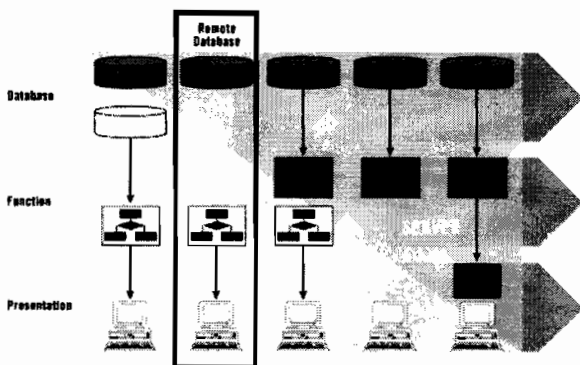
In three-tiered implementations using a PC Local Area Network (LAN), a file server is used for local data. The file server's DBMS is more robust and often tightly integrated with the enterprise server DBMS. The major vendors of DBMS products have stated their direction to provide more work group versions to support the distributed database model for client/server computing.

Remote Database

The Remote Database model (Figure 6b) is recognizable by the presence of all application processing on the client platform. The server provides all database functions. Usually, the client application requests data access directly from the server DBMS via SQL or ODBC drivers.

This model of client/server has been the easiest to integrate with data from other models or existing host-based systems. If the database is defined openly so all data is normalized and accessible directly from the DBMS without reformat, the remote database client can operate against the data regardless

Figure 6b:
Remote Database



of how the original system was architected or what tools and languages were originally employed. Closed database designs do not define each column uniquely to the DBMS and require application analysis to decode each table. Data models should provide full and open definition of normalized data to the DBMS.

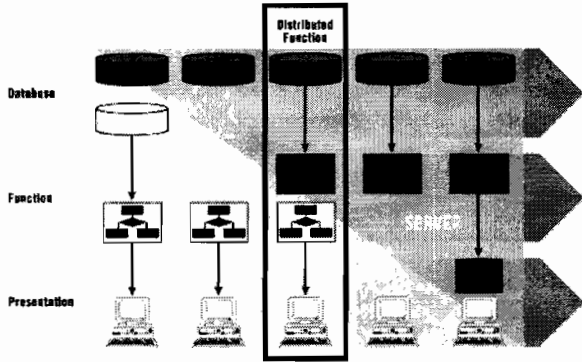
By designing applications to utilize established middleware, emerging technologies and standards will be easy for you to support. This approach also allows you to access data on many more server platforms. Applications could then support direct connection to SQL/400, Informix and other popular relational database management system (RDBMS) servers, such as SQL-Server, Oracle and so on.

Distributed Function

The Distributed Function model (Figure 6c) allows you to divide application processing, such as summarization or batched data entry, within the client program to utilize processing cycles on the client instead of the server. This model allows you to keep update processing close to the data on the server. Validation and dialog processes take advantage of the independent processing power of the client platform to increase response time and satisfy ergonomic requirements.

Often the client and server must operate in completely different technical environments. Windows programs are not written in the same language as OS/400 and Unix programs. No single language spans all environments and meets your requirement to take special advantage of the architecture of the chosen computer.

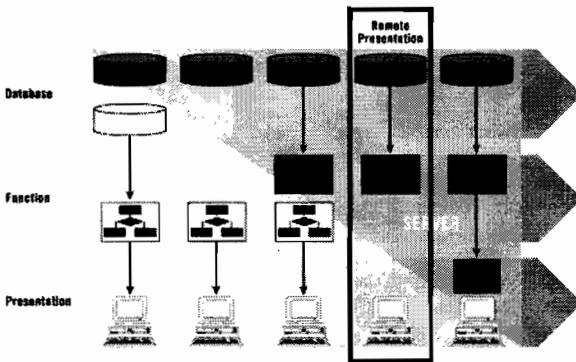
*Figure 6c:
Distributed
Function*



Remote Presentation

Using the polythitic example mentioned earlier, you can remove all dialog with the user to a separate program that can run on a workstation and communicate via messages to a server program on another platform (Figure 6d).

Object orientation is proving the most productive method for managing the complex concepts and facilities of graphical client environments. This provides full reusability of application methods and windows controls. The technicalities of programming windows and network communications are fully encapsulated to provide you with a truly point-and-click development environment.



*Figure 6d:
Remote
Presentation*

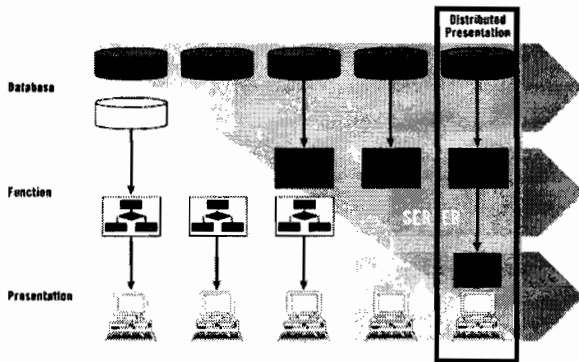
Distributed Presentation

The Distributed Presentation model provides you the choice of text-based interface via host session or graphical interface via client/server (Figure 6e). The same transaction program is available for execution from both nonprogrammable terminals and intelligent workstations. This provides flexibility to support user preference or network topology and maximizes the reuse of code.

This model allows you to rapidly provide the benefits of a GUI to your proven systems. Feature-rich tools allow you not only to provide buttons and mouse control to existing screens under Windows but enables you to implement other benefits of a GUI without changing existing programs:

- Reduce the amount of data on cluttered screens by providing pop-up windows;
- Define a menu bar to use other parts of the system that are related to the current screen without the traditional nesting of a "menuing" system; and
- Create intuitive methods of navigation between applications for improved end-user productivity.

Figure 6e:
Distributed
Presentation



A new graphical interface to transactions provides significant benefits, including the change to the orientation of the product line's interface from action-object to object-action. Action-object is the traditional menu approach: select the action you want to perform (maintain customer) and then specify the object to operate upon (customer 123). Other software vendors have taken a simple approach by merely extending this to GUI and packing as many menu choices as possible across the top of their windows: all with action titles like Maintain Item, Inquire Item and Copy Item.

Furthermore, you can add value to the graphical interface by implementing an object-action interface that allows you to start a business task by selecting the object first. For instance, double-click on one among a list of customers and you are presented with that customer's pertinent data. Select your action via push-button or menu to view a different sort order, inquire in more detail, copy, delete and so on. As a result, your end users will experience significant productivity improvements.

IV. CLIENT/NETWORK

Client/network computing is the next step in the evolution of distributed computing. Even as initial open client/server implementations are delivered, you are anticipating the next evolution of this trend. Network topology will soon replace operating platforms as the focus of technology enablers. Database and communication technologies are moving quickly to solve the challenges of deploying industrial-strength business systems across the network.

Today, the architectural division of labor inherent in client/server puts more of a load upon network resources to provide connection between the many parts of an application, instead of simply carrying screen data to and from terminals, or bulk transactions targeted for overnight processing. In addition, more services are demanded from the network as servers are provided for print services, local data management, network management and interconnectivity, both within your organization and with your trading partners.

By focusing the application architecture on the network instead of a given server platform, your product lines will remain operationally configurable well into the future and not locked to today's technology. Communications, DBMS and middleware products will provide more reliability and data integrity across client/server cooperative and peer-to-peer architectures than today's application programming interfaces. The vision of the user of an application operating on data across complex heterogeneous networks without concern for the location of the data is nearing fruition.

Communications middleware transports the messages from the client dialog to the server process. Directory services and network security are maturing and will allow remote procedures to operate independently of the client dialog (via asynchronous or parallel processing, queuing, and so on). Interconnectivity between DBMS products is evolving to the point where mixing and matching of middleware and systems software will make it unnecessary to focus applications solely on a single-server platform (Figure 7). A multiplatform architecture provides you with more choices of middleware for your network in the future.

Furthermore, system-management products are evolving to enable control and flexible deployment of client and server programs across your network topology. This new dimension to configuring business processes is a stepping stone to the distributed OO frameworks that will be in place later in this decade.

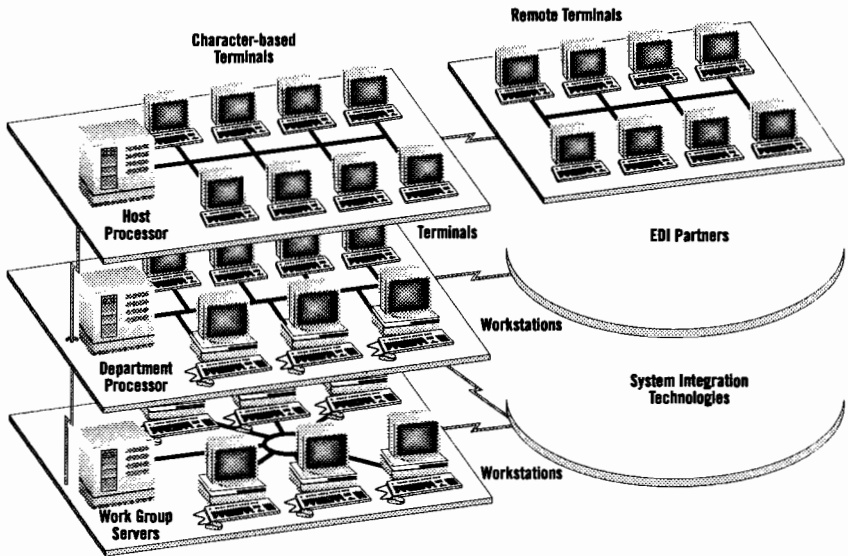


Figure 7: Network as the Server

Two-dimensional discussions of hierarchically tiered client/server architectures, as shown in Figure 7, are being replaced with three-dimensional network topologies. Soon, the fourth dimension of time will be an important factor in the implementation of business solutions that can execute unattended, at regular intervals or based upon business events.

V. CONCLUSION

The swift pace of technological change challenges you, as your company reengineers business and technical processes. Client/server architecture is destined to play a large part in the changes transforming enterprise computing in this decade.

Combined with evolving open systems, heterogeneous networks and OO techniques, client/server complicates your choices. Automation of the development process is key to managing the various aspects of tomorrow's systems.

Paper #1003 - Client/Server RAD: A Case Study

Ben Lloyd
Hewlett-Packard Company
5301 Stevens Creek Blvd.
PO Box 58059, MS 51LSN
Santa Clara, CA 95052-8059
(408) 345-8267

This paper will discuss the techniques involved in producing a Client-Server system using Rapid Application Deployment, and give a case history of using these techniques. We start with a discussion of Rapid Application Deployment, followed by a discussion of client-server architectures, and finally try to tie them together using the case study.

1. RAD: The Latest Thing

Trending towards the millennium, Information Technology consultants and pundits are in search of the golden fleece that will fill their coffers by providing a Great New Idea to sell. In particular, industry mavens such as James Martin, have perceived the need develop new techniques to sell their development methodologies to Generation X, who influence IT directions today, and are the MIS directors of the future. In analyzing the market, the trends are obvious: increasing demand for producing software solutions in a short amount of time and with minimal infrastructure investment. Outsourcing, packaged solutions, customized templates, and focus in the industry on marketing have all made custom development an anathema to business managers in all sizes of company, or at least, that is what their consultants are telling them.

If we are to be honest with ourselves, perhaps the greatest contributing factor to this trend is the perception by our customers of the cult of Information Systems Development: the tales of projects that are years behind schedule and millions over budget, which when delivered are hopelessly out of step with the current (i.e. new) needs of the business. As competitive market pressure increases on our customers, driving them to change how they do business, so must Information Technology change. In very few cases (particularly with large systems) can a development project succeed in addressing all, or even a good majority of the business requirements using a traditional waterfall methodology (even the more recently trendy ones such as Information Engineering (IE)). By the time such a system goes into test, the business requirements have changed, rendering it obsolete, redundant, or just plain unusable.

If we are to keep any development responsibility at all in house, therefore, we must adapt, and adapt quickly.

Enter RAD. Rapid Application Development (or Deployment, or Design, depending on which consultant you're paying to tell your management what you already know) is one approach to this problem. Know by other acronyms, such as ASD (Accelerated System De...), RAD attempts to maximize the speed of application development/deployment, while minimizing the tradeoffs one makes when relaxing the stricture of using a methodology such as IE. RAD succeeds by expanding responsibility for the success of the system to include those who will benefit from its success: the Users.

If it is done right, RAD can produce valuable results in less than half the time of more traditional methods. This may be slightly misleading because RAD should only be applied fully to projects smaller than a certain size, those projects which would traditionally take 12-18 months or less. I believe, however, that RAD can, and should be applied to parts of larger projects to keep them going: I would hazard a guess that, independent of the size of the total project, if you don't deliver some tangible results within nine months, your management is going to start evaluating alternatives.

The basic principle behind RAD is that we eliminate all parts of the analysis/design/development process that are not used *directly* in producing the next required deliverable, or that are in a form that is not immediately understandable by the Users. This means that you can eliminate all or some of the following (the quantity you choose to eliminate should be directly proportional to your willingness to accept, and take, risks):

- Analysis Report
- External Specifications
- Internal Specifications (with the exception of Transaction Definitions, see below)
- Design Report

...

If you work in a company that requires its IT employees to make a pledge to produce any or all of these documents, then you have a significant marketing effort to make, and you may be best off hiring a consultant to convince your management to adopt your ideas. Or, you could find a sympathetic executive to fund a skunkworks.

2. Requirements for using RAD

Regardless of the attitude your company has about the sanctity of the Information Systems Life Cycle, the requirements for using RAD on a project are not trivial. You *must* have *all* of the following in order to use RAD:

1. A strong sponsor from the business (i.e. User) side who is willing to commit the most valuable person on his team at least half-time to the project (with full responsibility to make decisions as to system and process design), and who understands the RAD process.
2. An experienced IT project manager or leader.
3. A small (2-5 person) development team that has worked together for at least 1 year, and has similar experience with the development tools and technology being used.
4. Software tools that allow rapid prototyping, and that facilitate a quick transition from prototype to production system.
5. A work environment in which the development team (cum business rep) does not have to attend numerous meetings, provide application support, or give demos, and can have frequent *ad hoc* discussions of design decisions.
6. A well-defined project charter which clearly outlines the scope of the project and all its deliverables (as well as, within reason, what will not be delivered).

You will note that I do not (as some rapid development methodologies would) specify a CASE tool as a requirement. There are (three) reasons for this: first, I do not believe that CASE tools are at the point where they can easily generate code from a sufficiently high level of abstraction (high enough for your customer to be comfortable understanding it) completely without human intervention (certainly not in the GUI client-server architecture your customers are demanding, or in which you are operating); second, one of the primary reasons to use CASE tools is to validate each step of the analysis and design phases with your customer: if you have your customer on the team with you (and don't give him the silent treatment because he has an MBA and doesn't understand C++ syntax), he will validate it for you; third, another primary reason for using CASE tools is to increase reusability: this, I believe is one of the attributes we may have to sacrifice for the sake of speed, and I have yet to meet a developer that will blindly accept a piece of "reusable" code without validating, and usually adapting it in some way.

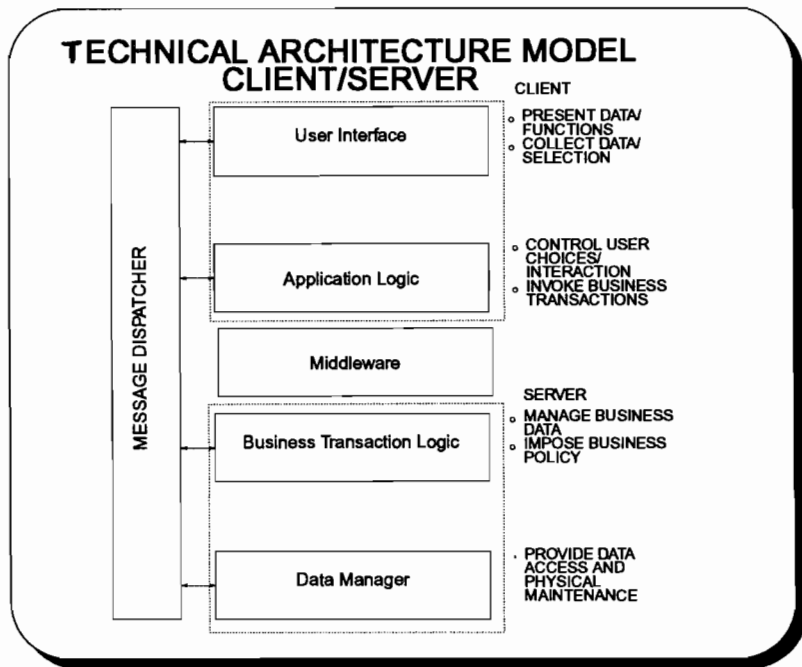
All this is not to say, if you have a CASE tool and are happy with it, that you shouldn't use it. On the contrary, there are distinct advantages to using a CASE tool, and if you can use yours to produce code, then you're better off. Rather, I would say that if you do not currently use a CASE tool, do not go out and buy one for your first RAD project.

I also do not mention Object Oriented development methodology, tools, or languages (or OO, for short). This has to do primarily with a client-server issue that I will discuss below, and with the fact that, in my experience, blindly adopting C++ will not help you: if you have used C, your first C++ will be far from OO, and will just confuse the matter, and it is more difficult to call C++ from C (in the HP world). Again, if you are currently using an OO tool, then by all means continue to do so, but don't try to cut your teeth on one when you have an end product due in six months.

3. What about Client/Server?

To this point, I have not discussed client/server at all. This is primarily because, in many ways, applying RAD to client/server applications is little different from applying RAD to traditional monolithic applications. There are some critical differences, however, that affect the choices of project deliverables and the organization of the development team. I will address the latter now (separately), and incorporate a discussion of the former in the following list of steps in RAD projects.

HP has defined (for its internal use) a four-layer client/server model (several other papers undoubtedly discuss this):



The pieces are, from top to bottom:

1. The User Interface, or User Presentation Manager

The User Interface, or User Presentation: this is the GUI, block-mode, prompt-and-accept interface to the application with which the end user interacts. You will probably have to justify the use of a non-GUI interface, so my discussion will assume this. I would recommend that for this piece, you use as high-level a tool as possible: since you will be developing using a rapid prototyping method, you will need to be able to respond very quickly to requests to change the appearance of a screen/window, as well as its response to actions (button pushing, menu-pulling, etc.). Further, if you are developing for multiple client platforms (as I was), it is critical that you not develop in native mode, only to have to port later. There are several decent cross-platform development tools that hide the implementation details from you (we used a product from Germany called Dialog Manager, but there are plenty of others available, e.g. Galaxy, XVT, UniFace, SmallTalk [if you really want to do OO], to name a few), and unless you are doing some really special things in MSWindows, XWindows, MacWindows or Presentation Manager, you should avoid coding with their intrinsics for an IT project.

[A small aside here. Many people will take exception to that last statement, and they deserve more than a blithe trashing of their beliefs. I am targeting this paper at those who develop transaction processing systems, in which the user fills in information in a form, maybe looks some things up on a database, makes some calculations, and submits a transaction. In short, the focus here is on client/server transaction processing, in which the client generates a transaction that the server(s) process and return a result. Client presentation features such as graphics, multimedia, etc. are beyond the scope of this discussion, and most probably will have to be developed in some "native" constructs. I believe that those are the exception, rather than the rule, and therefore the primary application should be developed in some GUI 4GL, and native calls in some language like C only used where necessary.]

2. The Application Logic, or User Transaction Manager

The second layer of the client/server model is the User Transaction Logic layer. This is the part of the application that, to be succinct, allows the developer to customize the order and fashion in which the User has access to the underlying Business Transactions. Further, it provides any other "local" transactions for the client (e.g. maintaining a local log of transactions, retrieving and saving local files, displaying date and time, etc.). For our purposes, the User Transaction Logic layer formats Business Transactions, passes them to the server(s), and retrieves results.

It is at this layer (and the one below) that I do not believe OO technology should be used (certainly not exclusively). The reason for this is that there is no standard interface to Objects and their methods from outside the language in which they were developed. If an application from a higher level wishes to access the services at this level and the ones below it, it should be easy for that application to do so, regardless of the technology it uses. For example, if the client is written in COBOL (though I say it softly), a collection of C++ or Smalltalk objects are, basically, inaccessible, without a front-end API (probably written in C). And as long as you're writing the API in C (for example), you might as well save code (and a lot of trouble), and write the whole level in that language.

2.5 Middleware

Between the second and third layers is the client/server interface (or middleware, to some). There are several choices available that will run on several platforms, but at this point, I think the safest (both from history and longevity) are: Sockets (Windows Sockets on the PC, Berkeley sockets elsewhere), NCS (Apollo's Network Computing System), and DCE (Distributed Computing Environment, which specifies a Remote Procedure Call [RPC] facility based on NCS). I would recommend DCE RPC for the following reasons:

1. It is emerging as a standard.
2. It is relatively inexpensive for servers.
3. It is essentially free for MS Windows clients (see note).
4. It shields you from the underlying TCP/IP stack (i.e. it will use UDP sockets, stream sockets, Windows Sockets, WSOCKETS sockets, etc. on Novell, Lan Manager, TCP/IP, Internet, etc. without your application having to change).

NOTE: On the PC, there are few options for DCE RPC, some expensive, at least one very inexpensive: Gradient Technologies and DEC (both in the Boston area) have developed or are developing full implementations of DCE for the PC running MS Windows. Each has a per-seat charge that is not insignificant. If you are developing for a small number of users, then one of these might be the best for you because they have implemented most or all of the DCE-defined services. However, if you are implementing widely, the Microsoft RPC product, which comes with the Windows-32 SDK is interoperable with DCE servers, and has only a few slightly annoying, but quite manageable limitations (no full pointers or file transfer methods). It does not have the time and security features specified in DCE (which the other products mentioned do), but if you don't have those now, you probably won't miss them.

3. The Business Transaction Manager

The third client/server layer is the Business Transaction Manager, which encapsulates all the business rules surrounding each transaction (access, concurrency, roll-back, validity, etc.). This layer knows the physical structure of the underlying database(s) and generates database transactions (using SQL or IMAGE, for example) to the appropriate databases. Further, it knows if databases are distributed, and how to generate distributed transactions to those databases, though it may not know whether a single database is implemented as a distributed database. I would say that parts of this layer could be implemented in OO technology, particularly for reusability, and to bullet-proof your database.

4. The Data Manager

The fourth layer is the Data Manager. This is, in essence, the database. It is the interface between the program and the physical storage of data on disk (or tape, etc.). It has a defined interface for communication with the Business Transaction Manager (usually SQL).

5. The Fifth wheel

In the diagram, there is a fifth component, the Environment Manager, which theoretically allows each layer to communicate with the other layers, but I have yet to see a reasonable implementation of this concept, so I will focus strictly on the communication between the second and third layers (the client/server link), assuming that all other communication is done through procedure calls (or object method invocation).

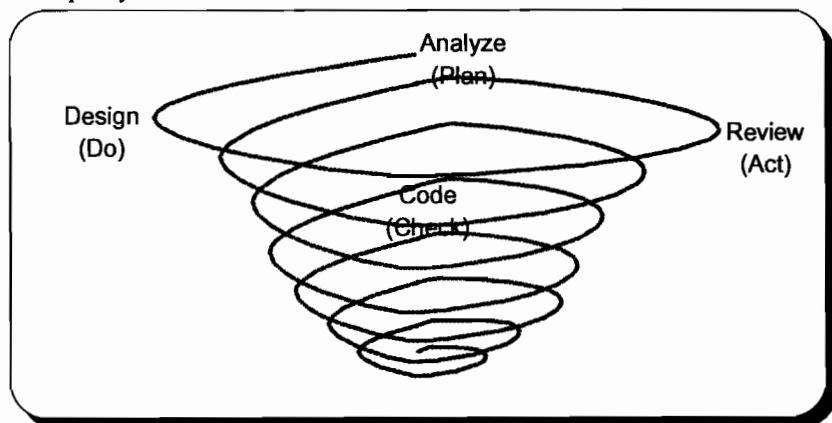
Given this model, then, the most important and most sensitive part of implementing a client/server application is the definition of the Business Transactions. The goal of client/server (as a technology or approach) is to define a set of Business Transactions that may be called by multiple clients. Therefore, if anything should be done extra carefully in your development cycle, it is this definition. In fact, it is the one part of the client/server approach that does not lend itself extremely well to RAD.

4. How to do/be RAD

The key to RAD is having Users involved at every step, providing input, answers to questions, policy decisions, evaluation, and moral support. Participating in a RAD project can be exciting, but is also likely to be rather draining (read "Soul of a New Machine" if you don't believe me). Do not count on executing two RAD projects concurrently or consecutively with the same team.

Each methodology (Martin's IE, TI's IE, Ernst & Young's Navigator, etc.) has a slightly different set of steps, and names for those steps, but they all can be condensed to a basic sequence of events that must occur during the development

process. One can visualize the process as a helix much like a funnel cloud. I call this a quality vortex:



With each successive pass at design or coding, it gets closer to the end product, and closer to what the users want. Basic steps could include some or all of the following:

1. Requirements Gathering.

In this step, the general functionality of the application is defined, including classes of business transactions (e.g. order submittal, inventory reconciliation, credit check, account deposit, etc.), data (e.g. order, inventory, customer, account, etc.), and external agents (e.g. customer, purchasing system, shipping system, account rep, etc.).

Deliverables: Context Diagram; 2/3-level Functional Decomposition; Conceptual Data Model;

Techniques: Facilitated joint sessions with users and sponsor

2. Elementary Process and Triggering Event Definition

This is a set of elementary processes and their definition. An elementary process is one that has a single triggering action, and can complete without any subsequent intervention from external agents (including time lapse). A third deliverable, the state-transition diagram defines the life-cycle of each business area element (data model subject area/conceptual entity). All three deliverables should be cross-referenced to ensure completion/coverage (each business area requires a state-transition diagram, and must be referenced in elementary processes which create/report/update and delete the business area object).

Deliverables: Elementary Process List/Definitions; Business Area State Transition Diagram; List of triggers; Coverage matrix;
Techniques: Project Team discussion; individual assignments with business representative to gather data.

3. Business Transaction Definition

As noted above, this is the most crucial step of the process. It should produce a list of transactions, and for each transaction, the list and description of each data element to be passed to the server, and each returned.

Deliverables: Transaction List; Transaction Definition List; Transaction Behavior Description; Transaction data element list.
Techniques: Joint Sessions with actual end-users to develop the lists; Cross-referencing with data models.

4. Logical Data Model

This should be completed either immediately before or after the transaction definition step. The output of both steps should be cross-referenced with each other for completion.

Deliverables: Entity Relationship Diagram; Entity Definitions; Entity Attribute Lists;
Techniques: Joint Sessions with end-users to develop initial list; normalization may be performed by project team members;

5. Screen/Window Definition/Map

The users should be able to define the general ways in which they would like to navigate the system. The window definition should include the general and specific types of data fields that should appear in each window/screen, and the anticipated navigation from that screen. To do this, one starts in the beginning posing the questions, "When you power the system on, what do you want to see, what do you want to be able to do, and where do you want to be able to go next?" The answers to these questions will help you design the initial prototype.

Deliverables: Screen/Window definitions and layout; Screen Navigation Map.
Techniques: Joint sessions with end users.

6. Develop initial client prototype.

Having obtained an initial idea of design parameters from the previous step, the development team should now create an initial prototype (with dummied functionality [aka no client/serve transactions]). Use whatever tool suits you best, but it should be fast: the initial client prototype should take no longer than 3-4 weeks to create.

Deliverables: Initial (dummy functioning) client prototype;

Techniques: Any way you can, recognizing that you may not have time to throw anything away in the future stages of development.

7. Iterative client prototypes

Obtain feedback on the prototype as it moves through its maturation process. It is best to have your user representative provide feedback on each new feature so that little time is wasted on the wrong track. Gradually, during this phase, you should introduce actual working transactions to ensure that the approach you are taking is valid, that the performance is acceptable, and that the transaction formatting will allow the client to function as required.

Deliverables: progressively better versions of the client.

8. Server and Transaction Prototypes

Before the initial client prototype is developed (or even defined), the server transactions can begin to be prototyped. If you are using an RPC product for your middleware, each transaction can be roughed out with no actual server functionality, but with the true interface defined (using some sort of Interface Definition Language [IDL]). The actual server team can start development (and should start as soon as possible, since it will undoubtedly take longer than the client development).

Deliverables: progressively better versions of the server.

9. Peripheral support software

No system is an island. Every system we build today will require interfaces with existing or planned systems. If possible this communication should be performed through standardized interfaces, moving data back and forth either directly or through data hubs or warehouses. The specification of external interfaces can take a great deal of time, and I would devote one team member to this "data migration" task, coordinating with other systems providers, identifying data sources, and developing interfaces. For this task, the business user can be particularly helpful in convincing other organizations to allow you

to use their data. It is essential that these other organizations understand exactly what you will be doing with their data (if you misuse, mis-represent, or corrupt their data, they will come after you).

9. Fully functional client/server application

In this iterative prototyping scheme, the final prototype is, by definition, the end system and the specification for the end system. The users sign off on the prototype, and, in effect, sign off on the specification. If you are using a throw-away prototype, this is the point at which you create a production version. If, on the other hand (and, in my view, preferably), you are prototyping with a production-quality tool, you need simply to tie the loose ends (not to oversimplify the process), and perform code optimization, walk-throughs, and prepare for system test.

10. System Test

The system test should be constructed in at least three parts, the first designed to test the client-server interaction, and the others to test the supporting batch jobs on the server (data loading, interfaces, exports, etc.) and the installation. Each test should be designed so that it can be completely iterated in less than 1 day. This will allow quick regression tests after fixing defects discovered during the previous test cycle.

11. Pilot/Alpha Test

It is critical to conduct a pilot or alpha test. In my mind, these two are quite different, serve different purposes, and should be used in different types of project. I see the pilot test as, in essence, an expanded prototype review, with a selected set of users who will provide input as to the functionality of the system and the prototype. This type of test is particularly appropriate in the case where your end-user base will be geographically and/or functionally distributed. During the prototype iteration, you will have obtained feedback (most likely) from a small subset of those distributed users. Opening the prototype up to a wider audience will help to ensure that you do not end up with a product that is useful only to a small percentage of your targeted users.

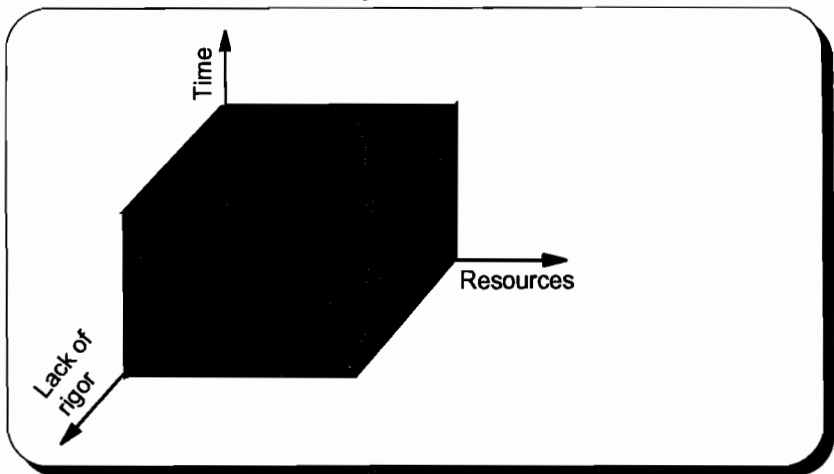
The Alpha test, on the other hand, is more of a structured, controlled test, in which the testers are using the system in production with a set of acceptance criteria on which they are evaluating the product. The end result of an Alpha test is a checklist and a user-signoff saying that the product works *as documented*. The Alpha test is important in those projects where you have a clear understanding of the desired product (which is easier if you have a geographically and/or functionally

homogeneous set of users), and are more interested in having several different cross-checks to validate system completeness and accuracy.

If you have a built good regression test, the Pilot test will provide you with valuable input into system operation, usability, and functionality. Because of the potential for continued iteration, however, it will be longer than an Alpha (and in some cases, may actually be followed by a true Alpha). I would recommend that for the first release of a product, a pilot test be conducted, while in subsequent releases (with generally fewer drastic changes), an Alpha may be more appropriate.

It is important to re-emphasize the RAD tenet: in order to gain speed in the development process, we must sacrifice some of the rigor in more intensive approaches (such as IE). In the strictest sense, this would imply a loss of quality in the final product. I follow the belief that if the people on the project are concerned with, and focused on quality, and can code well (making few stupid mistakes in coding), this loss of rigor need not result in a lower quality of output. This is where it is crucial to have an end-user representative on the team. Every single "if" statement that an engineer writes is a business decision; many of the techniques used in approaches such as IE are intended as communications tools, allowing the end-user representative to validate requirements and design. If the end-user is on the team, the engineer need simply ask him/her the question, and incorporate the answer in the design/code. The end-user then validates the end-result of the prototype, and so validates the design.

It may be helpful to view the development process as a set of interdependent demands or attributes. In other words, when we graph the attributes of the process we create a multi-dimensional object with finite volume. In three dimensions, it would look something like:



To use an analogy: If we fill this object with a liquid (which has a minimum, relatively constant volume), you can see that decreasing one of the dimensions has the effect of increasing another. In other words, if you try to reduce Time to Market, you must either add more resources or sacrifice some rigor (and potentially some quality as well). Note also that you can only add so many resources or eliminate so many rigorous steps to reduce time. At some point, there is a minimum time below which (without significant advancements in development technology) you can not go.

5. How does it work in practice? A case study of LitStation

In 1991, HP took a survey of its sales office branch managers, and discovered that one of their biggest dissatisfiers was managing and using product literature. At the time, each sales office maintained a literature room, stocking the product literature used at that office. If a sales rep needed a data sheet, he/she wandered down to the lit room and described the product and the data sheet to the literature coordinator (and, generally gave him/her a name and address to whom to send it along with one of his/her business cards). If the literature was out of stock, the coordinator had to order it from the central warehouse, which could take a couple of weeks (in the US). The cost of maintaining that literature stock, along with the waste caused by scrapping obsolete literature was a major problem.

In 1992, our IT department, which develops applications to support the Field, decided to tackle this problem, and to prove several development concepts as well. These concepts became the nucleus of the technical goals of the system:

1. Validate HP's client-server model (see above) in real-life use;
2. Employ the use of the RAD process from start to finish.
3. Develop a GUI client with an easy-to-use development/prototyping tool.
4. Develop a client-server API to allow other applications easily to call our server.
5. Complete the development in less than 1 year with 2 engineers.

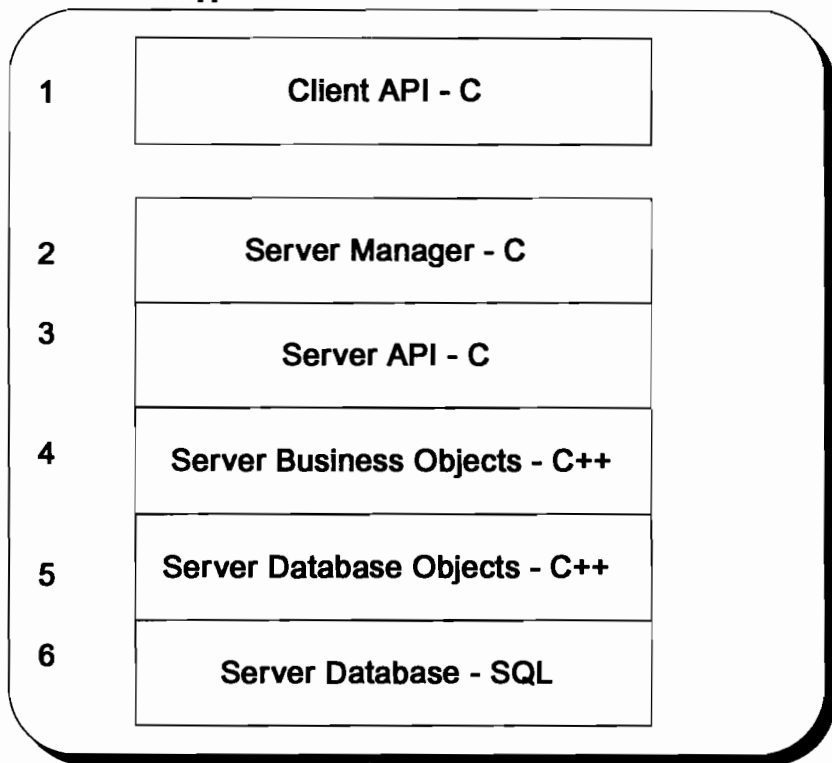
At the beginning of the project, neither engineer had produced a true client-server application, neither had used C to any extent, developed a GUI, or seen a socket. We were starting from Ground Zero.

In the winter of 1992, we held a Joint Requirements Planning (JRP) session to determine basic requirements for the application, followed by a JAD session to identify specific design attributes (we produced a context diagram, a logical data model, and a process decomposition diagram). This allowed us to begin to create our first prototype.

With cost a major consideration in this project, we evaluated several GUI development tools, and initially selected Borland C++/ObjectViews for developing the prototype (at the time, we saw the PC implementation as a stop-gap measure, and did not see any future for a Motif client). At the time, a PC implementation of NCS was outrageously expensive, so we elected to go with Sockets for the client/server interface. We also selected HP Allbase for our data manager.

We thought that the ultimate future of client-server computing would be in distributed objects, but did not think the technology was ready for us. Therefore, we chose to layer the server to allow maximum future flexibility in migrating to new communications methods:

6. API/Server Application Levels



In this diagram, the initial client would have a C API (level 1), with one call per business transaction. Each call would perform 5 functions:

1. From the input parameters, create a transaction buffer to send to the server.

2. Send the transaction buffer to the server (using a C++ encapsulation of Sockets).
3. Retrieve the resulting transaction buffer from the server.
4. Disassemble the resulting buffer into the output parms.
5. Log the transaction and results locally on the client.

Level 2 (the Server Manager) essentially reverses the process from level 1:

1. Receive a formatted transaction buffer, and identify the transaction type.
2. Disassemble the transaction buffer into its components, which server as input parms to:
3. Make the appropriate server API call, which exactly mirrors the client API in the procedures it provides.
4. Assemble the output parameters from the server API call into the results buffer.
5. Send the results buffer back to the client.

Level 3 (the Server API) defines the business transactions and how they are to be performed. In general, each API procedure instantiates the C++ objects from level 4, and invokes the object methods to perform the business transaction, returning their results to Level 2.

Level 4 (the Server Business Object level) consists of a set of object classes centered around the primary data classes of the data model (e.g. Literature Order, Fax Order, Literature Piece, etc.). These objects contain the actual business logic to transform an object from its initial (null) state through the various states it can take, until the final transaction state is reached. In addition, the business objects contain references to the Database Objects (level 5), and manipulate them to update/query the database when needed.

Level 5 (the Server Database Object level) consists of a set of object classes which correspond 1:1 to tables in the database. Each of these table classes has 8 standard methods:

1. `value_blnk_upd(struct *)` takes a structure which mirrors the data in the database table, and updates the private data elements of the table object. This and the next method are how the object's data is instantiated. This method updates every single data element, regardless of content.
2. `value_upd(struct *)` acts like `value_blnk_upd`, but only updates those data elements whose corresponding field in the input structure is non-blank or non-zero. This allows selective updating of data elements in the object.
3. `get_values(struct *)` returns to the structure the values of all the corresponding data elements in the object instantiation. This is how the objects data is retrieved.

4. `row_insert()`. Having instantiated the object's data elements with `value_blk_upd` or `value_upd`, `row_insert` will insert a row with all these data values into the appropriate database table.
5. `row_select()`. Having instantiated at least the key values for the table, this method will select the qualifying row from the database, loading the data into the object's data elements.
6. `row_update()`. Based on the key value, update the database table with the values in the object's data elements (using null indicators where appropriate).
7. `row_delete()`. Based on the key value, delete the corresponding row from the database.
8. `row_validate()`. Run all appropriate validation on the values of the object's data elements prior to updating the database.

Finally, Level 6 (the Server Database) is an Allbase-SQL database, accessed with embedded SQL from the database objects.

This approach may seem rather complicated, but we hoped (and indeed it has proved to be true) that it would allow us future flexibility to adopt new technologies:

- To convert to NCS or DCE/RPC, we would simply removed level 2 (or levels 1 and 2), and build a server around the server API (in fact we have done this in our third product release, migrating to DCE/RPC, and eliminating level 2 from this diagram; more later).
- To convert to distributed objects, we could simply remove levels 1, 2, and 3, and migrate the business objects to distributed business objects which could be manipulated directly from the client.
- To convert to an Object-Oriented Database, we could eliminate level 5 and replace level 6 with an OODB, allowing us to store business objects directly in the database. At least conceptually, then, we could eliminate all but levels 5 and 6 if the technology panned out.

Theoretically, we could have 6 engineers each working on one of these levels with minimal interference, but realistically, given the scope and timing of our development project, we felt that they should be tackled by no more than two engineers. Because GUI design and development skills are quite different than those required for developing servers, we chose to split up the responsibilities for development between client and server (though with the level 1 API being developed by the server engineer). This allowed the server engineer to code furiously once the requirements for the business transactions had been developed, while the client developer could iterate prototypes to get just the right presentation for the end users.

7. **Socket connections**

Our first implementation of the client-server interface used stream sockets, a connection-oriented protocol in which the client opens a socket to a specific service (maintained in `/etc/services`) on a specific machine (by IP address). We implemented the server using `inetd`, which meant that whenever `inetd` on the server received a client connection request, it would launch our server process and pass the socket connection over to that process. In an `inetd` connection, the server sees the socket as file # 0.

Note: On HP-UX systems, the write side of the `inetd` socket (that is, when the server writes data to the socket going to the client) may be the same as `stderr`. Therefore, any operations that write to `stderr` will write to the socket, a behavior that you probably don't want to have.

This gives us a 1:1 connection between client and server, with 1 server process serving each client. We later found this to be somewhat wasteful of resources, and have created a new architecture based on DCE/RPC more effectively to use server resources (see below).

Because a client user could walk away from the client without logging off the server, we had to implement a time-out feature on the server. This meant that if the server did not receive any transactions for that time-out period (e.g. 20 minutes), it would terminate. The next time a client requested a transaction, it would recognize that the server was no longer alive, and open a new socket to a new server.

Similarly on the client we determined that if a server process died for any reason, the client should not be hung. This caused us to implement a transaction time-out on the client side as well, where the client only waits 25-30 seconds for a response from the server before deciding the server is dead and attempting to reconnect.

Both of these time-outs were implemented using the `select()` facility, which allows us to determine whether there is data queued in the socket.

8. The jump to cross-platform development tools

We thus began server development in the late summer of 1992, but faced a new challenge on the client side: the project sponsors now wanted to be able to run the client on Motif-based workstations as well as Windows-based PCs. At the time, there were extremely limited options for running Windows applications under Motif, none of which would provide the performance that was needed. This forced us to change course on the client (and cost us approximately 2 months in the development cycle).

We had to cut off development to evaluate the several cross-platform GUI development tools available at the time. Through a series of events, we chose a product called Dialog Manager from a German company. In Europe, HP oem's Dialog Manager for packaging with some of its computers.

Dialog Manager is a GUI scripting language (4GL-like), and an API to communicate with C. It works in the following manner:

- An AppMain procedure in C is the first thing called by Main(), essentially passing control of the application to the Dialog Manager script.
- The Dialog Manager script can specify C callbacks for specific actions, which can thus return control to the C application for specific actions.
- The C application has access to API calls which can modify/instantiate Dialog Manager objects. This allows loading of list boxes from C, etc.

The scripts can either be developed using a WYSIWYG GUI editor or by entering the script directly in a text file. Either method produces a source script file which can be compiled to reduce its size and increase the speed at which Dialog Manager loads it when running the application.

In our experience, we found that after a few weeks of experimentation, we could code the GUI more quickly using the scripting language directly than using the WYSIWYG editor.

Thus, in the summer of 1992, we re-wrote the client GUI using Dialog Manager, and migrated from Borland C++ to Windows C/C++ (Dialog Manager appeared to work better with the latter).

9. An end-user JAD session to evaluate the prototype

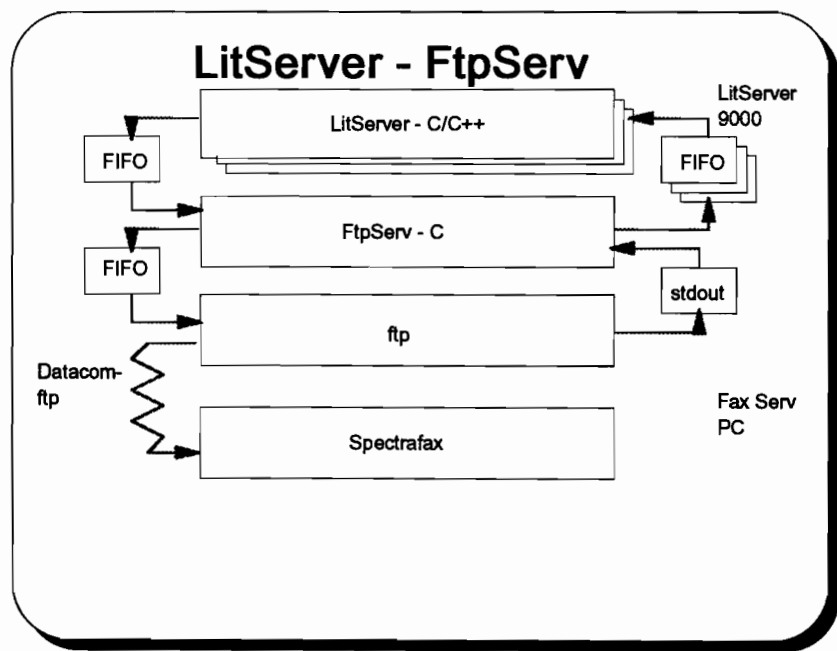
In August of 1992 we held a second JAD session to run the new Dialog Manager prototype by a set of users. We had to do this because our end-user representative was not intimately familiar with the way that the target audience would actually use the system. We had hoped merely to validate our approach to that point, but ended up redesigning a significant portion of the client user interface. Since we had migrated the application to Motif, we would be able to use SharedX (a part of HP Empower) to validate with the JAD participants the changes we made as a result of their input. SharedX allowed us to run our client on a workstation in Santa Clara, CA, and share the display with several sites around the US. This tool allowed us to get real-time feedback on each change we would make to the client.

10. Adding to the scope

To this point, we had been focusing on automating the process for finding and ordering hard-copy literature. In September, 1993, our scope was expanded to include faxing. We spent time developing an interface with the HPFirst system, run out of Boise, to allow us to fax a large subset of the available product literature directly to a customer. This type of interface was new to them, and we spent a good deal of time perfecting a design for the interface.

We decided to use ftp to send fax requests to HPFirst. We first experimented with sending the request directly from the client, but found the use of an MSDOS-based ftp program to be somewhat disconcerting, and a little too slow (we faced the overhead of launching the MSDOS window, running ftp, connecting to the server, transferring files, exiting ftp, and evaluating the results of the file transfer). After a 3 weeks developing this approach, we decided to switch the responsibility to the LitStation Server. In this way, the transaction would be sent to the server like any other transaction, and the server would bear the cost of managing the ftp connection.

For a variety of reasons, we decided not to attempt individual ftp connections from each client's server, but instead to maintain a single "ftp server" that would manage the connection, provide queuing, analyze command results, etc. The FtpServer has the following architecture:



As you can see, this architecture allows us easily to change to another connection method (e.g. UUCP , email, or some other modem-based connection) without substantially affecting the server: we merely change the use of ftp to using another method. The queuing and success reporting remain exactly the same.

To implement this architecture, however, added 4 weeks to the schedule.

11. Memory problems, and how to address them

During the autumn of 1992, we had been having significant memory problems with our application (causing segmentation violations and bus errors on the workstation, and General Protection Faults in Windows). We discovered this only when we migrated the application back to the PC (we had been using the development tools such as SoftBench on the workstation to improve our productivity). It turns out that the HP workstation is extremely forgiving when it comes to memory errors, and our application would continue to run for a long time after the actual memory corruption. The PC is not forgiving at all. We were thus faced with the problem that all C programmers face in managing their heap: how to discover where the memory errors were occurring, and how to prevent them from happening in the future. The first problem could be solved by several commercial applications that evaluate heap memory use to identify potential and actual problems with corruption (e.g. Purify). On a small budget, however, we chose to do it ourselves (in the long run, I don't think that was a good decision).

We implemented a scheme to track heap allocation and cleanup, and to determine where a memory corruption was occurring when we had one. The scheme is relatively simple. I will describe the PC version because there are built-in facilities to support it:

- Replace each call to malloc or realloc or calloc with a macro that writes the source file, line number, and allocation size to a log file and calls the real malloc,realloc,calloc.
- Before the real malloc,realloc,calloc call, call HeapCheck to determine whether the heap is corrupted, and write the result to the log file.

This allows us, when a fault is discovered, to determine the last time the heap was OK, and the first time it was corrupted. In most cases, these were not very far apart in the code, allowing us to scrutinize the code for problems. In some cases, where this was not accurate enough, we would call HeapCheck directly from the code to isolate more closely the problem code areas.

The workstation has no direct HeapCheck function, so we implemented a Heap Management object that would manage heap allocation and amounts by allocating 2 bytes more than the requested amount, and putting 2 signature bytes at the end of the allocated space. Realloc was adjusted to move the signature bytes

accordingly. The Heap Management object then kept a hash table of the allocated spaces and their lengths. To check the heap, we simply walk through the hash table and evaluate each allocated space for their signature bytes. If the bytes weren't there, we would know exactly which memory space was corrupted, and could report it.

These routines saved us a great deal of headache, but much of that could as well or better be performed by the products mentioned above.

12. The byte-order and structure packing problem

Very early on in our development of the PC client, we discovered the two gotchas of multi-platform communication at the sockets level:

1. Byte order for numeric values is different between the PC and HP-UX.
2. Structures pack differently on the PC and HP-UX.

Problem #1 is well-documented in the sockets manual (we were using HP's WSOckets implementation of Berkeley sockets), and using `htons`, `htonl`, `ntohs`, and `ntohl` to convert all integers solved that problem, for the most part. The confusing thing here is that some of the sockets routines called on the PC return integers that are *already* in network order. This made our sockets implementation all the more difficult.

Problem #2 is a real bear: while it is possible to define individual elementary data types (char, integer, long integer, short integer, etc.) to be the same between two platforms, the rules for how these elements are packed in a structure differ. Take the following structure, for example:

```
typedef TestStructX {
    char    xyz[5];
    long int i;
    short int j;
} TestStruct;
```

On one platform, where long integers must always start on a 4-byte boundary, the above structure would have 3 unused bytes between `xyz` and `i`. On another platform, where long integers must only start on a 2-byte boundary, the structure would only have 1 unused byte between `xyz` and `i`.

There are two solutions to this problem, both of which may be used in conjunction, if desired:

1. In a structure, put all 4-byte integers at the top, followed by all 2-byte integers, followed by 1-byte elements (unsigned chars, chars, BYTES,

etc.). If one of the platform has a 4-byte restriction on starting new integers (even if they are only 2-bytes), you can put padding elements between them:

```
typedef TestStructX {
    long int    i;
    short int   j;
    char        xyz[5];
} TestStruct;
```

or

```
typedef TestStructX {
    char        xyz[5];
    char        pad1[3];
    long int    i;
    short int   j;
    char        pad2[2]; /* generally not required */
} TestStruct;
```

2. Instead of blindly copying the structure to the communication buffer:

```
memcpy(comm_buf, &TestStruct,
        sizeof(TestStruct));
```

maintain a pointer to the comm_buf, and walk through the structure, copying each element:

```
char *buf_ptr = comm_buf;
memcpy(buf_ptr, &TestStruct.xyz,
        sizeof(TestStruct.xyz));
buf_ptr += sizeof(TestStruct.xyz);
memcpy(buf_ptr, &TestStruct.i,
        sizeof(TestStruct.i));
buf_ptr += sizeof(TestStruct.i);
memcpy(buf_ptr, &TestStruct.j,
        sizeof(TestStruct.j));
buf_ptr += sizeof(TestStruct.j);
int buf_len = buf_ptr - comm_buf;
```

The first approach is definitely easier to implement, but there may be times when you may wish to use the second.

By far the best approach, however, would be to use a product such as DCE (or Microsoft) RPC. This will perform all the packing, unpacking and byte order work for you.

13. Distribution Strategy

For our first release (May, 1993), we chose two release strategies (one for the PC and one for the workstation):

For the PC, we would distribute floppies with a Windows-style "SETUP" installation. Updates would be performed by transferring (using ftp) a PKZIP file containing all the necessary files to update the client, and a well-known BAT job to perform the update.

For the workstation, we would ask UNIX administrators to install over the network (we use an HP-internal package called ninstall, but /etc/update with a netdist server would work just as well). This method can also be used to install updates to the server from the development machine.

As the popularity of the PC client increased, we decided to implement a network installation/distribution. Our second release (December, 1993) was distributed using this method. We used another HP-internal tool called RS-WIN, which has the following functionality (easily emulated for a single application installation): 1: create a network share (the LanManager term for a networked disk) connecting to the server with the installation package on it; 2: run the installation from the network share. The advantage of this approach was that we did not have to modify the installation software at all: it runs the same over the network as it does from a floppy.

At the time this paper was written (June, 1994), LitStation had approximately 2200 users around North America and Europe. In September, 1994, we plan another release that should increase the user population to approximately 5000 world-wide. There is no way we could support this user base with a non-client-server application.

14. Cross-platform C

Very early on in the initial development cycle we decided that if we were going to support multiple platforms, the client code (and client API code) would have to be completely portable between platforms. We were successful at this by using a very few rules:

1. Code in ANSI C wherever possible.
2. Where non-ANSI or platform-specific calls were required, enclose the calls in `#ifdef .. #else .. #endif` to provide the same functionality.
3. Where certain features were provided on one platform and not the other, evaluate their usefulness, and if appropriate and easy, code on the other platform (we did this with the codelibs stringx routines, which are part of the HP C++ library, but are not on the PC).

4. In the Dialog Manager code, define sizing variables to allow sizing/placement of GUI objects based on a run-time parameter.

Using these rules, we are able to maintain a single version of the client source code which may be simply transferred between platforms, and recompiled in order to work.

15. Final conclusions: What would we have done differently?

Many of the elements discussed in this paper may be completely obvious to you, but they were not to us. Having learned all these things during development, I feel safe in giving the following list of things I would not choose to do again:

1. Develop a server completely from scratch with prototyping using C or C++.
2. Use a GUI development tool that has no support resources in this country, or at least has no support resources in a compatible time zone.
3. Use a GUI development tool that does not support multiple platforms.
4. Begin development without a relatively clear picture of the technology to be used.
5. Agree to add major new functionality in the middle of the development cycle without extending the schedule.

Now, here are the things that I did not do in the project that I would, were I to do it again:

1. Develop Elementary Process and Triggering Event definitions to ensure completeness of the defined business transactions.
2. Develop state transition diagrams for each business area element to further ensure this completeness.
3. Have the person developing the system test plan on the team for at least 4 weeks prior to the start of system test.
4. Develop the installation process early on to allow its testing/use for installing all prototypes. This would exercise the process more.
5. Get more users from geographically and functionally diverse areas together sooner (in an early JAD) to define the initial prototype design.
6. Obtain training and experience in the tools to be used in the project before the clock starts ticking.

In summary, RAD and client-server are not incompatible, but rather mutually contribute to the quality of the end product. I believe that both the RAD approach and client-server are here to stay, and that it is essential that we, as IT professionals, master the skills and techniques necessary to implement them. If we don't, someone else will.

Paper Number 1004
A Comparison of Distributed Application Technologies

Gerald Duggan
MTS/Software Development Engineer
c/o Ella Washington
Hewlett-Packard Co.
408.447.1053

Handouts will be provided at time of presentation

Using High-Performance Real-Time I/O with HP-UX

Robert T. Cleary

KineticSystems Corporation
900 N. State Street, Lockport, IL 60441
Telephone (815) 838-0005, Ext. 259

Abstract

High-scan-rate data acquisition and UNIX-based computer operating systems are often considered to be incompatible because of the lack of deterministic response in these operating systems. Most data acquisition systems require CPU intervention for each scan clock 'tick'. Some systems need CPU action for each channel being acquired. This paper describes a number of real-time I/O systems that combine Hewlett-Packard computers or workstations operating under HP-UX™ with high-throughput real-time I/O front-ends.

The many utilities and network connectivity associated with a full-featured operating system (OS) such as HP-UX make it an attractive choice for use with a data acquisition system. However, such systems that operate above a slow to moderate channel scan rate are subject to data overrun, because the OS may be servicing other processes when it is needed to handle the time-critical data from the data acquisition front-end.

The systems described in this paper use real-time I/O based on the CAMAC and VXibus standards. The operating system latency of a typical system can limit the per-channel data acquisition rate to less than one sample per second per channel. However, by using the list processing and multi-scan buffering hardware techniques described here, one system scans 3500 analog channels with a per-channel scan rate of 10 milliseconds, giving a continuous real-time I/O throughput to the host computer of 700,000 bytes per second.

Background

There is an increasing need for test and measurement systems with a high throughput rate to the host computer. Some examples of this include jet aircraft, rocket and automotive engine testing, munitions studies, automotive safety tests, large-scale physics experiments and petrochemical process modeling. In fact, many test applications that formerly involved only a few data channels at a relatively slow scan rate now require a larger number of channels at a rather high scan rate. A parallel trend involves the movement to UNIX-based computer operating systems. The primary advantage of *open* UNIX systems is the portability of software applications from one operating system to another. However, the lack of deterministic response to real-time tasks in most UNIX-based operating systems can cause loss of data in high-performance data acquisition applications. This problem extends to HP-UX.

To help solve the data acquisition system response issues, real-time UNIX operating systems have been developed. These include VX-works™, LynxOS™ and HP-RT™. These kernels provide a high degree of deterministic response and are optimized for real-time tasks. Many users, however, strongly prefer the rich environment of a true UNIX operating system, such as HP-UX for the networking, display and other facets of their test applications. This paper

describes several techniques that are applied in the data acquisition system hardware to minimize the effects of operating system overhead and lack of deterministic I/O response. While these techniques are applied to systems using HP-UX, they can provide enhanced performance on systems that use real-time operating systems such as HP-RT.

Basic Approaches to Enhance Real-Time Performance

Since the primary problem is the lack of deterministic response, with the response delay representing a range of time for an I/O call to complete, the basic approach is to make these I/O requests less often and to move as much data as possible in a single DMA (direct memory access) block associated with each request. Several methods to accomplish this are described in this paper. They are:

- Provide scan processing within the data acquisition system so that the host computer deals with a single DMA block of data representing all of the channels contained within a single scan.
- Use a multi-buffer memory in the data acquisition system interface so that the data words from a number of scans are contained in a single DMA block.
- Include one or more auxiliary processors in the data acquisition system, if control loops are needed, so that the host computer is not burdened by this processing requirement.
- Include local memory in the data acquisition system for transient data capture. Acquire the data into this memory, then transfer the data to the host computer between data acquisition bursts.

Scan Processing

A number of system configurations based upon the CAMAC (IEEE-583) standard have been implemented to reduce the effects of the operating system. One of these configurations is shown in Figure 1. A Hewlett-Packard 700-Series computer is the system host. The CAMAC I/O chassis contains a controller that is interfaced to the host via a SCSI (Small Computer System Interface) channel. For maximum throughput, it is important that this be a second SCSI channel, separate from the one used for the storage devices. This approach allows the acquisition and storage of data to be interleaved operations. A standard CAMAC chassis contains slots for the controller plus 23 I/O modules. These modules may be connected to thermocouples, strain gages, pressure sensors, switch contacts or other input transducers for data acquisition. Output modules may be included to drive power supplies, relays or other output devices for control. If more than one I/O chassis is required, additional controllers can be connected to the SCSI bus. Information regarding the CAMAC (Computer Automated Measurement And Control) standard may be obtained from the Institute of Electrical and Electronic Engineers [1].

On an HP 750 workstation, the overhead for one SCSI transaction is about 8 milliseconds. If we assume that we are storing the acquired data in system RAM and there are no other processes running, the time to acquire 500 channels of 16-bit data, *without any list processing*, is:

$$500 \times 0.008 \text{ seconds} = 4 \text{ seconds}$$

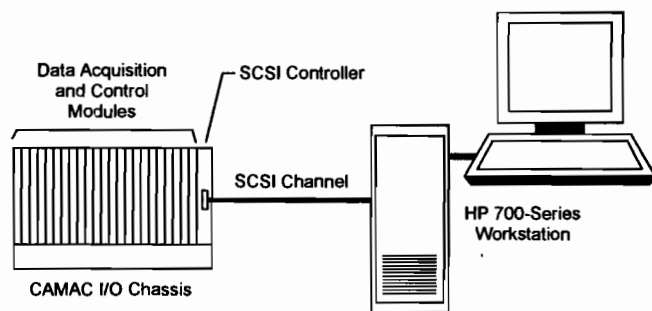


Figure 1 - A CAMAC I/O chassis interfaced to an HP 700-Series Workstation via SCSI.

Under this scenario, the absolute maximum per-channel scan rate is about 0.25 Hz (a 4-second period). If we assume that 75% of the time is consumed by other processes, this rate is reduced to 0.0625 Hz (a 16-second period between scans). In this example, DMA block transfers cannot be used because the host processor must select the channel address for each word to be transferred.

The CAMAC controller shown in Figure 1 contains a hardware-driven list processor that includes a 64-kilobyte storage memory. Prior to data acquisition, one or more scan lists can be downloaded to the list memory. A typical list instruction consists of the following elements:

- The module slot number for the command to be executed (N)
- The subaddress—generally, defining a data channel—within the module to be accessed (A)
- The function to be performed—read, write, or dataless command (F)
- The list execution type—single-word transfer, block transfer, embedded write, list end

The four basic types of list commands are described here:

- *Single-word transfer*—This command will transfer read or write data for one channel or execute a dataless command, depending upon the function (F) selected.
- *Block transfer*—This command will transfer a block of data, typically to or from a RAM memory in an I/O module. The register location is defined by the NAF, and the block size is determined by an additional word associated with the instruction in the list.
- *Embedded write*—The computer channel must be declared as *input* or *output* for an entire block operation. However, for a read-block transfer, it may be desirable to include some write operations with predetermined data for triggering, gain-change or other similar operations. These embedded write commands are included in a read list, and the desired write data is in the list associated with each such command. When an embedded write command is executed in the list, no computer channel transfer occurs.
- *List end*—When this command is executed, list execution ceases, and the block transfer is concluded.

This list protocol is simple, yet it is quite powerful. Figure 2 is a block diagram that depicts the relationship between the host workstation or computer, the CAMAC controller, and the real-time I/O modules. Acquisition of the data for one scan of *all* of the channels is initiated with a single I/O call. Again, assuming an HP 750 workstation, the overhead for one SCSI call is 8 milliseconds. The DMA throughput for this configuration is one word every 1.33 microseconds. The total acquisition time for 500 16-bit channels is:

$$0.008 + (500 \times 0.00000133) = 0.00867 \text{ seconds}$$

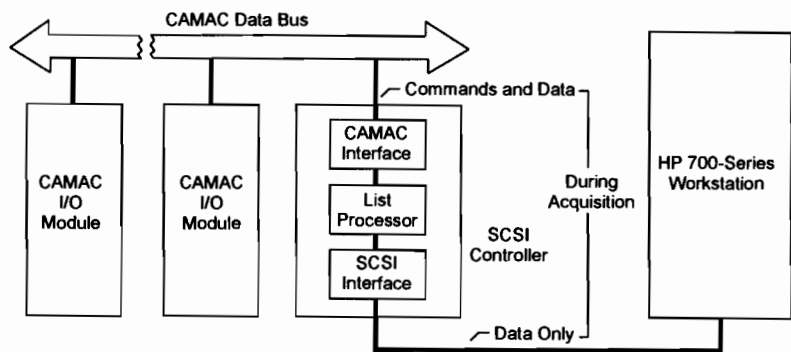


Figure 2 - A block diagram of the SCSI-interfaced system including list processing.

Under this scenario, the absolute maximum per-channel scan rate is about 115 Hz (a 0.00867-second period), with computer overhead representing over 90% of the time. If we assume that 75% of the time is consumed by other processes, this rate is reduced to about 29 Hz. *This is nearly a 500-fold increase over the maximum scan rate without the list processor.* A hardware-driven list processor was used rather than a general-purpose CPU to provide guaranteed deterministic response and extremely low latency. This example depicts data acquisition. However, write lists can be downloaded and selected for control operations.

Multi-buffer Memory

In the previous example, data acquisition throughput was increased dramatically by the use of a list processor to select the channels to be scanned and a single DMA block for one entire scan. While this is sufficient for many systems, some applications require even higher throughput. An example is given here of a jet aircraft engine test requirement with 3500 analog channels that are geographically distributed, and a complete scan must be executed at a 100-Hz rate. The host computer is an HP 827.

This system uses the CAMAC Serial Highway per IEEE Standard 595 [2] along with block transfer enhancements [3]. The topology of the system is shown in Figure 3. Three distributed I/O chassis are interconnected by 50-megabit-per-second fiber-optic links. This highly

deterministic highway system was developed for data acquisition and control, and it avoids the variable delays found in local area networks such as Ethernet or FDDI. The highway driver contains a list processor, just as in the previous example. In addition, it contains a 4-megabyte ping-pong multi-buffer. This memory allows the acquisition of data at one rate and the transfer of that data to the host computer at a submultiple of that rate. The timing for a typical example is shown in Figure 4. This example assumes that we are acquiring data from 3500 analog channels every 10 milliseconds and transferring that data to the host computer at a 100 millisecond rate. The 10-millisecond 'tic' is controlled by the highway driver. By using a scan list module in each chassis in addition to the list processor in the highway interface, we can do the following:

- Acquire data over the fiber-optic highway at a rate of one channel per microsecond, since the channel addresses are contained in the scan list module and do not need to be transmitted over the highway. The list processor selects a block of data for one complete scan from each chassis. The total time to acquire data from 3500 channels is 3.5 milliseconds. If this optional module were not used, the highway rate is reduced to one channel every 5 microseconds, making the acquisition time 17.5 milliseconds.
- These 3.5-millisecond bursts of data are loaded in Buffer 'A' of the multi-buffer memory. This process is continued until 10 scans have been completed. The acquired data bursts are then transferred to Buffer 'B', and the process is repeated in a ping-pong fashion.

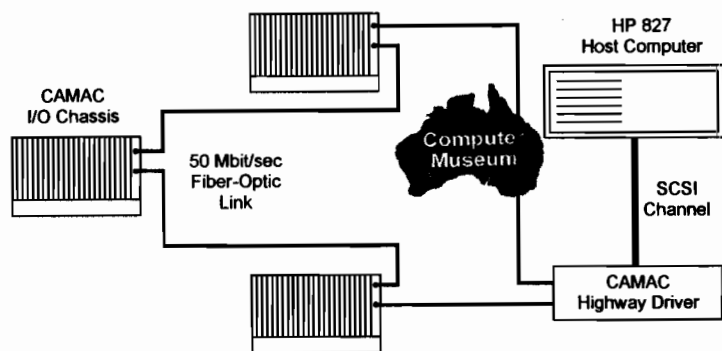


Figure 3 - A CAMAC Serial Highway system interfaced to an HP 827 computer.

To increase system efficiency, a DMA request for Buffer 'A' is made soon after the DMA transaction acquiring the data from Buffer 'B' is completed. By this mechanism, the data from Buffer 'A' can start to be moved to the host computer as soon as it is filled. This interface supports synchronous SCSI with a DMA transfer rate of 3.5 megabytes/second. The transfer time from the buffer to the computer for 10 scans of 3500 channels (35,000 16-bit words or 70,000 bytes) is about 20 milliseconds. The DMA pre-request allows the SCSI channel setup time to be "hidden" between data fetches.

The physical size of the multi-buffer is 4 megabytes (2 megabytes for each buffer). However, the logical size of each buffer is programmable to fit the particular requirement. In this case,

each buffer was set to become "full" after it received 70,000 bytes of data. For this example, the multi-buffer size could have allowed the period between transfers to the host to be as long as 28 seconds, while still keeping the 10-millisecond I/O scan. The 100-millisecond value was chosen to meet the desired refresh rate for near-real-time display of some channel data. This configuration has been operating at several sites for a period of up to one year without data overrun problems. While this application uses 800-Series computers for hosts, 700-Series workstations would also provide satisfactory results.

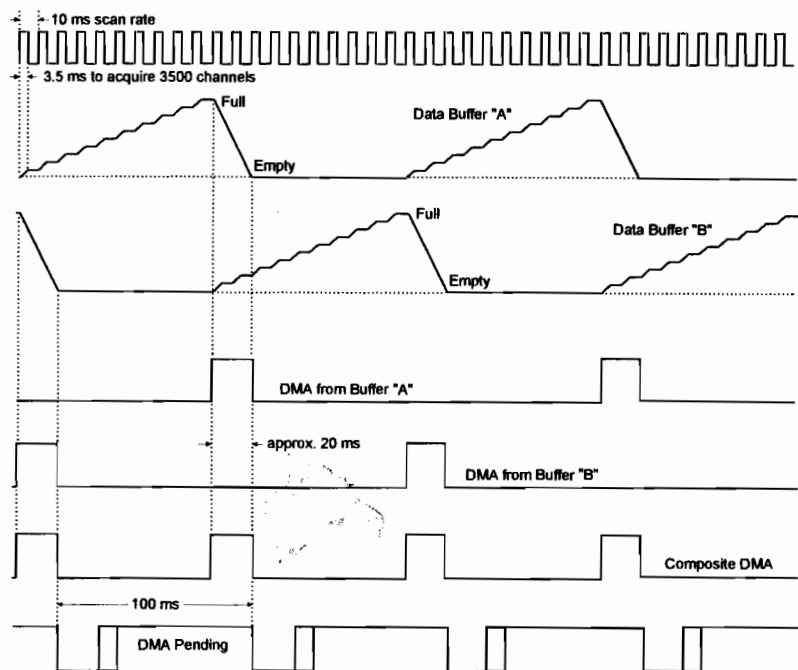


Figure 4 - The scan timing relationships for the Serial Highway system.

Auxiliary Processors

The previous examples represented basic data acquisition requirements. Another class of data acquisition system also includes a number of tightly-coupled control loops. The initial application of the system described here involves development test cells for automotive transmissions. Each system contains approximately 400 analog channels or 16-bit discrete channel equivalents (approximately 300 input channels and 100 output channels). About 80% of the channels are scanned at 10 Hz, with the remainder at 100 or 1000 Hz. The most difficult requirement was to provide 10 PID (proportional integral differential) control loops with 1000-Hz response.

The system configuration is shown in Figure 5. Each of these CAMAC systems is interfaced to an HP 755 workstation via the EISA channel. The CAMAC controller contains a list processor that supports multiple scan rates. Jump instructions in the list allow the higher-rate channels to be scanned a predetermined number of times for each scan of other channels at the lowest selected rate and placed in the data buffer. By this means the 10, 100 and 1000 Hz scan rates are accommodated and the data is transferred to or from the host workstation at the lowest rate—10 Hz in this example.

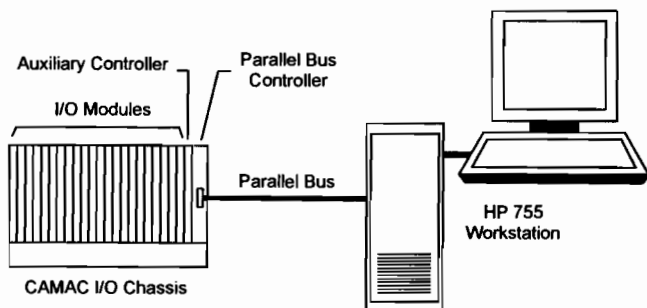


Figure 5 - A CAMAC system with an auxiliary processor for control loops.

The 100-Hz control loops are accommodated by a 68030-based auxiliary controller module [4]. A block diagram of the CAMAC subsystem is shown in Figure 6. Most of the input and output channels are scanned directly by the main controller. The remaining channels involve the fast control loops and are scanned by the auxiliary controller. In order to maximize the throughput of those input/output loops, no operating system is used on the auxiliary controller, and the routines are written in assembly language or C. These routines are downloaded from the host during system initialization, using the dual-port memory in the auxiliary controller. During the test, loop parameters and other data are loaded in that controller and status is read by addressing the auxiliary controller just as one would access any other module in the chassis.

Transient Data Recording

In the examples given earlier, scan processors were used to allow the data from one scan to be brought into the host computer as a single data block, and multi-buffer memories were used to combine the data from n scans into a single block that is transferred to the computer, reducing the effects of operating system overhead and latency. This assumes the continuous acquisition of data for a relatively long period of time. Another class of data acquisition is usually called transient recording. Generally, the data is acquired for a relatively short period of time, and the aggregate data rate is higher than can be acquired directly to computer memory.

A typical CAMAC-based data acquisition system connected to an HP 700-Series workstation is shown in Figure 7. The characteristics of this system are shown here:

- The CAMAC controller is connected to a SCSI bus on the host workstation.
- Eight, 2-channel, 500-kilosample/second transient data recorders are provided, and each recorder is connected to a 4-megasample local memory module.

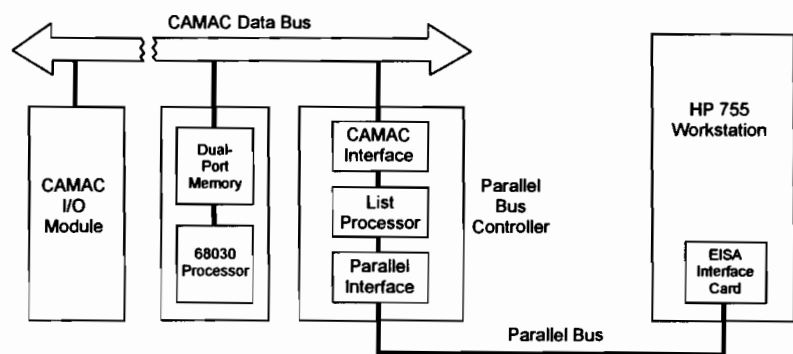


Figure 6 - A block diagram of the CAMAC system with an auxiliary processor.

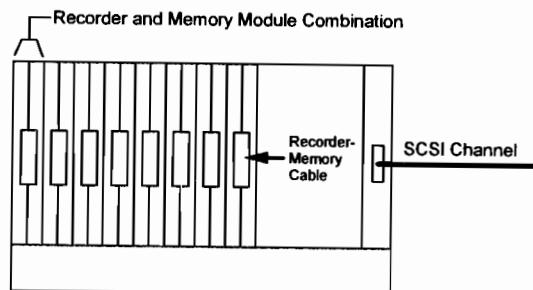


Figure 7 - Transient recording with local memory associated with each recorder.

The transient recorders can be programmed for scan rate, logical memory size and amount of pre-trigger and post trigger samples. Assume the following setup for this example:

- The recorders are to sample at 500 kilosamples/second, using the internal clock from the first recorder.
- The logical memory size is set to 4 megasamples (4,194,304 samples).
- The sample mode is set to 1/8 pre-trigger, 7/8 post-trigger.
- An external trigger is used.

The recorders are started under program control, and data is stored in the memory in a circular fashion. When an external trigger is received, the recorders continue gathering data for another seven-eighths of the logical memory size. Since 2×500 ksamples of data are stored in each memory, slightly more than 4 seconds of data are recorded. The total data stored after recording is:

$$8 \times 4,194,304 = 33,554,432 \text{ samples (67,108,864 bytes)}$$

If we assume that the data from the memories can be transferred to the host computer and stored on the hard disk at an 800-kilobyte/second rate, then it would require about 84 seconds to store the data and be ready for another test. A wide range of recorders is available, including one module from another manufacturer that can record data at a 1.3 gigasample/second rate.

Conclusions

This paper has described several methods for coupling high-performance data acquisition and control systems to computers or workstations operating under HP-UX. The key ingredient to good real-time performance is to remove the burden of channel selection from the host computer and to perform that selection in the I/O subsystem. Systems conforming to the CAMAC international standard have been implemented with the following characteristics:

- Local scan processing, presenting the data from each I/O scan to the host computer as a single data block. In the example shown, a 500-fold improvement in throughput was achieved, compared to CPU-controlled scanning.
- Scan processing with a multi-buffer memory added to allow n scans to be presented to the host, further reducing the effects of overhead and latency. Systems that scan 3500 analog channels at a 100 Hz rate have been implemented using this approach.
- An auxiliary processor can provide I/O preprocessing or execute control loops. Systems executing ten 1000-Hz PID loops have been demonstrated.
- When high-speed scanning must occur for a relatively short time, transient data recorders can be used with local memory to buffer the data. Even for this case, a rather efficient method to transfer the data to the host and storage may be important to reduce the minimum time between tests.

An important consideration in the high real-time throughput that has been discussed in this paper is the use of hardware-driven list processing to provide the low latency and high throughput. While these solutions are applied to host computers running HP-UX, the same techniques would improve the data acquisition performance of real-time kernels such as HP-RT. In addition to the systems described here that conform to the international CAMAC standards, similar interconnects are being developed for the open VXI (VME eXtensions for Instrumentation) standards.

References

- [1] ANSI/IEEE Standard 583, "CAMAC Instrumentation System," Institute of Electrical and Electronic Engineers, 445 Hoes Lane, PO Box 1331, Piscataway, NJ 08851.
- [2] ANSI/IEEE Standard 595, "CAMAC Serial Highway System," Institute of Electrical and Electronic Engineers, 445 Hoes Lane, PO Box 1331, Piscataway, NJ 08851.
- [3] R. T. Cleary, "Enhanced CAMAC Serial Highway System," *Nuclear Science Symposium*, San Francisco, CA, October 23-25, 1985.
- [4] ANSI/IEEE Standard 675, "Multiple Controllers in CAMAC," Institute of Electrical and Electronic Engineers, 445 Hoes Lane, PO Box 1331, Piscataway, NJ 08851

**Paper Number 1006
Migrating COBOL to HP-UX**

**Alan Meyer
Software Development Engineer
c/o Ella Washington
Hewlett-Packard Co.
408.447.1053**

Handouts will be provided at time of presentation

A Practical Guide For Integrating Imaging into Your Existing Business Applications

John J. Carney
Director, Image Application Product Development
Wang Laboratories, Inc.
One Industrial Avenue
MS 014-790
Lowell, MA 01851
(508) 967-7071

At first glance, you might think this paper is a discussion about imaging or your business applications. In fact, while we do discuss both imaging and your business applications, the primary topic is *practicality*.

When you consider any technology improvement for your business, I'm willing to bet that practicality is far more important to you than the technology itself. Your day-to-day business decisions center around what is best for your business. You evaluate where you are, where your business is going, and how you plan to get there. When you decide to integrate imaging technology, your decision is evaluated in the context of technology you have already purchased: technology you have invested in; technology that you have spent time and money training your people to use.

In other words, you determine what imaging technology will help you to continue to win, given the context of your surroundings. You must determine what is *practical*.

The dictionary defines *practical* as *relating to practice or action, rather than theory, speculation or ideals*. In a business context, practicality is what works. Practicality means that you can find a solution.

As you look for a solution, you may dream about and discuss the ideal. Words like *quick, easy, inexpensive, and effective* invariably creep into the discussion. Furthermore, the practical decision is generally one that is dependent upon past decisions. This means that the practical decision for one business is not necessarily the same as the practical decision for another. What is practical for me is not necessarily practical for you.

IMAGING: A PRACTICAL PERSPECTIVE

You have been reading about imaging. You've heard how it has revolutionized other businesses. Now, your competition is about to add this technology, so you're worried. From all you know, imaging would enable you to meet your customers' needs more effectively. You also hope that it would reduce your requirements for storing paper documents. But, you wonder, how can I make it practical for my business? How can I continue to use the applications that my business has come to depend upon? After all, as important as it may be, image technology is but one component within your business' universe. Your business is centered around existing mission-critical applications that often impact your entire enterprise: they are integral to running your business. Without them, your day-to-day business would grind to a halt.

How do you reconcile the need for new technology with the need to maintain existing applications that are your lifeblood?

I've talked to other customers and users. Their answers are uniform and clear:

- "By far, the *most important factor* (when selecting an image system) was the ability to *integrate* the system with *existing* equipment and *software*."

AIIM Study

- "The big payback, for almost all imaging, is *tying image* with *existing* data processing applications."

Systems 3X/400

- "Some form of *integration* with the *mainframe* is crucial."

Michael L. Thomas
Past President, AIIM

You know that imaging technology is important. You suspect that it may become even more important as it proves itself within your enterprise. But, for now, the imaging that you choose must work smoothly, integrating with your existing business applications.

MERGING THE OLD WITH THE NEW

Today's discussion will center around image systems and integrating them with business applications. We will take a look at five architectural models: What are they? Which ones work best with existing applications? How do you choose the best system for your needs? How can you tie imaging technology to your business' applications?

We discuss some of the practical considerations you need to make; considerations based on real world experience. We also discuss criteria for system evaluation and selection.

ARCHITECTURAL MODELS: WHAT ARE THEY?

Essentially, a business considering image technology has five options. Today, the following five architectural models encompass your imaging choices:

- Screen-level integration
- SQL integration
- Application-level integration
- Cooperative processing
- Workflow/reengineering, which is also called business process reengineering

What factors differentiate these five options?

Screen-level Integration

Screen-level integration, sometimes called presentation level integration, is the simplest model. It provides a means of binding imaging to an application, but transparently: it layers the imaging over the existing application. Integration usually takes place at the desktop, by tying the images to the application screen data. PC-based, this method connects the image services to the application, usually through Microsoft Windows-compliant PC software, often a terminal emulator. This software "screen scrapes", or extracts keystrokes or displayed fields, and passes this information to an image system, which displays the requested image corresponding to information on the application screen.

SQL Integration

A second architectural model for image technology couples the image technology with an application's SQL database. In this model, the application and images are integrated at the user's desktop: Using SQL statements, the desktop application queries the database server. The result list is used to compose image identifiers, which then retrieve images from the server to be acted upon.

Application-level integration

Application-level integration tightly couples the image technology with the application. The integration takes place on the application platform, as opposed to on the desktop. The application directly accesses the images, usually using an Application Program Interface (API). Document identifiers stored in the application database are used to retrieve images and to perform functions on them.

Cooperative processing

The cooperative processing model treats the image technology and the application as equally important. The application, client and server work together in a peer-to-peer relationship to redistribute the processing load between desktop and server. This model typically involves LAN or even WAN connectivity.

Workflow/Reengineering

The workflow/reengineering model views imaging technology as a component of work management software. The work management software is used to reengineer and streamline business processes, resulting in a system that guides and controls those processes using those work management rules. Bruce Silver, an analyst for BIS Strategic Decisions, says this model often involves "Destroying a business process in order to save it." While many workflow/reengineering initiatives are less drastic, the ability to manage workflow is almost always the primary goal --imaging is only one component of the work management solution.

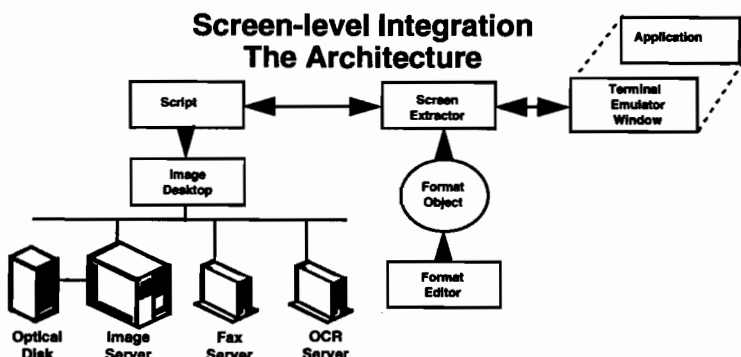
Architectural models: Which are appropriate where?

If your goal is to integrate imaging into existing business applications, screen-level integration or application-level integration are the two models to consider. These architectural models are appropriate for both new or existing applications. They make the best use of a business' installed infrastructure: hardware, software, networks, training, and existing staff.

The other architectural models are best suited for new applications, or for organizations ready to make dramatic changes in the way they do business. While SQL integration may use old data, it generally makes use of a new application. Cooperative processing brings to mind concepts like downcosting and rightsizing, and typically involves a move to distributed processing. In all likelihood, changes to the application will be extensive. Finally, enterprises considering workflow and reengineering models are generally rethinking their business processes, and will drastically modify the business application.

For businesses that want to integrate imaging with existing applications, screen-level integration and application-level integration offer the practicality most require. These methods are the most popular for image enabling existing applications today. These methods allow you to enhance your existing investment in applications, training, and personnel. The rest of this presentation focuses on these two models.

SCREEN-LEVEL INTEGRATION: QUICK, COST-EFFECTIVE IMAGE-ENABLING



Screen-level integration is a technique that layers imaging over the existing application. The application is running on a legacy host environment or minicomputer. For most screen-level products, a terminal emulator accesses the host application. The application is viewed through a terminal emulator on a desktop. A screen extractor program, the screen scraper, provides data extraction from the emulator. This program uses a DDE script to pass the data onto the image system. The script edits and checks the data, then passes it to the image system, which performs the actual imaging functions. All processing takes place on the desktop.

Screen-level Integration: Advantages

Screen-level integration offers many advantages. Many users find it especially appealing because it does not impact the existing application: screen-level image technology and the existing application remain separate entities, integrating the application and images at the user's desktop. Since the application remains separate, it requires no changes to the application. Production systems, which are often a business' most important systems, are not impacted.

Screen-level integration offers flexibility. When properly implemented, it offers the accessibility to multiple platforms from multiple vendors. This platform independence enables a business the security of knowing that a change of platform will not impact its mission critical data. Screen-level integration allows a business to keep its options OPEN.

End user groups are well served by screen-level integration for a number of reasons. The application is not altered. The source code is not even required. Second, no programmers are required -- individuals with a minimum of technical expertise can image-enable the application. With screen level integration, an end user department can image-enable quickly and autonomously. Development is relatively fast and simple. The imaging is in place in hours, not weeks, and the software and the people resources required to put the system in place are comparatively inexpensive.

If you are uncertain about the benefits of imaging for your business, screen-level integration provides a relatively low-risk method to pilot the technology. It is also a good model for demonstrating proof-of-concept, to show that imaging technology is worth the effort.

Screen-level integration: Disadvantages

While screen-level integration is easier to implement than other architectural models, many of the features that make it easy present difficulties. For example, administrative tasks such as backup and recovery are far more difficult -- the application is unaware of the existence of the image-enabling. Application utilities are useless. Image security is an issue because the normal security afforded with the application's database is not available for the image repository. Sometimes, the lack of file security and lack of password protection result in administrative headaches or security breaches for IT personnel.

The user interface is cumbersome and complex. The user interface generally requires three windows; even more with multi-session applications. With multiple windows to manage simultaneously, the chance of user errors increases. Moreover, since image location is a multi-step process that runs on a PC, it can be very slow.

Finally, screen-level image-enabling does not adapt well to new releases of the application software. Screen-level integration relies on precise screen layout and positioning, and any change means coordinating both the host application and the imaging application. Then all of the image workstations in the configuration must be updated. This means that if a field used as a key for the imaging application moves three spaces to the left, the image application will fail on all desktops. This requirement for consistency can result in administrative headaches.

Screen-level integration: Practical advice

Our primary morsel of practical advice is given in the spirit of Imaging World's columnist Herb Edelstein, who says, "A good consultant doesn't have all of the right answers -- just all of the right questions!"

As you explore various screen-level integration systems, be sure to ask the right questions. I've listed some categories and questions that need to be addressed.

Support concerns: keep your options broad

- What range of interoperability does this system offer?
- What host platforms does this system offer?
- What terminal emulators are supported?
- Will it support both MS DOS and Windows?

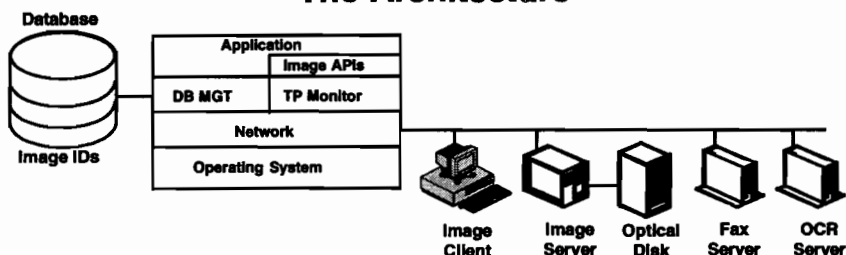
Technical Issues: keep it simple

- What scripting tools are available?
 - Must they be used?
 - Is your favorite one of them?
 - Do your end users know how to use them?
- Does the screen-level integration product require any MS Windows programming?
- What steps must be taken to extract and update application screens?
- Does the image system use a point and click editor for screen changes?
- Are both color and grayscale image types supported?
- How can screen layouts be consistently defined for all desktops?

Concerns for the future: keep your options open

- Is screen identification dynamic and sequence-independent?
- How scalable is the system?
 - How many clients can it support?
 - How many servers?
 - How many types and sizes of optical peripherals?
- How large do you project your user community will get?
- How big can the image repository grow?
- When does the system "run out of gas" or "hit the wall?"

Application-level Integration The Architecture



Application-level integration is an industrial strength approach to integrating existing business applications. If both approaches were laundry detergents, application-level integration would be classified as "industrial strength." In terms of implementation, the key difference between screen-level and application level integration is the degree of integration provided between the application and the image repository.

With application-level integration, the application database contains both mission-critical data and the image document identifiers. This means that the application can use the Application Program Interfaces to perform imaging functions such as printing images on a LAN-based printer, displaying images on a client's desktop, or retrieving them from the server. The key here is the application, and the imaging accessibility through APIs -- the application, the data and the imaging system are tightly linked through the APIs.

Applications image-enabled in this manner are generally easier to use, administer, and maintain, since they are extensions to existing applications, not newly constructed ones. They also provide tighter control over image services, allowing the image subsystem to be controlled by the application's native security mechanisms.

Application-level Integration: Advantages

Application-level integration offers a number of advantages. First, image location is faster than with screen-level integration. Since the image document identifiers are stored in the application database, the application can immediately and directly retrieve an image.

The application is in control, which makes the user interaction "bullet-proof." The application and the image system are tightly coupled, providing the images with the same level of security as with the original application. This inherited security also means that the integrity of the imaging repository is the same as for the rest of the application, and that passwords, database access, and file security set for the application are the same for the image system. All lines of defense remain in place.

Systems with application-level integration are easy to administer because the application "knows about" the imaging repository. Software release levels are more easily managed because the application knows where the images are -- it's not dependent on the desktop scraping them off the screen each time. A company can make use of MIS or IT expertise to help manage the images.

The user interface is simpler, with only application and image windows to manage. The simpler interface (two windows instead of three) helps control user error. And capabilities at the desktop can be limited by means of APIs, thus offering even more protection to the image repository.

With application-level integration, the user can access host application data that is not presented on screens. Report information or computed data from the application itself can be included.

Application-level Integration: Disadvantages

For all of its advantages, application-level integration does present some disadvantages. First, it requires application source code: since the source code must be modified, the source code must be found, written or licensed. Moreover, it takes programmers to change the source code; programmers who are familiar with the application. Finally, your platform choices are limited to those available through your image vendor.

Application-level integration: Varying levels of integration

Application-level integration can take place in a variety of ways. Let's look at some image-enabling scenarios.

In a typical third generation language scenario, the same application that accesses the database accesses the image repository by means of APIs. Common 3GLs in use today are COBOL, C, and FORTRAN. The language calls the image API to access image documents or the data dictionary for database management. Some have access to the network. The integration is very straightforward: the 3GL calls the APIs, retrieves the image ID from the database, and makes an API call to display or print the image.

In a variation on the 3GL scenario, much of the 3GL code is generated by software power tools such as application generators. This code can be customized by your programmers. In very sophisticated shops, a CASE tool is used. When these tools are image-capable, the job is much easier.

Fourth generation languages from vendors like Oracle, Computer Associates, and Software AG provide a high level of application portability and programmer productivity. The 4GL application is usually written in platform-independent language, and can directly access the imaging APIs. When image integration is built into the language, images can be accessed as easily as the database.

Any of these image integration scenarios yield significant payoff. Where is the optimal point to integrate image technology? Generally, the more sophisticated techniques offer tighter integration, better application portability, and a less extensive development cycle. However, it is important to have cooperation from your vendor. In general, database tools must be integrated by the database vendor.

Application-level integration: Practical advice

As with screen-level integration, it's important to ask the right questions. The right answers will vary from installation to installation.

Allow yourself choices

- How many processors are supported?
 - Does this vendor support the processor you are now using?
 - Does this vendor support the processor you plan to use in the future?
- Can you image-enable your current environment, then bring it forward to your new environment?
- Will your vendor support your TP monitor?
- Is your LAN or network operating system supported by your vendor?
- Can the image application tie-in work over a broad range of connectivity options?
- What are your image server options? Must your server be PC LAN-based, or are more sophisticated options available?
- Are your workstations and PCs supported? Can your character-based workstations be tied into your imaging application?

Think carefully about the options you require, both for your current installation and for the eventual system and platform that you will have in place.

Allow yourself the three most important choices

Be sure you know precisely how portable your application is. You want to keep your options open in three broad areas:

- **Platform Independence:** As you choose your imaging solution, you want an application programming environment that allows consistency across your required platforms. You want to be able to image-enable on one platform, and when ready, move this imaging solution to another platform, without rewriting code. Make sure that the API is consistent across many environments.

- **Database Independence:** Are you tied to one database, or can many databases be used?
- **Language Independence:** Are the proposed tools language-independent? If there are language requirements, do they match your staff's skill set? Will the language requirements serve your next, probably more open, platform?

It is *practical* to ensure that your investment is protected.

Go with standards

You want to be sure that your proposed vendor supports industry standard technology. Think about each component in the architecture:

- The system, both client and server
- The user interface: Generally, Microsoft or UNIX GUIs
- Documents: TIFF is the de facto standard; and RASTER is supported by ANSI, ISO, AIIM, and CALS
- Compression: CCITT? JBIG for black and white images, and JPEG for color
- Peripherals: SCSI for optical disks and TWAIN for scanners
- Media: Rewriteable 5 1/4" magneto optical disk

Look for an open and easy-to-use development environment

You want an imaging solution that is just that -- a solution, not one more problem that interferes with day-to-day business.

If a third generation language is used, is it open or proprietary? Are your programmers familiar and comfortable with it? Does the vendor include sample image applications? Is the source code available for these sample applications so that you can customize them to suit your business' needs?

If the vendor uses a fourth generation language, is it one that is used by industry leaders? Is solid image integration built into the 4GL tool, the application generator or CASE tool? The integration between the database and imaging should be in place when you purchase your application.

Don't let the architecture box you in

You want a solution that allows you to image-enable your mission-critical application *once*. To ensure this, ask questions about the following topics:

- **Scalability:** As you scale the system, do you have to change the application? The answer you are looking for is "no." Ask specifically about the application host, the clients, the servers, and the peripherals.
- **Performance:** What will happen to application response times and system throughput as you scale the system? The system should be designed so that performance is unaffected, both today and for the predictable tomorrow.
- **Coexistence/Migration:** Is there a clear path to your future system? Will you be able to set the pace for any migration?

SUMMARY

Screen-level integration and application-level integration are the two best alternatives for a practical approach to image-enabling existing business applications. Each offers advantages and disadvantages that must be weighed carefully.

If you ask careful questions, and evaluate your options clearly and objectively, you will have chosen the most practical option -- a solution that wins for you and your business.

IMAGE-ENABLING METHODS COMPARED

DIFFERENCES

Screen-level

Image technology transparent to coupled application

Application screen data drives image access

Enabled at user's desktop

PC script extracts data from application screen and uses as image identifier with function

Effort measured in minutes

Application-level

Image technology tightly

Application accesses images directly using APIs

Enabled at application platform

Application calls API with image identifier and function

Effort measured in days

STRENGTHS

Screen-level

Flexibility

Ease of customization

Rapid development by non-programmers

Inexpensive

No application changes required

Application-level

Tightly coupled image services

Ease of administration

Immediate image location

Security inherited from application

Processes image requests from all connected devices

WEAKNESSES

Screen-level

Slow image location

More prone to user errors

Sensitive to screen changes

Less secure

Hard to administer

Application-level

Requires application source code changes

Requires application programmers

Platform choices limited

#1008

**Future CASE: Enabling End Users to Customize
Applications**

Ken Robinson

Auto-trol Technology Corporation

12500 North Washington Street

Denver, CO 80241-2400

303-252-2815

e-mail: kenrob@Auto-trol.com

1.0 Abstract

CASE tools have been in use for some years now, and while some software developers are reaping the fruits of the automation revolution, most end users are shackled to extremely complex high-level language APIs and complicated and often mysterious proprietary macro languages. CASE must grow into object technology, repository standardization and common notation methodologies, and as it does so CASE-like techniques must be merged into commercial software to aid the user in customization.

This paper intends to present alternatives to end user programming in the form of CASE-like tools as an aid in customizing software products in an interactive, graphical fashion. Formal, discipline-specific notations/languages versus general programming languages will be discussed. Graphical application development environments will be examined, as well as their integration into commercial software applications. Programming by demonstration will be touched upon as an adjunct to end user customization and a viable successor to keystroke or macro logging.

An example of the problem to be solved: a mechanical designer does not want to learn and understand the delicate syntax of a language such as C++ in order to add a new "twist" to an existing operation in his/her solids modeling application. Such an application should have a customization interface that would allow one to "drag-and-drop" objects, methods and data into existing functional areas to alter their behavior without forcing users to write code. The terminology and notation used in this type of interface would match the discipline the user was working in. This kind of customization tool could start as a simplified CASE-like diagramming notation, combined with a class browser in an application framework similar to HP's SoftBench, Borland C++ for Windows, and many of the popular 4GL products such as PowerBuilder, which are used to build relational database applications.

2.0 Introduction

Recently, I volunteered to assist our Educational Services Department at Auto-trol with a customer training issue. They needed someone to spend a day with a gentleman from an aircraft manufacturing firm in Wichita, giving him an overview of programming with one of our proprietary macro languages.

As I began to chat with this gentleman about his expectations and his knowledge of computers in general, he made a very profound statement without even realizing it. He first told me that he was an artist/illustrator by profession and not a computer programmer, nor did he have any desire to become one. He then said that what he

really wanted was a “program to create programs”; an application that would let him build macro programs by interacting with *it*, instead of with a programming language.

I was stunned. This man had hit the nail on the head with his statement, for it was exactly what I intended to explore in my paper and present at the Interex conference: simply put, the need for software companies to provide their users with easier, more graphical ways of extending and customizing their applications. It's getting close to time for the long-awaited automation promise of CASE to be delivered unto the customer.

3.0 The State of CASE Today

Most people think of a CASE tool as a graphical tool for diagramming software systems. In reality, CASE (Computer-Aided Software Engineering) means any computerized tool that removes a manual step for the programmer or aids in the software development process. Just as CAD is the use of a computer to produce engineering design and drawings, CASE is *design automation* applied to software engineering [1].

Most CASE tools today support one or two structured analysis and design notations such as Yourdon/DeMarco or Gane/Sarson, Jackson data structure charts and the like. They have gained favor in many companies, although some early CASE experimenters have tossed the products out entirely due to limitations and too-high expectations. A CASE implementation of structured analysis and design techniques does not readily lend itself to describing interactive graphics systems, for example [2]. Structured analysis and design techniques are mostly data and transaction-oriented, and only the CASE tools that support state-transition diagrams help out with the aforementioned graphics system design. CASE, although in widespread use, is still regarded as a mixture of art and science by many, not quite a true engineering discipline yet [1].

CASE has struggled to gain respect as a productivity-enhancer, if not a “silver bullet”. It has had some success with certain areas of software development, enough that it makes me believe that CASE concepts can and should be applied to APIs (Application Programming Interfaces) in commercial software.

The attraction of CASE is two-fold: first, software engineers can use graphical notation to describe a complex system, partitioning it into small “chunks” to make it easier to understand and deal with. Second, good CASE tools have repositories where the diagrams and the data contained in them are stored, allowing the diagrams to be easily extended or modified. These CASE tools may also support the generation of some if not all of the code directly from the diagrams.

CASE is trying to take the monotonous, repetitive detail of programming away from software engineers so they can be free to concentrate on bigger issues such as analysis and design. Unfortunately, not much is being done along these lines for the end user. The end user is in an even bigger quandary than the detail-oriented programmer: not only do they have to extend and customize their systems by programming, they are not professional programmers in the first place! Writing code is simply a means to an end for these people. They are experts in their own domains such as engineering, accounting, etc. but most have little interest in becoming programmers.

4.0 How Should Users Talk to Computers?

If CASE as it exists today cannot help end users, what then? Much has been made of the “conversational model” of human-computer interaction. What end users really want, some experts say, is to be able to speak to the computer in an informal, natural-conversation-like language, rather than learning formal programming languages like C or Lisp. This is understandable. Most computer science-types spend a long time becoming adept at a language as complex as C, and they have a vested interest in learning it: they want to become computer programmers!

Nardi [3] states that there is a growing camp of experts who believe informal conversation will never work with a computer because it is too imprecise, too dependent on context for its transfer of knowledge. She feels that the argument against formal languages is weak, and cites the use of formal notation such as baseball scoring, knitting and even rules for ordinary games we all play such as football, golf, or board games.

People have no real difficulty in assimilating game rules; nor does an uneducated, untrained person who sews have a problem modifying a stitch pattern to knit a dress [3]. The reason we can do things like this is because we are interested in them! Accountants (as a rule) are not interested in programming; therefore, they may have a difficult time mastering (or in some cases even attempting) programming. Whether or not the language is formal or informal is not the real issue: it is whether the language is understandable by the user, and carries some meaning with regards to what the user is interested in.

Many cognitive psychologists believe that a person’s ability to assimilate new knowledge is directly impacted by their familiarity with the surroundings of the material [3]. In other words, professional programmers have an easier time learning languages because that is their chosen field. Engineers have a tougher time with C or C++ because they don’t think in terms of functions and keywords, they think in terms of engineering concepts.

CAD systems have been attacking this problem in one fashion or another for a dozen years or more. There are many proprietary macro languages in use by CAD vendors, but most are very quirky in nature and nearly as complex as the general languages like C. What is really needed are language standards for professional disciplines. James Martin remarked as long ago as 1967 that new languages should be developed that allow users to speak in the language of their chosen profession. Engineers, accountants, statisticians, mathematicians, would all learn the computer language of their professional calling as they learned that profession [3].

Of course, this implies international standardization of a magnitude the world has not yet seen. For those of you who followed the struggles of the Open Software Foundation (OSF) and/or the current struggles of STEP, it may seem like a good time to reach for a cold one, sit back and watch the fur fly!

5.0 Graphical Programming

There are many products on the market or under development today that support graphical programming. M.I.T.'s OVAL project was a prototype of a graphical development environment that facilitated object-oriented programming graphically, via Objects, Views, Agents and Links, hence the acronym "OVAL".

4GL products allowing graphical programming of relational database applications remove the need for programmers to write complex query statements and generate user forms automatically. These products supply GUI front ends and allows users to interact with forms and dialogs to create their applications. Users never have to write a single line of code in most cases.

HP's SoftBench is a C++ development environment that moves developers away from coding and closer to specifying applications graphically. Class browsers, forms and other tools let developers start with a solid framework of compiled and linked code for their project, including hooks to common utility functions like file selection mechanisms. Products like SoftBench are beginning to remove a lot of the mundane work from application development.

Other visual-oriented development environments have emerged in recent years. Products like Visual Basic and Visual C++ wrap tools and utilities into a GUI to facilitate easier application development. So far the end users of existing applications have not seen much in the way of this type of customization front-end in their products, though.

At the time of this writing I had come across a product called SynchroWorks from Oberon Software Inc. that provided system integration tools for business applica-

tions. SynchroWorks appears to have some facilities for visually customizing applications, presumably geared toward the ordinary user. Unfortunately, most mainstream commercially available software applications have not begun to take this kind of approach to customization by the user yet.

6.0 Object Technology

Object technology may facilitate end user programming by virtue of its “plug and play” nature. Not in the form of spreadsheet users writing C++ code, however. Object technology’s primary payback is re-use. To facilitate re-use, applications have to be broken up into smaller, more manageable pieces that are decoupled from the rest of the system as much as possible. With these high-level building blocks and some graphical tools such as icon palettes users could assemble and re-assemble functionality to create their own customized versions of an application [7].

Object technology is a perfect proving ground for the graphical programming concept because objects are a more natural, life-like way of representing things. Objects lend themselves to graphical representation quite well. But many OO “bandwagoneers” feel that if something isn’t OO, it just isn’t good enough. Dan Ingalls writes about how computers are supposed to help people, whether they utilize object technology or not. His vision of the “big picture” of computers helping users reads like a recipe for end user programming:

- We start by building general and specific computing components.
- Then we assemble those components to make useful applications.
- Later we may want to improve the components or change the overall structure.

Ingalls’ work in what he refers to as *instance-based programming* is an example of what is needed for the end user to *really* be empowered (apologies to OSF/Motif). It involves users creating objects (or maybe picking them from an icon palette?), and connecting them together, basically by specifying their static and dynamic relationships [8].

Ingalls feels that designing components and assembling them are complementary functions in computing, the former requiring analytical behavior and the latter a synthetic activity. Design is usually textual, and assembly is visual [8]. If we assume that this is an accurate assessment, end users would surely be adept at visually assembling the components which were designed and provided by the application manufacturer’s computer science gurus.

7.0 Programming by Example

Monkey see, monkey do. There is an application for this old adage in computer science. *Programming by Demonstration* (PbD) has been a widely-researched field for many years. More sophisticated than simple keystroke logging or macro recording, true programming by demonstration techniques watch what the user does with the computer, not simply to record the keystrokes for playback at a later time, but to capture the intent of the user, in order to create a more generalized program [4]. In a CAD application, for example, instead of capturing the steps involved in creating a 12" radius circular array of symbology, PbD techniques could generate a program that would simply let the user place any symbol in a circular array at any radius.

PbD has been experimented with for many years, and many approaches have been taken. Allen Cypher's excellent treatment of the subject shows how vast the research in this area has been.

Graphical or language-specific code could be generated by having the computer simply observe what the user is doing. This would be similar to keystroke logging only more sophisticated. I like to think of it as *super-logging*. Programming by example could be an adjunct to graphical programming environments and discipline-specific languages.

The drawback to user customization solely through PbD is obvious: you must be able to step through the desired task within the application domain in order for PbD to capture the task and decipher its intent. If the user wants to perform a task the application is incapable of doing (in its current state of functionality), they are out of luck.

8.0 The Desktop Metaphor -- Not Alone Anymore?

Studies have been done showing that the desktop metaphor for computer software may not be the final frontier. It works quite nicely for those of us who work in offices, but sooner or later computers will be coming out of the office and into the field, the shop floor, everywhere. For many of these new application users the office metaphor is an unfamiliar one. A machinist probably could figure out what to do with a folder or a trashcan icon, but wouldn't it be nice if the application's environment paralleled the user's?

The kitchen metaphor has been suggested as a good general-purpose environment because everyone knows what a kitchen is and what kinds of things you find there [6]. The possibilities for other metaphors are as endless as the uses for computers and the application disciplines they serve. The construction foreman, using a lap-

top or notebook computer on a job site, could interact with his/her applications via the "construction" metaphor. Imagine replacing the familiar trashcan icon with a scrap dumpster, or even more intriguing, instead of starting a new document, he/she might "obtain raw materials", utilize tools from a tool crib to mark up drawings with "as-built" information, unroll blueprints (read-only or red-line) to review, etc. These would be the same functions as in the desktop metaphor for the most part, just packaged and named a little differently.

The metaphor used in commercial application environments needs to parallel the user's environment, which in turn should support extension of the product on the user's terms, in their own language.

9.0 End User CASE for Tomorrow

Applying AI techniques to mainstream CASE tools may be done someday. Things like converting analysis models into design models, or generating *all* of the code from a design model could be possible. Tutoring systems could be developed that learned about the user on-the-fly. So-called "Analyst's Helper" tools could be developed that could aid in the assimilation of development methodologies, which typically take one or more years to learn [5]. The future of CASE does indeed look very bright, from the perspective of a software developer. But only a small percentage of software users are software developers and chances are, the customer who just bought an application has no idea what a CASE tool is, why it makes his product better, or how it could be used *by him/her* to increase his/her productivity. As these new concepts are applied to CASE, similar things must be done for end users. If some of CASE technology's flaws are fixed, CASE techniques can then be applied to end user programming

One of the big flaws of early CASE tools (and most of them today) is that they were designed to build systems from scratch. Not much thought has been given to building CASE tools to extend existing systems, except where extensions are not tightly coupled to the rest of the system and can be treated as "black boxes". Even if CASE could support extending systems, an end user would be overwhelmed by the complexity of the various notations and staggered by the price of most tools. This type of customization tool must be supplied by the software application vendor, not a commercial CASE vendor. It must be simple enough so as to not require a degree in computer science, or even more than a passing knowledge of traditional computer programming. Many products exist today with customizable APIs, even macro languages for extension, but this approach would be easier to learn, and certainly more fun to work with!

Simplified end user CASE must emerge as commercial CASE for software developers evolves into the super-tool it has been touted to be.

10.0 The Future

I see a future of endlessly extendable applications. I envision products with graphical APIs that surface some portion of the application functionality in a language the user already speaks, as opposed to the language of a computer scientist. The good news is: this future may be less than ten years away. The highlights of this new breed of application:

- A formal notation that parallels the language of the user's chosen field. A mechanical designer would be able to program his application in terms of machining operations, design features, dimensions and tolerances. An accountant would speak to his computer's internals in a language of balances, spreadsheet figures, etc.
- The formal, discipline-specific language notations will be supplemented or even implemented with a graphical programming environment. Objects representing real-world things in the user's field will be graphically represented. Operations on these objects will be specified with mouse movements to denote links between objects, views of the objects, and agents acting upon those objects. The API functionality surfaced in each application will be wide-open for extension by the end user in this simple, graphical format.
- Since the graphical approach will be based on the underlying formal notation, customer support will not be adversely impacted by the fact that every user will have their own "custom" version of the product. Each customer support call will be preceded by the transfer of the user's "customization environment" and the extensions they've created, allowing the support personnel to quickly and easily match the user's environment.
- Super-logging, or Programming by Demonstration will replace or supplement traditional keystroke logging and macro recording. Users will be able to create a program by executing the steps they wish the program to make. The logging facility will generate a program file in the formal notation to be used as a starting point. This program would be stored in the customization API's repository along with other custom programs. The customization interface will then be able to display it graphically for further modification (if necessary) by the user. Most important, these Programming by Demonstration techniques will create generalized programs as opposed to specific ones, in order to facilitate their application a variety of situations.
- The desktop metaphor may only be one of many environments for applications to exist in. By standardizing other metaphors around the functionality at the core of the desktop, it will be possible to "toggle" an operating system (yes, they'll still be around in this future) from one environment to another, depend-

ing on the perspective of the user or the nature of the application discipline. The metaphor and the notation language must parallel the environment of the user, regardless of field or discipline.

- Finally, if the folks who put together the standard for these formal notations are very, very careful, there will be an underlying commonality between all of them that would facilitate exchange of information and programs across disciplines, where applicable. Say the mechanical engineer is looking for a part table matrix program that sounds quite a lot like his buddy up in Accounting's custom spreadsheet software. This could be dicey, given that the underlying APIs for the two applications might be radically different. At a minimum, the "dumb" diagrams could be brought over (please, let there be a STEP protocol for their format) and "smartened up" interactively by the user.

11.0 Conclusion

This particular future looks very bright for end users. Imagine extending a computer software application as easily as a musician might transpose a song, or a seamstress modify a dress pattern. If you aren't a musician or seamstress, you say, it won't be that easy. But everyone knows how to do something. Users' areas of expertise will be supported by their own unique language that computer applications that serve them understand. Users will know this language well, because it will be the language of their trade or chosen field. They will easily drag and drop graphical objects, linking them together, adding or subtracting data and operations to and from their application(s) to get just the right functionality for a particular need.

And most important, the end user will never have to open a programming manual again.

12.0 Bibliography

1. *The Case for OO CASE*, Nicholas Wybolt, Object Magazine, July-August 1992
2. *Implementing CASE Tools*, Ken Robinson, Proceedings of the 2nd Annual IDE User Group Meeting, March 1989
3. *A Small Matter of Programming*, Bonnie Nardi, MIT Press, 1993
4. *Watch What I Do: Programming by Demonstration*, edited by Allen Cypher, MIT Press, 1993
5. *Object-Oriented CASE*, Daniel Rasmus, Byte Magazine, December 1992
6. *The Kitchen Interface - A Lateral Approach to GUI*, Duncan Langford and Cienwen Jones, SIGCHI Bulletin, April 1994, Volume 26, Number 2
7. *Irresponsible Journalism*, Marie A. Lenzi, Object Magazine, June 1994
8. *Instance-based Programming and OOP as we know it*, Dan Ingalls, Object Magazine, June 1994

**Paper Number 1009
HP SourceReader UX**

**The HP3000 Debugging, Development, & Source
Understanding Tool of the Future**

**Gary Robillard
Hewlett Packard Company
Software Technology Division
Roseville, California
(916)-786-8000**

What is this paper about?

The purpose of this paper is to describe the HPSR (*HP Source Reader UX*) program that is used within HP to access and navigate compiled source code listings. It will give a brief description and history of HPSR and then explain how it is used to increase engineering productivity.

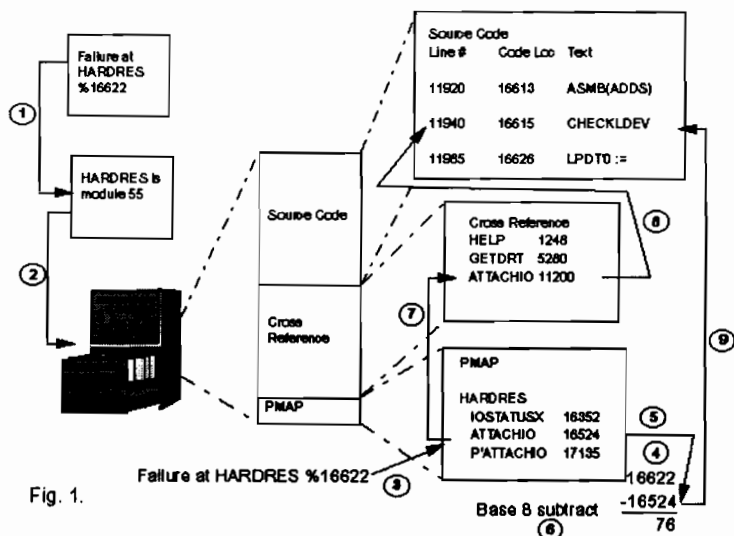
What is HP Sourcereader UX?

HPSR (*HP SourceReader UX*) provides software engineers developing or maintaining MPE software quick, easy access to compiled source code information. HPSR can display and navigate through large volumes of source code in a Unix MOTIF windows environment. Storage and retrieval of this information on a high capacity storage device, such as a CD or hard disk, eliminates the need to print, archive, and manually access listings. In this day of ever increasing environmental awareness it allows us to take another step toward the "paperless" office.

Why HP Sourcereader UX?

Debugging software on MPE platforms during development and maintenance has never been easy in the absence of true source-level symbolic debugging support from the compilers, linkers, and debuggers. As a result, application programmers down to system-level developers need to know details such as stack relative addresses of variables, code offsets for procedures and source lines, machine code for source lines, and line numbers for identifier names. This information comes from manually consulting the compiled listing which contains the source listing as formatted by the compiler, the identifier cross-reference section, and the segmenter/linker section. Often times this information is generated by different tools and presented in an inconsistent format.

Anyone who has debugged MPE programs knows how cumbersome this process can be. To find the source code for a procedure in a stack trace, one must consult their listing library and find out which compile unit the desired procedure resides in. The cross reference is then consulted from the proper listing to find out which page the procedure is on (*the linker map may have to be consulted to confirm this procedure resides in this compile unit*). Once the procedure is located, the code offset from the debugger or dump analysis tool stack trace must be found in the statement map. One then locates the correct statement number in the listing. This cycle repeats for each procedure in the stack trace. Figure 1 shows the complex manual process that must be used to accomplish all of the above.



HPSR facilitates quick and efficient debugging of HP 3000 Computer systems by allowing the user to display source code at any point within a specified procedure, segment or module. The user can then quickly scroll the display or jump to any other location. Relevant information can be displayed on the screen within seconds, including identifier definitions, reference materials, and for MPE V, the machine instructions corresponding to the displayed source lines. The program also has many useful auxiliary functions including searching, printing and displaying procedure caller/callee information.

The history of HP SourceReader

In 1986, the Commercial Systems Division (CSY) of Hewlett Packard began to look for a way to reduce the amount of time that engineers were spending in overhead activities (*things like walking to the library to find a source listing, finding the right place in the listing, and then finally being able to start working on the problem, only to find you needed another listing of a different module, etc.*). When the HPSR application was being developed, the following goals were established:

1. Elimination of paper listings to save time, space and mundane labor.
2. Use of emerging technology to make engineers' time as productive as possible.
3. Ease of use to minimize learning time and errors.
4. Minimal impact upon organizations to supply source code in order to avoid the need to reformat source code or modify procedures.
5. Cost-effectiveness to make it easy for support organizations to justify the required expenses (*such as for the CD-ROM readers and the subscription service that supplies the CDs with the MPE source code*)

All of these goals were met, and HPSR is being used in all the Response Centers and Expert Centers worldwide.

Initially, HPSR was developed on a PC based platform, but as the working environment at HP changes from PCs to UN*X based workstations, the need for a UN*X based version of HPSR was seen. At SWT in Roseville, Calif. a project team was formed to implement the UN*X based HPSR. It was decided to use a motif based interface since there are many advantages to using a UN*X workstation over the PC. For example, many engineers can share the same CD-ROM drive or hard disc, as well as the application being a true X-Window application.

HP Source Reader and CDs

HP Source Reader actually consist of two main parts. The first part is the data preparation system, which is used to generate the CD-ROMs from the compiled source code. The second part is the program that accesses the data on the CD-ROM. The original PC based program is still available, as well as the newer UN*X version. The data preparation process has not changed much and the structure of the CD-ROM is

identical for both the PC and the UN*X versions of HPSR. This means that the same CD can be used for either version.

For MPE V and MPE iX, a new CD-ROM is generated each time a new version of the operating system is released. Each CD-ROM has a capacity of more than 700MB of data. That is the equivalent of a 35 foot high stack of paper listings! Figure 2 shows how the manual process shown in figure 1 is done using the HPSR access program and a CD-ROM containing the source code listings.

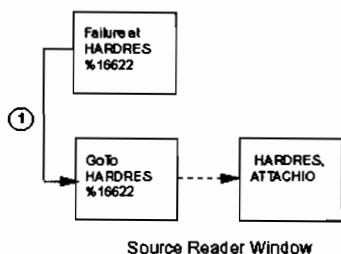


Fig. 2.

The only difference between an audio CD and a data CD-ROM is the meaning of the information that is recorded on the optical media. An audio CD has digitized music, while a CD-ROM has computer data. The data on a CD is recorded as a series of "pits" positioned in a continuous spiral beginning at the center of the disc. The pits are read as ones and zeros when illuminated by a laser source.

The CD-ROMs for HPSR have a DOS directory structure utilizing the ISO 9660 format. The directories are set up so that each program listing on the CD is under a directory for that library (for MPE V the only type of library is the SL, for MPE iX there is an NL, SL, and XI library directory). There are also documentation subdirectories which can contain manuals, documents, etc. The program and library directories contain compressed source code, cross reference, identifier maps, procedure maps, and optionally the object code that goes with the listing. The document directories contain a set of files including the compressed text, a page list, a table of contents and an index for each document

Getting the Source Listings on the CDs

In order to get the compiler listings into the proper format to be placed on the CD-ROM, a program called a filter is used to reformat the listing into the necessary files. The files on the CD-ROM are designed and organized to facilitate rapid retrieval of the desired information. When HPSR was first developed the filter programs were being designed to run on a PC. The main problem encountered was the lack of speed and disc capacity that was available at the time. The 286 was still state of the art! Certain large programs, such as the INITIAL program that is used to start MPE V would take more than 24 hours to download the 80,000 + line listing, filter it and upload the filtered files back to an HP3000. It was then decided to rewrite them to run on the HP3000. At that time they were written in PASCAL/V, and the filters were run on a Series 70. The 24 hours dropped to less than 3 hours! Finally, the programs were ported to PASCAL/iX and the filtering of a large module is now a matter of minutes instead of hours.

There is a different filter program for each language that a program is written in. Since the tool was initially designed for MPE operating system support personnel, the languages that currently have filtering programs are PASCAL/V, SPL/V, MODCAL/V, PASCAL/iX, and C/iX. The filters were designed to use a three pass philosophy. The first pass parses each record of the compiler listing and determines which part of the listing it represents. The information to be retained is then reformatted and written to intermediate files. The second pass performs certain cleanup tasks such as removing duplicate identifier information. The third and final pass generates the files that will be copied to the CD-ROM.

Once all the files are filtered and ready to be placed on the CD-ROM, they are pre-mastered to copy them to a tape in what is known as High-Sierra format. The files output by the filters are standard DOS file images, and the CD-ROMs are recorded according to the High-Sierra standard. The program generates a tape that is compatible with the optical company that does the mastering and duplication of the CD-ROMs. We always "burn" at least one W.O.R.M. (*Write Once Read Many*) disc to insure that the pre-mastering and filtering is successful.

It should also be mentioned at this point that once a program or module is filtered, the files can also be copied to a hard disc on either the PC or the UN*X system. At SWT in Roseville we take advantage of this feature to use HPSR as a real time development/debugging tool. As an engineer completes a fix, the compile job will automatically create an HPSR listing file. Then an MPE iX command file is used to automatically process the listing file through the filters and even to copy it to a directory on our UN*X server so that it is available for immediate use during testing/debugging, etc.

Accessing the CD-ROM source listings

A major goal of the design team for the original PC version was to make the access program easy to use. This is important because the engineers using the program may not be familiar with the internal workings of the platform on which the program is running. The project team for the UN*X version of HPSR had a goal to implement all "core features" from the PC access program as well as to add new functionality. Of the "core features" in the PC program, the only one that has not been implemented yet is the displaying of reference documentation.

The following were the major design objectives for the first version of HPSR/UX:

1. Include all "core features" required to use the product as a replacement for the PC HPSR. The only exception, as mentioned previously is the ability to display reference documents.

Additional functionality that was previously unavailable in the PC access program are:

- a. Caller/Callee procedure display
 - b. Support of "disk sets", made up of multiple disks (*either CD-ROM or hard discs, or any combination*), including support for configurable search orders within each disc set.
 - c. Save/Load environment feature to save all source display windows in an environment file that can be used later to restore (*load*) the current environment.
 - d. Support the display of line numbers in the source window
2. Utilize new features/functionality in a UN*X/Motif environment that were not available in the DOS based PC access program.
 3. Deliver acceptable average response times for all operations except those that perform long sequential searches.
 4. Provide a user interface that is intuitive enough so the product may be used by a user (*whether familiar with the PC version or not*) without the need to reference any manuals.

Focus on HPSR/UX

The remainder of this paper will focus on how the UN*X based HPSR access program is being used at HP. At HP SWT in Roseville, California we do what is known as CPE or Current Product Engineering work on products that run on the MPE V and MPE iX platforms. CPE consists of fixing bugs as well as designing and implementing product enhancements. HPSR helps our engineers to be more productive, as well as allowing us to quickly learn new products.

HPSR Command Overview

When HPSR is first invoked, it first checks a file called the pathset file to determine what directories to search, then a license check is performed. The license check insures that the UN*X workstation that is executing the HPSR program has been properly licensed to do so. If the license check is unsuccessful, a message is displayed informing the user that the license check failed and the program terminates.

Before giving an example of how HPSR is used, a brief overview of the windows/commands is presented.

The MAIN HPSR Window

The main HPSR window controls the HPSR application. There are 3 possible choices of pull down menu boxes in the main window.

FILE

Allows the user access to the three file related subcommands that are available in the main window.

GOTO

Allows the user to display source windows. Subcommands allow the user to select whether the source to be displayed is selected by a procedure/function name, a module/segment name or an application/program name.

(In MPE, each module is located in either a library or an application program. In MPE V and MPE iX compatibility mode, procedures are grouped into segments. In MPE iX native mode, segmentation is not used. In order to provide a consistent user interface, HPSR/UX defines "native mode segment" to be interchangeable with "module".)

When one of the subcommands of the Goto menu are selected a new menu will be displayed. This menu will allow the user to enter the name (*with wildcards allowed*) of the procedure/module/Application the user wants to display. The first thing the user needs to do after entering the name is to click on the Find button. Then the HPSR application will display each match it finds. The right hand scroll bar (*or the up/down arrows on the keyboard*) can be used to scroll through the list. The highlighted entry is the one for which the source will be displayed when either the Goto Entry or Goto Header buttons are clicked on. If Goto Procedure/Function is used, then Goto header will open the source window at the procedure declaration, Goto entry will attempt to open the source window at the first executable line of code of the procedure.

SETTINGS

Allows the user to define a new pathset file or to enter the command that is to be used for printing.

HPSR Source Windows

The HPSR source windows display the source code that the user has requested. While in the source display window double clicking the left mouse button on an identifier will open a new source window displaying the source for that procedure, if the identifier is not a procedure then a GOTO procedure window will be displayed. Double clicking the right mouse button on an identifier will display a show identifier window to display the declaration line for the requested identifier (*if the identifier is a procedure then the procedure declaration or an external or forward declaration may be displayed*).

The menus available in this screen are:

FILE - Use these menu items to Close or hide this window. Hiding the window iconizes it. You may also select to show the main HPSR window or any additional info windows, which will raise the selected window(s) to be the displayed window.

GOTO - Same as the Goto in the Main HPSR window

SHOW - Use this menu to show additional information about the source code in smaller 'additional information' windows. These windows are not shown in the MAIN HPSR window.

The additional information windows are:

SHOW IDENTIFIER - Displays the definition of an identifier. The identifier map entry information is displayed at the top of the window.

SHOW LOCAL VARS - Displays all local variables for the procedure the

cursor is located in, or if the cursor is not in a procedure, displays the global variables.

SHOW CALLS TO - Displays any procedures/functions which make calls to the requested procedure/function.

SHOW CALLS WITHIN-Displays procedures/functions that are called from within the requested procedure/function.

SHOW PMAP - Displays the Procedure Map (*PMAP*) for the current module/segment or application/program. The procedure/entry name is followed by STT number, the code offset, the entry offset and if applicable the segment number. Not all of these fields are applicable to all languages.

PRINT - Use this menu to print the selected (*highlighted*) text. Text is highlighted by holding the left mouse button and dragging the mouse up or down.

DISPLAY - This menu selects how the displayed data will be formatted.

The display options are:

ABSOLUTE - Display the code offsets from the beginning of the segment (*if available*)

RELATIVE - Display the code offsets from the beginning of the procedure (*each procedure will show it's code offsets starting from 0*)

INNERLIST - Display the machine instructions for each source statement. (*Only available for MPE V or MPE iX Compatibility Mode*)

INCLUDES - Display shared include files.

IGNORE CASE- Do case insensitive searches.

An example using HPSR/UX and an MPE V stack trace

Given an MPE V stack trace from a memory dump of an MPE V system failure 206, an engineer would need to look at the actual source code to try and determine why the system failed.

The stack trace is as follows:

MPE VERSION: HP32033G.23.00. (BASE G.23.00).

DUMP TAKEN: 6/23/92, 5:13:25.7 PM

```

***** HALT 17
****SYSTEM FAILURE #206 ; STATUS %140030; DELTA P %016622
-f cpcb,stack
FORMATTED STACK. DST 417
***PXGLOBAL***
007 117230: 001650 016600 177777 000003 000002 000000
007 117236: 016000 000000 000152 000152 000444 000354

SEG REL DL: 001650 SEG REL DB: 016600 JMAT INDEX: 000003 JPCNT INDEX:
000002
JOB IP LDN: 000152 JOB OP LDN: 000152 JDT DST INDX: 000444 JIT DST INDX:
000354
JOB TYPE: SESSION DUP: YES INTERACT: YES JCUT INDEX: 000000

* CURRENT PROCESS *

BANK ADDR. X DELTA'P STATUS DELTA'Q SEGMENT PROCEDURE
OFFSET
007 143421: 000004 056622 140030 000015 HARDRES (155)
007 143404: 000004 047452 140026 000021 DIRC (153)
007 143363: 000014 041117 140026 000013 DIRC (153)
007 143350: 000014 040616 140426 000013 DIRC (153)
007 143335: 000014 045653 140123 000301 CILISTF (266)
007 143034: 000671 044744 143037 000602 CIINIT (171)
007 142232: 000202 040015 141037 000004 CIINIT (171)
007 142226: 000001 011770 060473 002163 USER SEGMENT
007 140043: 000000 002571 061001 000007 USER SEGMENT
007 140034: 000000 040000 140050 000004 MORGUE'ABORT (203)

```

The engineer is trying to locate the source line that aborted the system. From the memory dump the engineer has determined that the code aborted in segment HARDRES at offset %16622. This is determined from the stack marker trace of the process that was running when the system failed. Segment HARDRES offset %16622 is the last procedure that executed before the system failure occurred. Note that the %56622 DELTA'P is converted to %16622 because bit 1 of the DELTA'P is the logical/physical mapping bit in MPE V.

To use HPSR to look at the HARDRES source code, the following GOTO screen would be used:

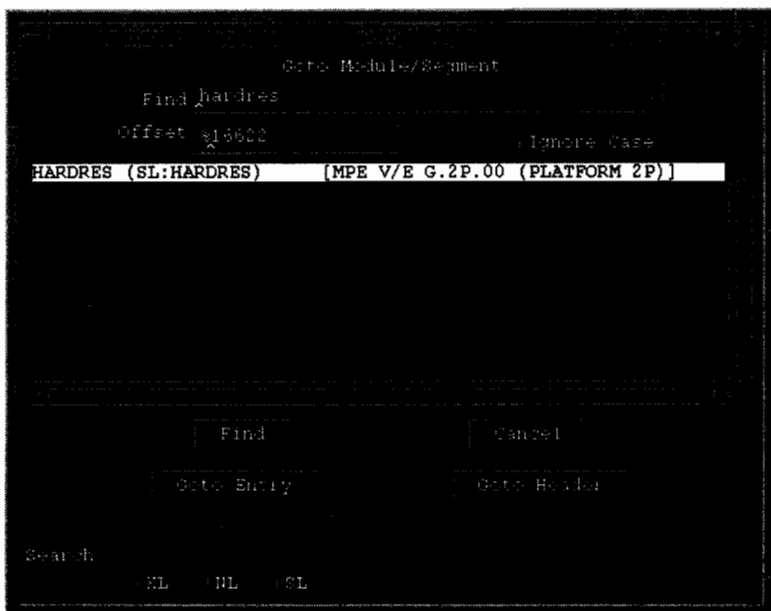


Figure 3a.

Figure 3a shows the HPSR goto screen that is used to display a source window at Segment HARDRES offset %16622. HPSR locates the HARDRES segment in the system SL directory on the CD-ROM, and displays the source code starting at the requested offset. The source window that HPSR displays is shown in figure 3b. The cursor is positioned on the source line corresponding to the return point from the SUDDENDEATH procedure - the engineer has found the call that aborted the system.

The window that displays the HARDRES source looks like this:



Figure 3b.

From the code, it is apparent to the engineer that SUDDEDEATH is called if CHECKLDEV returns a CCL (*Condition Code Less than 0*) result. In order to determine what the value of LDEV that was passed to CHECKLDEV, the engineer needs to know the memory location of the variable LDEV. With HPSR the engineer simply double clicks the right mouse button on the identifier LDEV and HPSR locates the identifier map information for LDEV and displays it in an additional information window as shown in figure 3c.

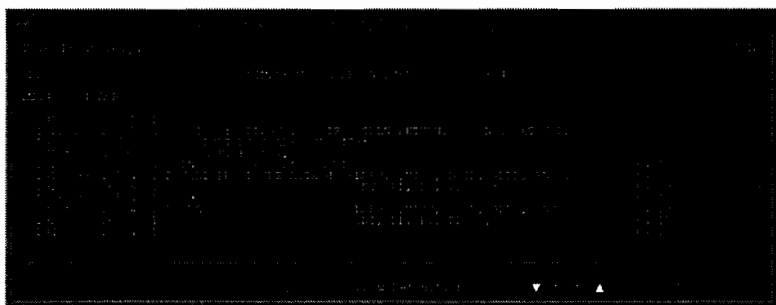


Figure 3c.

Looking at the definition the engineer now knows that it is found at location Q-%14. The engineer can then examine the contents of that memory location in the memory dump and determine why the system failed.

If necessary, since this is MPE V, the engineer could also have chosen to change the format of the display to include the innerlist of machine instructions for each line of source code. This would also indicate that LDEV is at Q-%14.



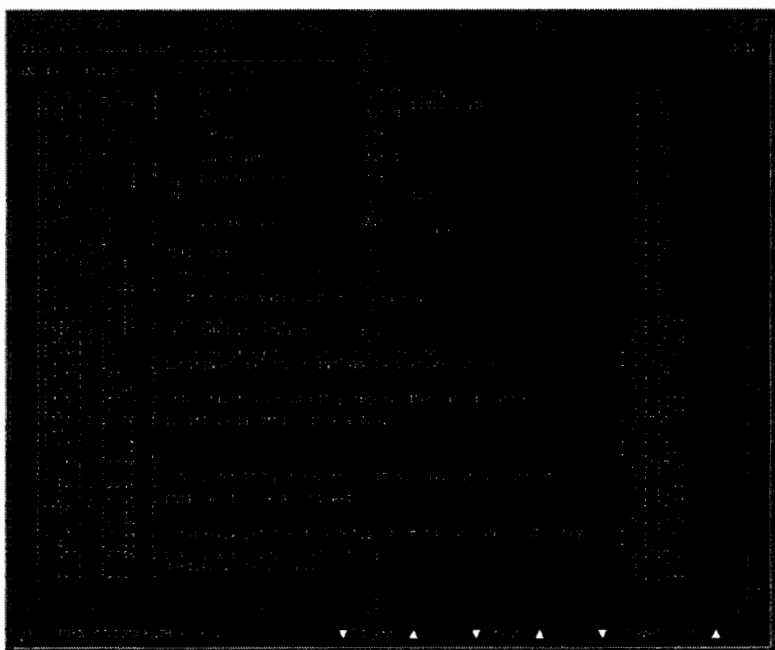


Figure 4a.

Figure 4a shows the display options menu that is used to select how the displayed source will be formatted. In this case, the engineer is requesting an innerlist of the machine instructions that correspond to each line of source code.

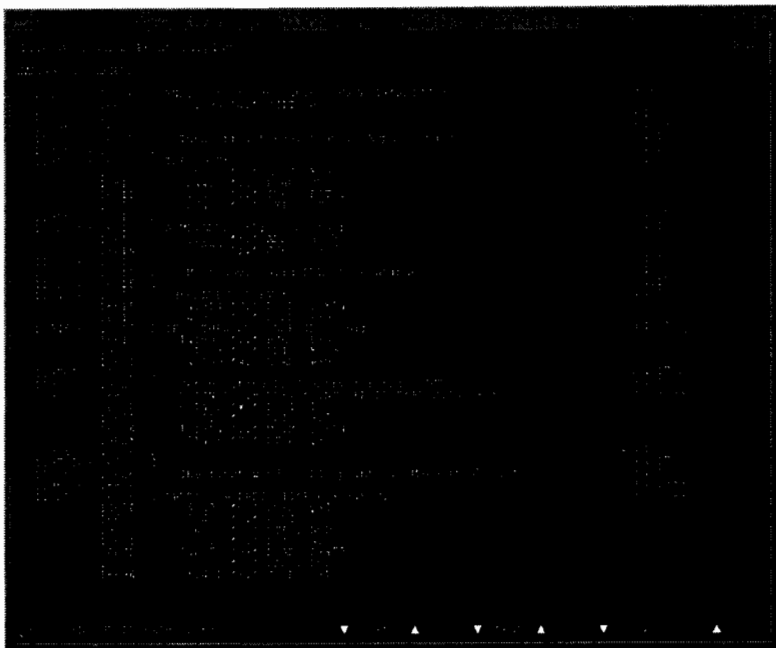


Figure 4b.

Figure 4b shows an MPE V source window with the innerlist display option enabled.

HPSR as a source navigation tool

Learning the internal workings of a new software program, particularly when you will be responsible for maintaining (*fixing bugs, implementing enhancements, etc.*) can be time consuming. Particularly in the absence of Internal and/or External Maintenance Specifications. HPSR reduces the time it takes to learn how a piece of software works. Normally, when reading a paper listing, you will find a line of code that makes a procedure call. You then have to locate where in the listing that procedure is, look at the procedure, and then go back to where in the code you were before the procedure call. With HPSR, you simply double click on the procedure name, and regardless of where the procedure resides (*as long as the source for the procedure is known to HPSR*) a new source window is opened with the new procedures source listing in it. You can then peruse the new procedure, perhaps making notes in a documentation file, then close the window and go back to where you were in the previous window. The ability to simply double click on identifiers and record definitions vastly improves the speed at which unfamiliar source code can be navigated. It is particularly helpful when looking at complex record structures that may have several layers of definitions before you actually get to the piece of information you are looking for. I have had times with a paper listing

where I have had to look in the crossref listing so many times that by the time I found what I was looking for, I forgot why I was even looking for it, and had to go back to square one....

As mentioned before, one of the first things we do here at HP SWT is to filter any new software that comes into our division. This way, even if the engineer with primary CPE responsibility is not available, another engineer can access the code quickly, and is more easily able to work on a problem that comes in.

Conclusion

HPSR provides substantial increases in productivity. The UN*X version of HPSR provides even more convenience and ease of use over the PC version since multiple windows can be displayed on one screen, and information for documenting problem analysis and source navigation can be cut and pasted from the source window into an editor window or a document. HPSR is in use in virtually every HP Response Center and Expert Center worldwide! There are also several software development labs (*including SWT in Roseville and in Mountain View*) that are using HPSR as the tool of choice for debugging and developing code.

-

The Client/Server Headache: Performance How to Get Relief

by Dan Gillis
Manager Advanced Technologies
Dynamic Information Systems Corporation
5733 Central Avenue
Boulder, CO 80301
(303) 444-4000
Paper Number 1010

The Client/Server Headache: Performance

How to get Relief

Introduction

Like all new technologies, client/server is progressing through its "life cycle" right on schedule. First heralded in the early 1990s as a solution to all end user data access needs, it has since begun to mature as it is actually implemented at organizations worldwide. With that maturation, as in life, brings a growing awareness of both its strengths, and its potential weaknesses as a data access solution.

In my work as a consultant to organizations looking to implement technology solutions to improve productivity, the single biggest weakness I've encountered with client/server is performance. Companies invest tremendous resources to create new avenues of data access--attractive PC screens, point-and-click GUIs, intricate network connections--only to find this new-found access is often frustratingly slow.

This paper will briefly discuss the reasons that client/server performance becomes an issue in operations, and will outline the various strategies I've found to be most helpful. I've included three case studies to illustrate how these solutions worked in real-world situations.

Great Client/Server Design! What Could go Wrong?

When we first began to understand the concepts behind client/server, the possibilities for its application seemed endless. The executive suite could finally have those summary numbers--in a colorful chart no less--any time they wanted to see them. Our sales reps could have instant access to product availability and pricing information--maybe even from the road! Our customer service people could have access to any corporate information they needed to solve a customer problem. Marketing could have any report they wanted--without having to go through MIS every time they wanted a new set of numbers.

But as we've begun to implement these systems, our expectations have inevitably dropped. The front end screens were well designed, and certainly made navigating the computer system and report preparation easy. And even with the endless stream of three letter LANs and communication protocols, we could set up a pretty straightforward connection to get all these PCs talking to the host systems. But it doesn't seem to work the way we envisioned it.

The very freedom of access that we've given our users means they can kick off horrendous queries that slow everyone on the system. The problem only multiplies with more and more users. This problem is so pervasive, that I've dubbed it the "Client/Server Exercise Program." It works as follows:

Client/Server Exercise Program

<u>Action</u>	<u>Reaction</u>
1. Click on a button	Get coffee
2. Do some work	
3. Click another button in case the first really wasn't working	Get coffee
4. Do some more work	
5. Click it again	Call Home
6. Start in on your in-basket	
7. Mumble about the PC and click it again	Bathroom!

Richard Simmons couldn't have designed a more strenuous physical exercise program. Every time you click a button, you get up and exercise.

Same Old Problem with a New Name: I/O Bottleneck

What's going on? Analysts have called it network traffic overload. And data transfer lag time. Even disk contention. But it boils down to a term we all discarded in the 80s: I/O bottleneck. Processes will only run as fast as the slowest portion of your computing process.

Let's take a look for a moment at how database retrieval systems work.

Data is typically stored in a database or file structure. This structure defines the relationships between all the data entered, and allows access to that data by a variety of methods. The speed of retrieving that data is most heavily influenced by where that data is stored.

If the data being searched is located in the systems *main memory*, retrieval will be very efficient. All the computing resources are spent on locating the data.

As we all know, computer systems are forced to switch data back and forth between main memory and disk to manage efficiently. If, however, the system has *cached* the data you're looking for into "temporary storage," this retrieval method is reasonably fast.

Next in line for retrieval speed would be data that's stored in *network memory*.

But, most often, we're looking for data that is stored in *disk*. 32 I/O's per second is the current physical limit of disk transfer speeds. When searching through large files, this physical transfer speed limit presents a real bottleneck for performance. While the latest advances in RISC chip design have given us better relative I/O by using pre-fetching, on-line retrievals are often hampered by the sheer size of the database that must be searched to find the specific records of interest.

These physical searching limitations are the culprits to our client/server designs gone awry. The most beautiful designs won't overcome slow system response to queries. So what are some options?

Performance Improvement Options

There are a variety of options that have been tried over the past several years. They range from what I call "limitation" solutions to hardware and software options.

Hardware Solutions

The one solution that most companies choose first is bigger hardware. Faster network, faster drives, more memory to cache and easily justified. But it's one of the most costly solutions. And may not speed up the kind of ad hoc queries you're trying to provide.

Hardware solutions range from:

- SCSI drives
- 100 Mb network
- Faster CPU
- More memory
- Fiber optics
- More cache memory

Limitation Solutions

If purchasing additional hardware is too extravagant, many companies look to limit the activities that impact performance the most severely. I'll list the most common

limitations that I've found:

- Break down large files into multiple smaller files to speed searches
- Filtering or restricting access to corporate data during peak operating hours; running against the larger files in batch
- Activity timer aborts query after set length of time
- Pre-selections done in batch at night for most popular queries
- Roll-up summaries prepared in batch for EIS systems and accounting
- Transaction log kept on PC during peak hours, uploaded to server for batch updating in off-hours

Unfortunately, these limitations, intended to make the system work more smoothly, often have the effect of frustrating the very end user the system has been designed to support. I've even seen the recommendation by a very respected client/server vendor, that no more than 20 - 30 users should be hooked up to client/server systems at any one time. Why? Not because that vendor's inherent performance is so slow, but because of the physical limitations of processing ad hoc queries and passing the results down to the client.

There are better ways.

Software Solutions

I have an innate predisposition to finding software solutions to performance problems. There are several available that I'd like to briefly cover. Some, I think are better than others for client/server computing. But you'll need to evaluate your own organization environment to select the appropriate mix for your individual situation.

Data Warehousing is an increasingly popular alternative to companies with large amounts of data to process and provide access. A Data Warehouse is basically a revamped historical database with a new name. By off-loading your on-line data periodically into a separate data structure and providing indexed access, you can leave the detail in, and roll up the summarized data if necessary, without negatively impacting the performance of your production systems.

But the Data Warehouse has several drawbacks for client/server performance you should be aware of:

- The data is not on-line
- The warehouse can be complex to set up and manage
- Without sophisticated indexing, the warehouse will not significantly improve retrieval performance
- Updating and indexing must be managed

More basic still is the selection of the *Database System*. Database system vendors spend millions each year telling us that their particular solution is the fastest performing platform. Benchmark after benchmark attest to each RDBMS's performance rating. To do so, most relational database systems rely on some form of query optimization, and in some cases, SMP and MPP processing.

Query optimization, also called pre-optimization, is essentially a search optimizer. This optimizer can evaluate a query before it is processed to schedule the search, sort and merge steps in the most efficient manner. In a query with multiple selection criteria, the optimizer will attempt to use the table with the smallest number of entries to search first to narrow the search to a smaller subset before reading other files for the appropriate "matches."

While pre-optimization is a giant leap forward, it can be of little help in queries where the qualifications are coming from multiple tables with a large number of entries. Similarly, multiple table joins are still uncomfortably slow.

The latest releases of Oracle, Sybase and Informix have some level of support for parallelizing queries. These can give you substantial performance gain, but it's not for the faint of heart. Most Symmetrical Multiprocessing or Parallel Processing Machines are hundreds of thousands of dollars, and can require months of tuning to give you the performance gains you need. While the database vendors have been wise to support these machine types, it's really nothing more than throwing more, smarter hardware at the problem.

My personal "favorite" solution is *Inverted File Indexing*. Inverted file indexing is the "dark horse" candidate for improving query performance. While more complex to initially implement, I've found that it can be far more important than the database or file structure you use in determining client/server performance.

Inverted file indexes offer three strategic advantages in client/server performance:

- Dramatically improves server performance for multiple selection criteria and joined retrievals
- Offers a count of the number of qualifying records *before* the retrieval is initiated and downloaded to the client
- Ability to implement summary indexes to speed ad hoc summary reporting and analysis

Inverted file indexes allow you to instantly determine the number of records that qualify in a query--a query that can have multiple selection criteria, even across

multiple tables. I've seen systems speeded up by a factor of 10 or more with the addition of these indexes. This performance improvement is even more critical in client/server applications.

I'll cover inverted file indexing in a bit more detail at the summary of this presentation in the case studies we'll discuss.

Database Design

The final area of performance tuning I'd like to cover is the basics of database design. I sometimes like to call this section the "Database Blues," because so many of us have tried some great new design technique, only to find that it ruins performance.

Normalization is one such example. Normalization can provide a number of benefits, but increasing performance is not one of them. One general rule of thumb I give to clients is to normalize in stages and test after each one to make sure that your performance is degrading beyond tolerable limits. I've yet to see a high-performance, functional database normalized over 4 levels.

Table joins is a key area that impacts retrieval performance, if you're using those joins to direct the path of the query. It will come as no surprise that the more joins you have in a query, the slower it will be. The dilemma for client/server designers is that these very joins are the ones the users need most: "How many products were sold last month to customers in the Northeast that were priced at the promotional discount?" A perfectly reasonable business question. A nightmare for the DBA who sees the "spaghetti-joins" necessary to select that data from the 3 - 4 different tables.

One other database design point I'd like to bring up is SQL. The Standard Query Language often isn't. The SQL implemented by one database vendor is usually not portable to another file structure or database. If you're evaluating different databases for performance gains and are sold on their platform independence, talk to some of their customer sites to see if that is indeed true. I've found that most often, it's not.

Client/Server Case Studies

I'd like to share a few personal experiences in implementing client/server solutions, in hopes that you can glean some "real life" options for your own situation. I've selected three clients that I've worked with over the past year:

Boots Pharmaceutical
Betz Industrial
Midmark

EIS application
MSDS Documentation application
Automated sales force application

Boots Pharmaceutical

Boots Pharmaceutical is a pharmaceutical development and manufacturing company headquartered in the Chicago area with offices in the UK. They began looking at client/server as a way to give faster, more flexible access to daily and weekly sales data to management so inventories could be managed more effectively. In-store sales data was being collected daily across the country, but there was no effective way to summarize that data and make it quickly available to the marketing and finance people in headquarters.

Their first efforts to install a client/server system pointed up the need for a way to process the large volume of data quickly, and allow managers to drill down for any information they wanted to pursue. They chose to add inverted file indexing--OMNIDEX--to their corporate database. They created Visual Basic™ screens for the managers to use to point-and-click to the information they wanted to see. These Visual Basic screens used a client/server API that called the OMNIDEX indexes on the server to first provide a qualification count. Any time the qualification was modified, a new qualification count was immediately sent to the PC screen. Once the user was satisfied that they'd identified the records they wanted to see, Boots used specialized summary indexes to quickly summarize the large amount of data. The program then downloaded the totals into the client program where they could be further summarized by view, graphed or put into a report.

The stunning statistic that I like to quote when I talk to other clients, is that this whole project took less than three weeks to prototype and initially implement. Boots has since been refining the system, adding new query screens and update mechanisms, but they had a working client/server system that performed in seconds off of millions of records in less than three weeks.

Betz Industries

Betz is a worldwide chemical manufacturer. Anyone in the chemical or related industry knows that these companies must maintain copious records of environmental data on any chemical they produce or handle. Mandated as MSDS, or Material Safety Data Sheet, a chemical company such as Betz can have literally thousands of documents on the various chemicals they handle.

Once produced, these MSDS documents are the primary reference used by both the company and the EPA to identify chemicals with similar properties. For example, if a chemical spill occurs, the EPA may need to quickly identify the contents. Descriptions of the spill are cross-matched against the MSDS records to look for candidate chemicals that could be involved. Betz wanted a more efficient way to access the volume of records and to have the appropriate information immediately available to the user.

Betz set up a client/server system, again using Visual Basic screens and the OMNIDEX inverted files, that allowed them to use full keyword retrieval and multiple key searches on their multi-million record database. But speed of retrieval wasn't their only concern. Once the appropriate data was located, the amount of information that needed to be downloaded to the client slowed the process dramatically. We recommended switching from the commonly-used messaging download to a file transfer. We saw dramatic performance increases, from a typical download requiring 90 seconds or more to 2 seconds.

Midmark

Midmark is a medical equipment manufacturer that wanted to automate their remote sales force. The goal was to be able to download the corporate information they needed in their sales situations--product inventory, pricing, customer credit history, etc.--into individual sales PC, and upload new order information from the clients into the server on a periodic basis.

With Visual Basic front-end screens, we helped them design a system that allowed the sales people to upload and download information between the corporate and client systems quickly and easily, either remotely via modem, or while they were at the home office through either a direct connect or a serial connection to the server. We installed the inverted file indexes on the server database to provide them quick access to the server information they needed for their sales calls. Visual Basic screens then let them navigate through that information on their PCs while on the road.

Go for it!

Those of us in the industry know that one of the surest ways for job security is to be responsible for improving system performance--you can never have enough.

I strongly recommend inverted file indexes to provide multiple keyword retrieval for large databases. I recommend looking at alternative transfer technologies such as file transfers for files over 6K bytes. And, as always, you should give your users the benefit of your knowledge in what kind of queries can and should be done in a client/server system. I've found these methods to be enormously helpful in boosting the client/server headache: performance.

Above all, I encourage you in your efforts to develop efficient client/server systems. The productivity--and yes, performance--gains are well worth the effort.

If you have any questions, or would like to discuss any of these recommendations further, feel free to call me at DISC at (303) 444-4000.

How to Effectively Select Software Development Tools.

PADMA SREENIVASAN
Hewlett-Packard
100 Mayfield Avenue, 36LD
MountainView, CA 94043
Phone: (415)691-5113
Fax: (415)691-5485

Introduction

The 90's Software Development Environment is a complex, disparate system. The traditional Life Cycle phases overlay the Client/Server, Object Orientation, Surrounding the Legacy Systems, Re-Engineering, etc. along with well defined Methodologies associated with each component. The market is enticing the developers with a myriad of tools and methodologies based on viable technologies carefully layered over the above Architecture. It is very confusing even to a well seasoned developer how to select the technology, how to implement with the best possible tools to retain the competitive advantage, to gain momentum in providing the lead in emerging technologies at the same time creating new market strategies by taking the risk. IBM and Digital are good examples of being the late bloomers in the RISC technology field, using UNIX operating system, not to mention the NT venturing (CHICAGO, CAIRO... may be TOKYO to come next!).

The mind boggling choices may lead to misconceptions in understanding and developing a profitable path (process) in search of the "Holy Grail". Since there is no 'one size fits all' theory of selecting/implementing the software development tools one needs to be a great artist in creating a customized total solution to the software tool selection process. This paper describes a systematic process in selecting a suite of tools for software development components. *The list of tools is neither an endorsement nor a recommendation for any vendor or his products. It is there for the purpose of illustrations only.*

The need for a well defined process in selecting tools

Development tools have proliferated today's market at a staggering pace. This proliferation demands a high technical knowledge in the capabilities of the tools in each of the market components and trends. There is a clear advantage in using the appropriate tools. The advantages are:

- Ease of use
- Short learning curve
- Rapid prototyping and quick development
- Reduced cost
- Productivity improvements
- Adhering to Standards enables portability, Interoperability, Interconnectivity
- Transparent access to different databases

The disadvantages in selecting the wrong composition of tools are:

- Complexity of implementation
- Steep learning curve while mixing and matching
- Configuration control leads to unmanageable software versions and the communication between applications may grind to a halt
- High cost and low productivity.

The steps and Strategies

Since Software Development is a broad category, before selecting tools one must **have a clear vision** of what the software needs to do. In a customer engagement, the consultants must **understand the customer's objectives, business scope and policies**. Understanding and being able to articulate the customer's problem situations thoroughly with no assumptions made or else the said assumptions verified/validated before proceeding, is the **first step**. In many cases the customers' culture or cultural differences should be accounted for since the players have an International market stage to perform. Needs and wants are two different entities. While selecting tools one must **perceive the requirements** clearly and conceptually.

The next step is to **create a framework** around the business scope/objectives. At the framework level one need not be concerned with the technology or tools. A framework provides the Global View of the complete project. Let us consider the example of XYZ company which decides to play in the International arena, the framework will have the break down structure of regions like Europe, South America, Asia Pacific etc. It need not define, at this stage, how to implement, what time frame, which products etc. . The framework defines a global process or methodology encompassing the whole enterprise or workgroup. For example, an Information Technology (IT) framework will address the software/hardware aspects (Figure - 1) without any references to the technology or process models.

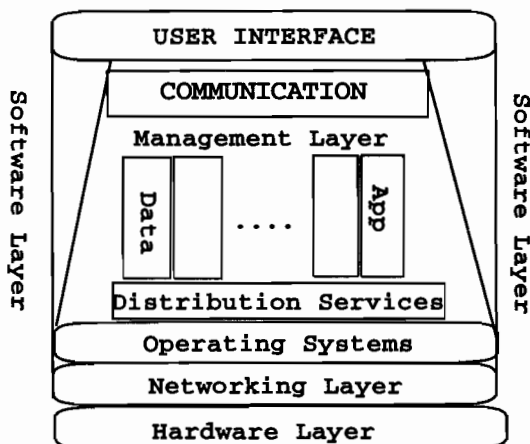


Figure - 1

A framework should involve all the three main components of an Application:

- * User Interface or the Presentation component
- * Application Logic
- * Data Access/Management component

The user interface or the presentation part mainly deals with user input/output. Application logic figures out how to interface the data that it receives from the users and how to present it back to the user again later. Application logic can build transaction processes with security and data integrity aspects. The data access or management component manages the storage, retrieval and validation of data, and enforcing security to a degree. The placement of the application logic, either with the presentation or with the data layer - PCs or the more powerful Work Stations? - eliminates certain tools in the market. Some terminal emulation tools (3270 or VT100-X) lack the application's logical capability/aspect.

One way of selecting the suite of tools is to look for the best of the lot in each of the above three categories. For example, VisualBasic, Visual C++, VisualWorks, OS/2 Presentation Manager, Motif... are some of the tools to create the user interfaces. Pen computing provides another form of user input. With "Virtual Reality" tools flooding the market Graphical User Interfaces (GUI) have a new meaning to "look and feel" concept; It is "wear and walk/fly"!! A framework design must take into account the "**People, Process, Technology, Management**" algorithm to be cost effective, thus gaining a competitive edge. The existing architecture with the computed risk analysis of the business infrastructure changes, organization restructuring and business process re-engineering should pave the path to a more solid foundation to a framework.

Choosing the tools along the above three Application components is based on the technology one adopts. If Object Orientation is the technology then PowerBuilder, Neuron Data- OpenInterface are examples of the Presentation tools. There are tools specific to not only the technology but also for the Operating systems: UNIX specific tools, PC based tools, NT tools, OS/2 tools and so on. The mind boggling array of tools should be used discriminately. Choosing the wrong set of tools may lead to productivity stagnation, setting up hurdles/obstructions or losses. A well architected technology with the appropriate tool mapping for each components of the framework, will aid us in jumping high over the hurdles.

Hence, the definition of framework leads to the the blue print- **The Architecture**. This level will assign the **technology** component to the **methodology** defined at the the framework level. An architecture should be modularized to include the Network architecture, the communication architecture, the Hardware architecture, the operating systems and the database structures etc. An architecture will include tools for the various components. It is worthwhile to have a tool selection process defined at this level. Here is an example of an Information Technology (IT) Architecture: (Figure - 2).

The Techno(ana)logy

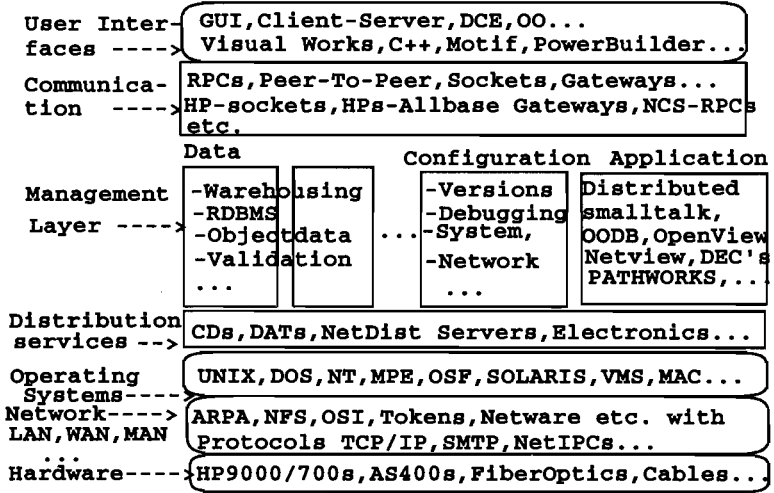


Figure - 2

In the above illustration one would choose one or more Operating systems with one or more Network protocols but one Presentation tool, with one Communication mechanism and one set of data access tool. In a distributed client server environment one will have different set of

hardware and network configurations with one or more databases along with the database management tools. Choosing the tool suite is synchronously geared to choosing the appropriate technology first. For example, if one chooses the Object Oriented(OO) technology built on Client/Server architecture, with the scope of accessing the legacy data to reduce cost and time, to improve productivity while having a hardware setup of PCs (desktop environment) and HP-UX work stations on an Ethernet LAN, using TCP/IP protocol (the legacy data is DB2 (IBM MainFrame), the recommended client tools are: [2]

Client --> Visual C++, VisualWorks, PowerBuilder, VisualBasic, Galaxy
 Galaxy and VisualWorks run on both PCs as well as UNIX systems while
 PowerBuilder and VisualBasic are PC only tools.

HP provides an Allbase gateway to DB2 based on LU6.2 communication. Informix, an RDBMS vendor also has a DB2 (DRDA) gateway, so does Gupta Technologies. The reusable code of OO technology based on a proper methodology like Booch or on the simplistic methodology by Coad/Yourdon, Shalor/Mellon or FUSION (HP Lab's) for analysis and design phases, is cost effective and conducive to a team development environment. IBM DB2 gateway using DRDA will be on HP-UX as of July.

The above recommendation is listed in a report "Client Development Environment Recommendations" by Jarek Baryeka et al.[2]. One tends to agree with these authors when they warn the software developers in selecting tools take into consideration "the diversity of size and the Complexity of IT projects-no single tool can do it all for all the projects". The criteria these authors used- performance, standard compliance, platforms, company history, scalability, portability etc.- are already built into the sieve. Refer to step six-the sieve-below.

Here is one such (TOTAL) solution:

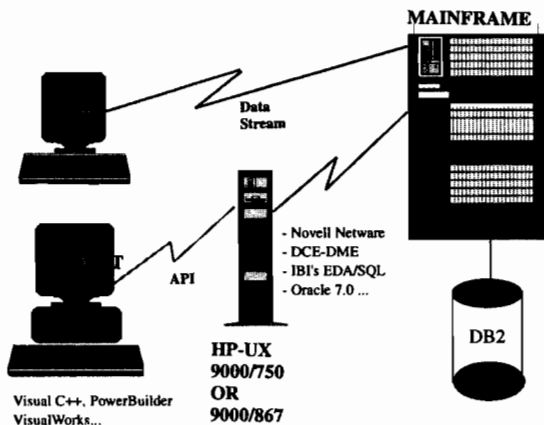


Figure - 3

The fourth step is to create a project plan. The most important step is to have a constructive project plan with well defined objectives, phases, timelines, deliverables associated with each

phases, roles with responsibilities, distribution strategies, training/support techniques etc. One can select tools available in the market for project management, change management and project control or any other aspect of a project. The tools here are Microsoft Project, MATE tool by Advanced Development Methods, FirstCASE by AGS, Teamwork by Cadre Technologies Inc., LBMS's all inclusive Process Engineer, Bachman tool, to name a few. A project plan must have the results of Cost analysis, Risk analysis along with control structures. To quote Doug Van Kirk "However, ... tools can also be misused. Applied to the wrong kind of problem, or implemented without appropriate planning, application builders can generate a jumble of screens that are as difficult to manage as the tortured spaghetti code found in many existing programs". (Infoworld, May 17, 193 page 56).

Steps one through four above have **built the infrastructure** necessary for the Software selection process. At step three we defined the Architecture, the all-inclusive, critical step. Choosing the appropriate technology for the selected architecture is the **fifth step** in the selection of tools process. Here is a list of available and emerging technologies - some of them are still in the nascent stage while the others are maturing fast, while some are waning even faster!!

- Distributed Computing Environment (DCE)
- Client/Server(2-Tier, 3-Tier, Multi-Tier Environments)
- Distributed Database Management, DME
- Object Technology
 - Distributed Object technology
 - Object Database Management
 - Structured Methods
- CASE/ICASE
 - AI/Expert Systems
- Platform based(PC/Workstations) software development
- Surround the Legacy systems
- Re-Engineering
- Re-Writing
- MainFrame Alternative (MFA), Down Sizing or Right Sizing
- Graphical User Interfaces(GUI) to access Databases
- 4 GL
- RDBMS
- Data Warehousing
 - Open Warehouse
- Gateways (Communication as well as Database) and APIs
 - Remote Procedure Calls (RPC)
 - Messaging
 - Peer-to-Peer communications along with Application to Application Communications (APPC)

This is an ever growing list. We may need a selection process for technology itself!. The business objective along with the scope of the project should guide us in choosing the appropriate technology. In fact, none of the above are mutually exclusive. One can argue the above list is not technological but more along the process or methodology structure. But remember technology is only a means that uses an appropriate methodology aided by the tools to achieve the end (business scope)!

A "**Survey of Advanced Technology,1993**" provides a selection process for technologies. To quote Vaughan Merlyn, Partner, Ernst&Young "*This is the first report I have seen that covers virtually all the important emerging and advanced technologies. More importantly, it examines and compares penetration, impact, and barriers to use, to offer a comprehensive picture of advanced technology diffusion. If you are considering any advanced technology, or are wondering how your organization compares with industry norms, this report will be of value to you*".[1]

Today, software development is synonymous with client/server(distributed) technology. Gone are the days of Traditional Life Cycles along with CASE and AI methods. Industry refers to these software as "Shelfwares"! "Artificial Intelligence(AI) has always seemed just a bit esoteric to make much of a dent in the average corporate IS budget".[4]

The war with the "middlewares" has just begun!

Middlewares

In a client/server architecture the client refers to the application that requests a service or data and the server is the application that honors the request. Client/server can also refer to the system the applications are running. For example a VisualBasic program running in a PC, a client, can request for data from the database existing at a UNIX box, a server, through another functionality layer, called as the "**middleware**".

A middleware is used predominantly for seamless data access/management while providing synergy among the existing applications. It is neither the client nor the server but everything in between. It is that layer that facilitates the communication between the application and the network protocols and below (Operating system, Hardware etc.). It is independent of and well insulated from the network protocols, operating system, computing platforms etc. It is based on the "Cooperative Processing" models of the 70s. *"The present status of middleware is one not uncommon with new technologies: flux, confusion and competition. Basically Middleware is a utility".[3]*

There is a long list of tools available as middleware. While selecting tools, following a well defined and accepted standard - The SQL Access Group (SAG) is developing such a standard - will enable portability and cross platform operations. In order to win the war with the middlewares the industry has sliced the middleware into three pieces- top level, middle level (middle of middle equals "noware"!!) and bottom level. Top level communicates with the application, the middle is for data access (gateways) and the bottom is I/O with the operating systems and the network. Refer to figure 4.

The name "Middleware" is patterned by TechnGnosis Inc., Boca Raton, Fla. This company's product is called as "Client/Server/Middleware".

"COSE"wares

COSE stands for Common Open Software Environment, founded by the "unlikely bedfellows" IBM, Sun Microsystems, Hewlett-Packard, SCO, USL, and Univel to ward off the fear caused by NT (Microsoft Corp), and Open System Foundation (OSF) is not moving at a high "Silicon Valley Speed"! COSE adopts the X-Open standard. There are many system management tools known as "**COSEWAREs**" are available in the middleware market today. Open System Foundation is also creating a standard for the management software as part of DCE. Distributed Management Environment (DME) will not be available before the end of 1995. Many vendors offer their own flavors. Tivoli System's DM/Framework, Unix System Lab's TUXEDO TM, EcoSphere's (recently acquired by Compuware, Mich) Ecotool, HP OpenView etc. are some such tools.

Unix International (UI) is building a UNIX-Only Atlas environment which is interoperable with OSF's DME.

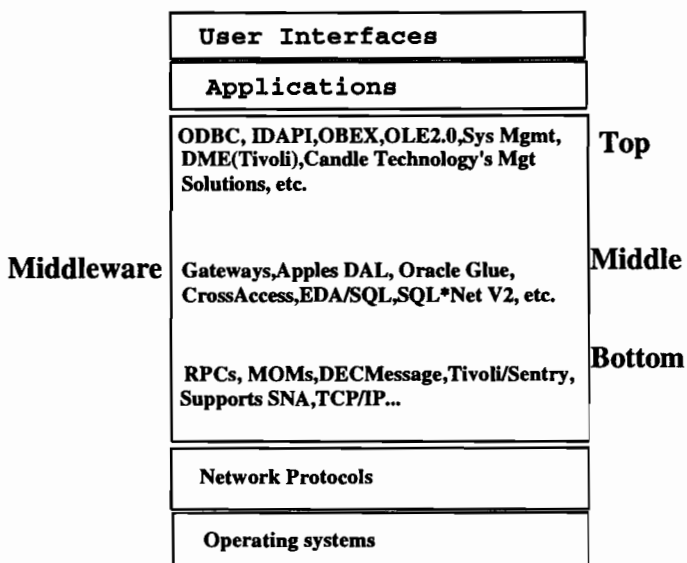


Figure - 4

Yet another cut at middlewares specific to the database access/management is "One-to-many, Many-to-one and Many-to-many". [5] The idea here is using a GUI front-end, applications are created with the help of these middlewares to access transparently a wide variety of data with different format, and residing in relational or non-relational databases. The term "one-to-many" refers to using a single front-end to access more than one databases. Powersoft's Powermaker, PowerViewer, Gupta's Quest ... are some such tools. As the phrase implies "many-to-one" is accessing one single database through many front-end tools. Because it uses a specific database, it is vendor provided/dependent frontware. Sybase, Oracle are some such examples. The preference here is to have a "many-to-many" access. IBI's EDA/SQL and CrossAccess are tools of this type. CrossAccess using DB2 SQL supports IBM's LU6.2 along with TCP/IP protocol with no RPC calls or no new APIs.

Judith Hurwitz's consulting group recommends Cross Access Data Delivery System from the Cross Access Corporation as the best of the "many-to-many" access tool. It is very helpful to note the 8-point questionnaire selection process applied by the Judith Hurwitz's consulting group. "We evaluate eight criteria in our Benchmark. The highest rating is five, the lowest is one. All the ratings are added together, then divided by eight, to devise an overall rating for the tool at the time we evaluated it. The rating is based on our knowledge of the product and our perspective[6]". The eight questions that this consulting group uses is already in the sieve (step 6) in one form or another.

The trend in today's market is to merge more and more middlewares into the operating system itself. Windows bear the testimony to it. Ergo, selecting a middleware depends on the Architecture.

Objectwares

One of the emerging technology is Object Orientation. Distributed Objects, modeling, and management tools mapped into client/server architecture, based on the Common Objects Repository Based Architecture (CORBA) standard, set by the Object Management Group (OMG), are becoming very popular because of their reusable structure and the reusable Objects' Library, offered by vendors like Interactive Development Environment. The traditional CASE models can easily evolve into Object models. Refer to Figure - 5.

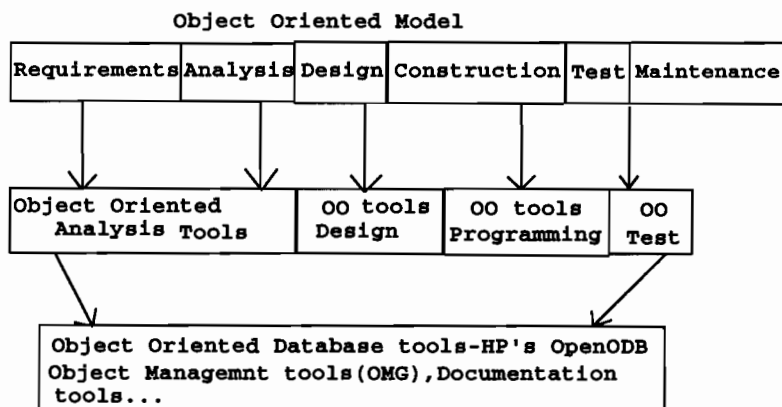


Figure - 5

Object Oriented tools fall into four basic flavors: Analysis, Design, Languages and Databases. Coad/Yourdon, Shlaer/Mellor, Booch/Rumbaugh, HP's Fusion are some of the well known analysis methods. Object International's Objectool follows Coad/Yourdon/Nicola approach. Object Oriented Structured Design (OOSD), Hierarchical Object Oriented Design (HOOD), Real-Time Structured Design by Shlaer/Mellor have set design guidelines. C++, Objet-C, Smalltalk/VisualWork, Digtalk, Eiffel, Ada are some of the common object oriented languages. Through Object Oriented DBMS data can be shared easily due to their modularity, reusable code structure and the reduced control complexity. Again selecting Object Oriented tools depends on the project's scope and objectives.

The next step is to **create a list of currently available tools** for the chosen technology. One way to slice the tool market is to look at the different categories. Refer to Figure - 6.

- The **basic tools** like Operating Systems, 3GLs, Compilers and Debuggers, Testers ...
- The **infrastructure tools** or the middlewares like data integration tools, Data Management tools, Data repositories, data dictionaries...
- The actual **Development tools** like Rapid Application development tools, Prototyping tools, Graphical User Interfaces(GUI) tools for clients and Database Server tools, On Line Transaction Processing(OLTP) tools, Configuration and Version Control tools, Object Oriented tools, Cross Platform Development tools like UNIX to PCs and PC to UNIX ...
- The **Information Access tools** like query/report, 4GL, Data Analysis and Modeling tools etc.
- **Project Management tools** like Document, Process, Requirement, Contract, and Performance management ...
- **Application Generation tools** ...
- **Others/Miscellaneous tools** like Database Gateways, APIs etc.

Figure - 6

Creating an up-to-date matrix of available tools in the above categories -Figure 6- will help us focus on the scope of the project and the means to achieve it effectively, quickly and not to miss out on the very critical time-to-market aspect. Some tools are so versatile that they can be applied to more than one technology mentioned earlier in the list of technologies.

The plethora of tools specific to operating systems pertaining to the current architecture or the adopted one, help to eliminate many tool suites, even though the trend in the market is to provide tools that run on multi platforms. There are cross platform generation tools like Open Interface by Neuron Data, Aspect by Open Incorporated, XVT software and Galaxy by Visix Software...

A word of caution by Monash Information Services, "Buyers make a big mistake today when they focus on technology-based feature lists before adequately understanding needs and benefits. Software buyers are constantly disappointed during the purchase process because they forgot that the critical first step is defining the required benefits. Evaluating the features is the second step..." [7]. The same authors recommend creating a short list of vendors and products, and matching this list with the benefits (analysis) list in choosing the right vendors and

products along with a "Success Model" which is similar to the framework and architecture (The blueprint) mentioned in steps three and four. The "success model" mentioned here has the well defined path to the solution of the problem along with the type of applications to be developed with in the productivity framework. This model should have the outline for work breakdown structure (WBS) and the daily roles and interactions between the project teams, time gaps, cost analysis and an alternate path in case the risk analysis results are negative. In this article-"Know The Stakes And You Won't Have To Roll The Dice"- the authors Richard Currier and Curt Monash warn a buyer about the "six Deadly Traps in Buying Software". They are: "*Believing the initial requirements list is accurate or complete, Underestimating the importance of ease of use, Focusing on features and not benefits, Misusing available references, Buying by committee, and Buying vendors rather than the products*".[7]

Once we have the list of available tools to depict the architecture we chose, we put these tools through a **Software Evaluation Process (The sieve)**. This process (sieve) contains the following main sections: See figure - 7.

The Main Sections of The Software Evaluation Process (The Sieve).

1. General Information
2. Usability of Tools
3. Software Building Power
4. Tool/Prototype Attributes
5. Phases of Life Cycle Supported
6. Special Issues - Standard compliance etc.
7. Bibliographies
8. Hewlett-Packard Experiences
9. Summary

Figure - 7

Refer to Appendix -A for details.

Each section is composed of many sub-sections. If one needs a high level evaluation as opposed to this elaborate design, then one can create a "LITE" version of it by including just the categories needed. Assigning a relative ranking from 0 through 5 to the criteria "Limited, Adequate, Needs Modifications, Superior..." one can really come up with the least risky suite of tools. This is time consuming but reduces the risk and gets us the sacrosanct tool suite to ameliorate the software selection process.

A Case Study

Recently I was involved in a customer engagement where the customer needed a fast and easy access to legacy data in an IBM system, stored in ISAM/VSAM files. The database used was COGNOS(PowerHouse). In the existing structure the data was kept in many databases geographically scattered throughout the company. The data itself had different format. The users had old ascii terminals to access the information they needed. Normally to generate a report it will take months due to the lack of integration process between the existing systems. The customers objective was to reduce the cost, time to get the report and to increase profit by migrating to workstations and providing every user with a powerful PC. The data was updated only once a month because of the lack of resources, the storage problem and so on. The customer hired HP consultants to come up with a total solution. We formed a project team consisting of complementary skill levels; Analyzed the customer's current architecture with their objectives, scope and business policy. Created a framework around a 3-tier client-server architecture using VisualBasic for the client on a 486 vectra, Twisted pair to connect to TCP/IP LAN, Cambridge Technology Group's (CTG) subsidiary Open Environment Corporation(OEC) tool kit for creating a HP-UX functionality server to access the data with in 2 days!. This configuration met all customer's requirements (needs/wants). We selected VisualBasic and the OEC toolkit after evaluating them through a "Lite" sieve. We created a prototype as a proof of concept which evolved into production.

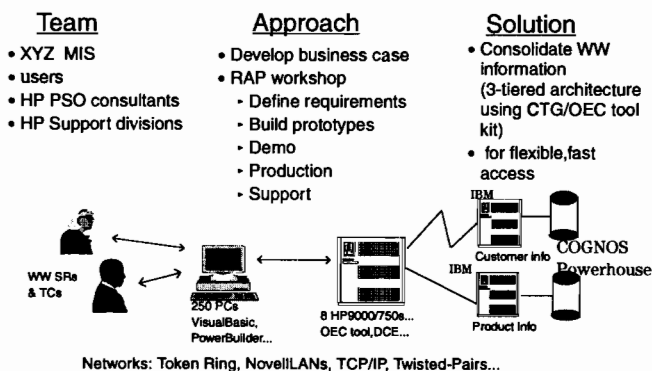


Figure - 8

An Alternative

An alternative process is to create a cross mapping of problem situations-to-tools. The problems can be looked in the categories of "people, process, technology and management". For example, one can create the following matrix:

- Converting the current C/COBOL application into C++ for migrating to Object Orientation technology
Tools: SMARTsystem by PROCASE Corp, Mastero II by Softlab Inc...
- Re-engineer a Mainframe application into client/server
Tools: Maestro by Softlab Inc, CTG/OEC toolkit, HP's DCE, TeamWork-CadreTechnology Inc.
- Develop a GUI front-end to the legacy system
Tools: KASEWORKS, Inc's CASE:PM, Flashpoint by Knowledgeware, Inc, Meta Software's MetaDesign, MS's VisualBasic, VisualWorks by ParcPlace, COGNOS PowerHouse, Neuron

Data OpenInterface...

- Information Integration tool

Tools: Atherton Technology-Software Backplane,CL/7 by Century Analysis, DEC's Cohesion, HP's Softbench C++, Synervision, Informix 4GL Toolbus, JAM by JYACC Inc...

- Object Libraries for Object oriented Environment

Tools: SE by LBMS, ObjectStore by Object Design Inc., Interleaf5...

- Building Real-Time applications

Tools: Interactive Development Environment's Stp, Meta Design, Verilog Inc's GEODE,Cadre Teamwork/SIM...

- Software (configuration, Version Control, test and validate), Project, System... Management tools --->Tools: HP's Softbench, Intelli Corp's OMW, Intersolv'sPVCS, ParaSet by Software Emancipation Technology(SET), MS Project, Digital's Autoplan, HP's OpenView, McCabe&Associates Toolset...

For example, the following table can be created, periodically updated and referred to:

<u>Problems</u>	<u>Solutions</u>
- To replace the Mainframe to cut cost	- Client/server distributed processing
- To reuse the existing code in creating more functional mission critical applications	- Use of middlewares and Rapid prototyping
- To improve On Line Transaction Processing	- Replace the ascii terminals with workstations or PCs
- To increase productivity	- Invest in Project/Time management training/tools
- To maximize reusability of codes	- Move to Object Oriented technology
etc.

Resources

These resources help with one's sophomoric attempt at selecting the necessary development tools. This is not a complete list and I am sure you can all add many more too this list.

- CASE Associates Inc., 14915 SE 82nd Drive, Clackamas, OR 97015. Tel: 503-656-0986

- Hurwitz Consulting Group, Inc.; 44 Pleasant St, Suite 200, Watertown, MA 02172-9746.

Tel : (617) 926-5500

- Monash Information Services, 888 Seventh Av, NewYork, NY 10106.Tel:212-315-3120

- Ovum Limited, 1 Mortimer Street, London W1N 7RH, England. Tel: 44-71-255-2670

- The Faulkner reports and the Gartner Group's consulting services...

- ButlerBloor Ltd, Challenger House, Sherwood Dr, Bletchley, MK3 6DP, England.

Tel: 44-908-373311, Fax: 44-908-377470

- OpenSystems Today, UNIX Review, PC tools and other magazines...

- Trade shows have Product Expo to see quick demos of any tool that has more Promise than Potential.

There are "Application Development Tools-Product Guide (ADT)" by Monash through Sentry Publishing Company (Monash IS), "Client/Server Development Tools Notebook" by Hurwitz Group, and "Ovum Evaluates" to help evaluate products available for any aspect of software Development. CASE associates publishes "Application Development Trends"(formerly of CASE TRENDS) with Vendor, tool, Platform etc. information matrix every month.

Conclusion

The software developers have diligently mastered the art of innovative software creations but the tool selection process still lacks the mastery. One longs for a less laborious and well accepted process for selecting software development tools in order to avoid the pit falls of the shelfwares, vaporwares and the no-wares of the software industry. Ergo, this paper outlined the following steps:

1. Define scope; Have a clear view of the business requirements, policies and objectives.
2. Create a software and hardware framework
3. Choose the technology along with a methodology -Architecture.
4. Create a project plan with roles and responsibilities, time frames and phrases, risk and cost/benefits analysis for re-engineering, restructuring, rewriting etc.
5. Create a list(s) of available tools for the selected methodology comprising all the aspect of software development. For example, languages, Operating systems, debuggers, configuration control, system management tools, database access, Project management tools...
6. Pass each tool through the sieve or a customized (Lite) version of it, while assigning a relative ranking on the capabilities/attributes of the tools selected. See figure 9 and 10.

The Software Evaluation Process (The sieve)

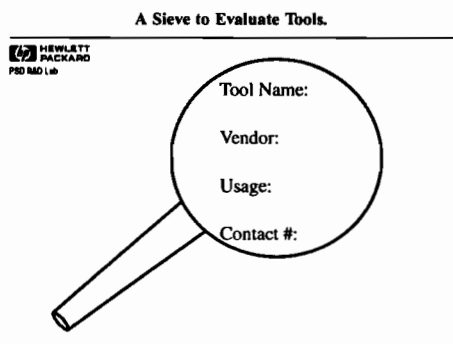


Figure - 9

This heuristics(sieve) approach in selecting the dynamic-wares, is no panacea but minimizes the risk of investing in stale or stagnant-wares!.

To put it all in a nut shell, refer to figure 10:

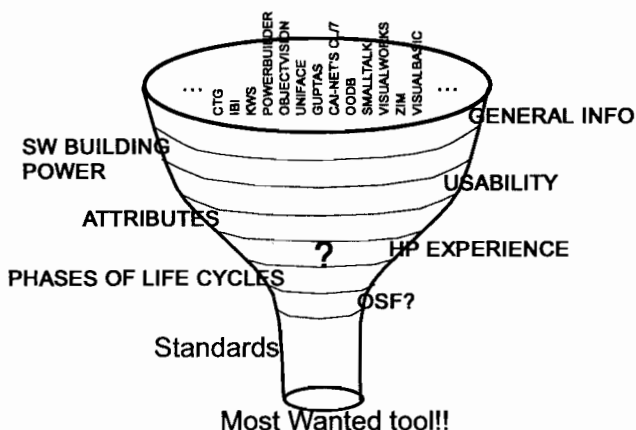


Figure - 10

Appendix: A

1. GENERAL INFORMATION:

A. Product Family Name:

B. Vendor:

Address:

Phone Number:

FAX Number:

C. Product Structure: Includes platforms and price information

Client:

Server:

Character Terminals:

Host Machines: (all platforms)

D. Distribution Media: The software is distributed on 3.5" floppies systems or DAT tapes ...

E. Databases Supported:

1. Reads:

2. Generates: Databases generated by the tool or database management tool

F. System Requirements:

Required Disk Space:

Optimum Disk Space:

Required RAM:

Optimum RAM:

Additional Utility Requirements:

G. Product Performance: Synopsis of product reviews and comparisons, Benchmarks

H. Vendor Profile:

1. a. Brief Description of Product Positioning/Product Strategy:

- b. Software Product Sales Dollars:
- c. Number of Copies Sold:
- d. Number of Software Product Employees:
- 2. Revenue Distribution by Sales Channel:
- 3. Revenue by Product Category:
- 4. Revenue by Application:
- I. Customer Type:
 - Revenue by Industry:
- J. Competitive Positioning:
 - Describe the technical proficiency of the vendor:
 - Top Three Competitors:
- K. Prototype Focus:

2. TOOL USEABILITY:

- A. Ease of Use: GUIs, Learning time,...
- B. Learning Resources: Training, availability, tutorial, demo, consulting,...
 - Training:
 - Consulting:
 - Technical Support:
- C. Configuration Management/Version Control: Direct support or interfaces to CM tools
- D. Network Capability: Describe network operations & functionality
- E. LANS Supported: List the LAN Managers
- F. Network Connection Type:
- G. Ties to Methodology: List methods, diagramming techniques supported
- H. Team Engineering: Describe multi-user access/sharing of file objects and the scalability of tools to support small to large teams
- I. Security: Describe access control available to the tool's repository
- J. Interoperability (tool-to-tool): Describe how interfaced to other vendor tools

3. SOFTWARE BUILDING POWER:

- A. High level Languages/Targets: List languages generated by the tool and the target environments supported by the prototype generated in those languages
- B. Reusability of the Prototype Components: Describe the prototype components stored in the tool's repository
- C. Functional Modeling: Describe models and/or templates used to build prototype with writing code
- D. Data Modeling and Data Management: Describe the data modeling, database design and re-engineering capabilities and notations
- E. Flexibility:
 - 1. Ease of Porting: For both the development tools and the prototype generated
 - 2. Ability to Customize: For both the development tools and the prototype generated
- F. Object-Oriented: Describe the extent and type of object-orientation and the building of class libraries

4. TOOL/PROTOTYPE ATTRIBUTES:

- A. Completeness/Accuracy: Of the user interface (if applicable) and whether the prototype is a throwaway or can evolve to production
- B. Security: describe security features which can be built into the prototype (e.g., protected field, read or change, etc.)
- C. Data Import and Export: Describe if/how prototype can import/export data such as bit map graphics

D. Evolving Development Capability:

1. Localizing : Support for international character sets:
2. Real-Time Performance: Describe operating efficiency/constraints of the prototype
3. Reliability: What is the known reliability of the prototypes generated?
4. Scaleability: How big or complex can the prototype be?

5. PHASES OF LIFE CYCLE SUPPORTED:

Use a scale:

None: Tool provides no support for the phase

Limited: Tool provides some support for the phase but is not the primary function of the tool.

Adequate: Support for the phase is one of the primary functions of the tool.

Superior: Tool's support for the phase is recognized as among the top ten contenders in the field.

Life Cycle Phases:

Project Management:

Process Management:

System Planning:

Requirements Analysis:

System Design:

Prototyping/Simulation:

Construction:

Verification/Validation:

Document Generation:

configuration Management:

Reverse/Re-Engineering:

Maintenance:

6. STANDARDS COMPLIANCE: National and International standards supported .

For example OSF, ANSI, X-OPEN, POSIX...

7. BIBLIOGRAPHIES:

A. Case Study Bibliographies: Article title, name of publication, date-page numbers. Also names of product users and companies cited in the articles.

B. Product Review Bibliographies: Author, article title, name of publication, date-page numbers. Names of product users and companies cited in the product review

8. HP EXPERIENCES:

A. Contacts:

B. Types of Applications:

C. In-depth Evaluations:

9. SUMMARY:

References:

- [1]. Survey of Advanced Technology, 1993 -Chris Pickering, SDI Systems Development, Inc.
- [2]. Client Development Environment by Jarek Baryeka et al. Version 1.0e January '94.
jinx@vallere.mayfield.hp.com
- [3]. The computer Conference Analysis Newsletter, July 22, 1993, n323 p9(1).
- [4]. Datamation June 1, 1993 v39 n11 p75(2). "Tools for fast, easy access to legacy data" by David Baum.
- [5]. Hurwitz Consulting Group, Inc. Client/Server Tool Watch. Cross-Platform Application Development Vol. 3, No 2, April 1994. The topic is "Middleware: The search for Universal Data Access" page 5.
- [6]. Same source as [5] above. "How Cross Access Stacks up Against the Hurwitz Tool Benchmark" - Page 12.
- [7]. A paper by Richard Currier and Curt Monash "Buying Application Development Tools: Know The Stakes And You Won't Have To Roll The Dice". "The Six Deadly Traps in Buying Software ". Monash Information Services is located at 888 Seventh Ave, New York, NY 10106; (212)315-3120. 1-800-5-MONASH(1-800-566-6274).

Other References:

- Software Magazine, July 1993 v13 n10 p55970 - CopyRight by Sentry Publishing Company Inc.; "Middleware needed to plug C/S holes" by Janet Butler.
- The computer Conference Analysis Newsletter, Dec 17, 1993 n334 p3(1). "What's hot & What's not" a speech delivered by Rich Finkelstein at the DCI's C/S conference.
- Client/Server Architecture by Alex Berson.
- Object Oriented Software Development, A practical Guide by Mark Lorenz.
- HP's CASEdge - CASE and Objects - A Primer.

How to Design GUIs Effectively Using GUI Standards and Specs.

PADMA SREENIVASAN
Hewlett-Packard
100 Mayfield Avenue, 36LD
Mountain View, CA 94043
Phone: (415) 691-5113
Fax: (415)691-5485

Introduction

One of the buzzwords of the 90s is GUI, or Graphical User Interface. The information integration process with its emphasis on client/server technology, uses Windows on PC platforms and Motif interface on workstations, Macintosh Windows, OS/2 Presentation Manager and Windows NT etc. for client interfaces. The main purpose for using GUIs is to create "aesthetically consistent, ergonomically and visually pleasing applications"[1]. Designing a character-based application is more tedious, lacks the "look and feel" of the application, let alone the ease of use! To a user the user interface is the application, because that is how he/she communicates - data inputs and outputs - with the application. The Graphical User Interfaces, as opposed to character-based interfaces, provide a variety of functionality to form a visual and behavioral foundation for the design of the application. The increasing processor power and the falling prices of memory devices aid in the age old concept of graphical interfaces to applications. The paradigm shift in the software industry towards client/server technology enables the creation of simple GUI clients to access directly the existing/legacy databases - relational or non-relational. There are many flavors of GUI, those that are front-end to a 4GL - hence supplied by the vendor -, those that are development environments, and those that are strictly user access only. "There are two kinds of GUI builders, one concentrates only on the look, the other addresses both look and feel. A builder that creates only the look of a GUI is called an Interactive Design Tool (IDT). One that generates the appearance and the behavior of a GUI is called a User Interface Management System (UIMS)" [2]. UIMS are generally more expensive because of their added capabilities.

Any vendor dependent software, like a GUI front end to a 4GL, needs to follow the de facto or de jure standards to provide maximum portability, ease of use and reusability of the software. This article briefly describes the standard for various components such as color, font, listbox, pop-up-menus, icons etc. used in designing the GUIs. I will concentrate on the Microsoft's Windows guide, Visual Design Guide for VisualBasic, and Motif Design/Style guide 1.2 version.

The need for a standard

In order to provide a consistent, uniform look of the application to the user community, it is necessary to adopt a standard. Unfortunately perhaps, there are many standards available to choose from. For example, the OSF/Motif standard defines a workstation GUI based on MIT's X-Window system, SUN's Open Look, IBM's Presentation Manager and Microsoft's Application Design Guide outlines a standard for PC windows environment. In fact I recently heard about the standard C++ user interface toolkit (Fresco) by the X Consortium! Adopting a de facto or de jure standard provides a universal language to communicate (the nonverbal communication aspect) between the applications and the developers. The developers know how difficult it is to provide a uniform look between the applications, for each application addresses different user needs, scope and business policies.

The development of GUI standards is still in nascent state. Industry is still struggling with evolv-

ing design methodologies for GUIs. Designing the application for a well seasoned programmer "who knows the ins and outs of programming are not only fast disappearing but also the user community is dictating the norm for application interfaces"[1]. Hence without standards the user is at a loss! Standard based consistent application interfaces are conducive to the creation of new applications based on the previous design-reusability. GUIs need to reflect intuitively the appearance of an application. Following standards ensures the application looks the way it works and works the way it looks (look and feel)! The developers community is divided into three groups: COBOL programmers who have never used GUIs and new PC programmers who haven't understood, let alone heard of, the importance of standards in GUI and those who are in between. Refer to figure 1. Imagine if the red light means different things in different places, how chaotic it will be, not to mention the meaning of flashing red, green or yellow!!!!

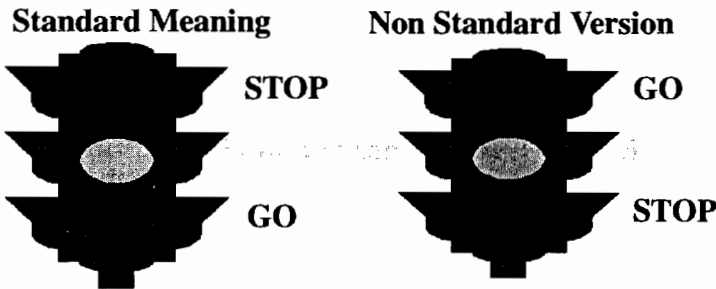


Figure - 1 (Standards)

Hence standards assist with the non verbal but powerful communication between users and the applications. Standards also increase the profitability by cutting costs through reusability, portability, interconnectivity, interoperability along with performance and security. The vendors gain a competitive advantage by reinforcing the standards in their design.

Issues to consider

Since a user perceives the user interface as the application, the GUIs must reflect the human elements in its appearance with pleasing colors, fonts, menus etc. The architecture along with the technology and the platform it is based on should play a major role in the design of the GUIs. What works in a 16-bit architecture may not behave the same way in a 32 or 64-bit systems. The 256 color resolution provides a finer tuning than a 16 color scheme. The location of the data or database, the power of the CPU boxes, the size of the monitor (14", 19" or 25") all contribute directly or indirectly to the GUI design. Refer to figure 2.

There are other issues related to the controls themselves. For example, where do we place a list/combo box? Since the real-estate is very limited, how many controls should be displayed with out cluttering the screens? When do we use radio buttons, when to use the check box, list box etc.? Can the user change the fields in a text box? Should the unavailable fields in a list box or sub-menu be grayed out? Should there be defaults for selections? How should the error messages be displayed other than being meaningful and directive? Should the data be validated in a text entry field?...

The Issues

- **The Human Elements**
- **The Architecture(HW,SW,NW...)**
- **The location of data or the Database**
- **The power of the CPUs**
- **The size of the monitor, memory...**
- **The color resolution**
- **The Project's Team work**

Figure - 2

One should take into consideration all these issues to reap the "optimum design potential" of a graphical user interface. GUI designs must be "user-centric and not ego-centric! The user should be involved from the beginning of the development process, and all aspects of the GUI should be usability tested to make sure they are simple, straight-forward and clear" [3]. The design goal must include not only the aesthetic appeal but also a seamless interface to the computer, thus providing the "best cognitive sense to the end user ... through intuitive means of direct manipulation" [1].

What Does this document contain?

The rest of the paper talks about System Controls, Application Controls, Icons and a case study using VisualBasic, based on both Microsoft's (MS) Window's style Guide and Motif's Design Guide. In Motif's terminology the controls are called widgets and gadgets.

System Controls

There are certain visual elements of a window interface that cannot be changed by an application, for example, the frame, the 3-D effect, certain color combinations and system fonts. In Microsoft Windows, a 3-D effect is achieved by passing a light source (from upper left corner) using dark (RGB #192-192-192) and light (RGB # 128-128-128) gray shades. Refer to Figure 3.

In Motif, 3-D is achieved by dithering, along with top and bottom shadows for any rectangular shape. Motif uses a constant light source (from upper left at a constant 45 degree angle) on a 2-D "visual slab" through which the widgets will be held in place. "Early attempts at creating the three-dimensional illusion were done by delineating light and dark chamfers with only black and white onto two-dimensional styled widgets. Various styles were employed, including explorations of rounded corners and drop shadows" [4]. Refer to figure 4.

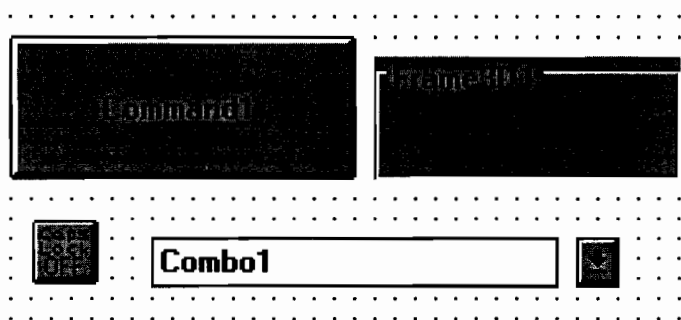
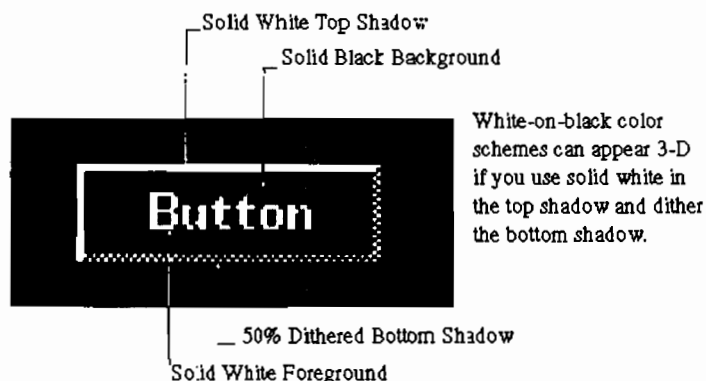


Figure - 3

Three dimensionality is used to provide visual clues to the action for functions. When we click on a command or a radio button, they seem to recede to denote the action being undertaken. This raising/depressing of buttons is achieved through shadowing in MS Windows and by dithering along with top and bottom shadows in Motif. Black outlines for edges, due to the VGA resolution-16 color combinations- give a dramatic contrast in window frames, icons, buttons etc. Any mix of white, light gray, dark gray and black can be used effectively to create the 3-D look.



Source: Visual Design with OSF/Motif, Page 28, Figure 2-12

Figure - 4

Colors

Color is another piece of the system controls. According to the "Commission Internationale d' Eclairage"(C.I.E-France) trichromacy theory human eyes have a few Blue cones to focus with, while we have the equal number of Red and Green! Hence a combination- a proper mix - of colors provides contrast and emphasizes the "emotional and psychological" aspects associated with them. The exciting red, the calming green, the vibrant blue add to the look and feel, in directing attention to the point, especially with icons. Colors are useful in grouping objects. Majority of the population are visual learners and hence using the same color for a group of related objects embellishes the association.

In spite of their colorful appeal, one should be very cautious in mixing them. For example, red and green are bad for eyes. Blue is for background but not for small icons or thin texts. Colors should be used as "redundant cue". If you notice, most of the default colors are subtle, and not "circus like". Motif Style Guide recommends "very dark screen objects on a light background, very bright objects on a dark background, and bright colors all command user's attention. Use colors as redundant aspect of the interface; that is, use it to provide additional differentiation among screen objects". Every widget in Motif adopts the four color spaces-top and bottom shadows, select color and the background- defined by the server, to create the 3-D effect. In Motif's color map black and white are the two fixed color spaces. This means in a 64 color map we have only 62 spaces and in 256 there is only 254 spaces. The type of display cards along with the color spaces used in a color map play a major role in defining/achieving the true colors. Though the reduced cost of color monitors contribute to the fast demise of the monochrome displays, they are still around. Dithering is especially handy here. One need to be cautious while using fonts with dithering, "because the fonts tend to break up when displayed against a dithered background". See figure 4.

Fonts

Like colors fonts also have emotional attributes. Fonts can inspire, create a positive mood, and motivate the users. Fonts can illustrate the association among group of objects through their various sizes and weights. Have you noticed that fonts are less visible on screen than on a printed hard copy? As we get older, we need reading glasses due to the size of fonts!

MS window uses 10pt Sans Serif bold for title bars and menu items while using an 8pt Sans serif bold for dialog boxes. Bold font is needed so that the unavailable items can be grayed out and hence lightly visible. Refer to figure 5 and 16a. In 5, the pull down menu "Restore" is grayed out.

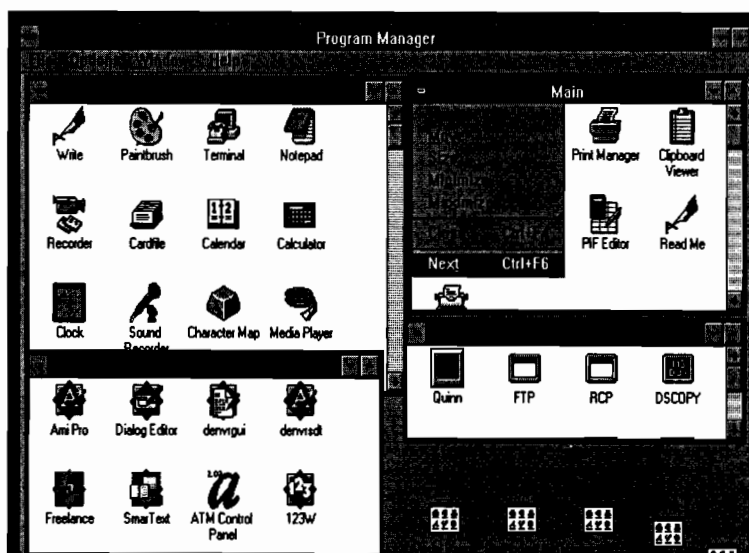


Figure - 5

Fonts that have jagged edges (due to low density of pixels) cause eye strain. Italic, serif fonts, *Shelley Allegro Script* are some examples. MS windows also uses San serif 8pt non-bold font (the smallest available without causing eye strain) for icons.

Motif recommends "whimsical" fonts for grabbing attention and "no-nonsense" fonts for finding

the "interface components quickly and effortlessly" [1]. Fonts are the communication link between the human and the computer and hence must be comprehensible at all times. Motif uses **San serif** for **system** and **serif** for **user** fonts. Widget labels and system messages use San serif font. To quote Shiz Kobara, "system fonts are specifically designed bitmapped fonts represented by what are called *glyphs* consistently positioned within their character cell...and can be laid out to be fixed pitch or variable pitch" [1]. When choosing fonts one must consider native language support, if the target audience for the application is international. Since some languages have 2 byte characters, enough spacing must be accounted for in their cells.

The application area that is specific to user inputs (edit areas) uses user's choice fonts. Courier is an especially popular font with users. As long as the user uses fonts that are different from the system fonts there will be less confusion. Here the user fonts should emulate the system fonts in size and in cell positioning.

Application Controls

In this section I will look at different Buttons, Toolboxes, Status Bars, Cursors and other controls based on both MS Windows and Motif (widgets and gadgets) design standards. I will not talk about the resource settings, the function calls etc. while discussing Motif widgets.

Buttons

In MS windows there are three different button groups: Text, Graphics and Control buttons. See figure 6 below.



Figure - 6.

These buttons have states associated with them, depending on pressed/not pressed actions. When choosing these buttons the user must ask the *what, where and how much real estate* questions. *What* happens when one presses the button? *where* do we want to use the buttons and *how* are we fitting these controls into the limited screen space?

Text buttons are used on a toolbar, forms or dialog boxes. VisualBasic Visual Design Guide recommends "Text buttons in a dialogboxes are usually a standard height of 10 Dialog Units on Forms, or on toolbars can vary in height (typically 22 pixels) and also use the 8pt San Serif bold font. If vertical space is a great consideration and aesthetics are not as important as the usability of the product, text buttons may be the best solution for a toolbar". Text buttons states are: "disabled, mouse down and up". All states are visible to naked eye (visual states). Refer to figure 7.

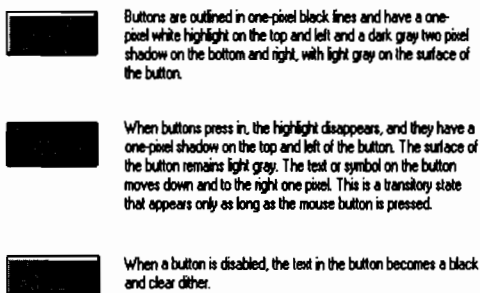


Figure - 7

While presenting many text buttons vertically as a group, allow 6 pixels between buttons and 16 pixels between group of buttons and horizontal layout calls for 8 and 18 pixels respectively. Graphics buttons are widely used in toolboxes, rulers and other controls. Here is the standard for these: **Toolboxes: 24 by 22 pixels; The graphic image on a button : 16 by 15 pixels.** If the user is concerned with screen space, then smaller sizes 22 by 18 and 16 by 12 are specified, respectively. The images in the button have a black outline with non-dithered white line. The Bitmap standard defines, "up, down, mouse down, indeterminate, disabled and down disabled" which are the six states for graphic buttons. See figure 8.

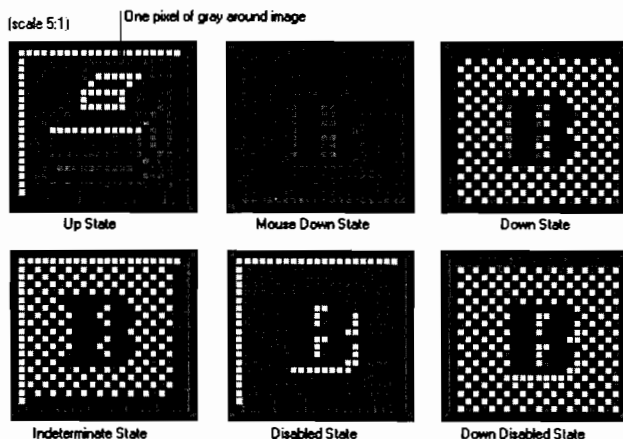


Figure - 8

The source for figures 6-8 is a screen dump from the VisualBasic Visual Design Guide.

Graphic buttons that require an action (function associated) with them have mouse down state while all the rest have any of the other states. From figure 8, one can perceive that all states

have one or two pixels gray border with black or red graphics. The indeterminate state is similar to upstate, except for the dithering. "This visual state occurs when a selected object shares several properties at once, for instance a selected body of text may have bold or italic." One needs to be very careful when including texts in graphic images -the graphics above text- because this elongates the button size vertically/horizontally. Menu buttons and menu bars mix graphics and texts. For example, in figure 9 we see menu items with/without texts. Radio buttons and check boxes usually don't have graphic images or texts on them. Toolbars - usually horizontally placed on the top part of a window - and Toolboxes most commonly found on the left side (figure 9) or floating - have images and texts displayed together. Toolbars also have dropdown lists, accelerated buttons and statusbars etc. 28 pixels is the standard height of a toolbar, 22 pixels for dropdown lists with 6 pixels between any button groups. 2 pixels of light gray for toolbars and 2 pixels of dark gray drop down shadow for toolboxes with 3 pixels for title details are the recommended. In a Monochrome (EGA) one may use alternate black and white buttons, if the buttons are attached with no spaces in between them. Refer to figure 9.

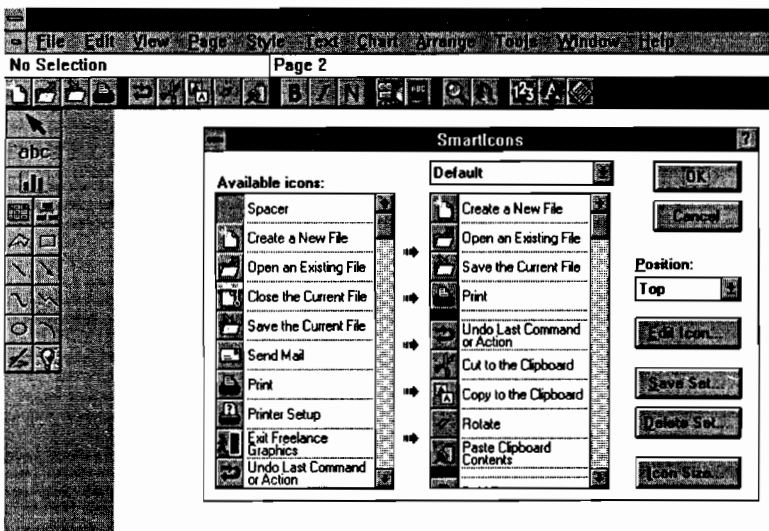


Figure - 9

Status bars, especially in MS VisualBasic, are at the bottom of the window horizontally placed with locked status information (red/gray 3-D insets) about the same height as the titlebar, 8pt San Serif non-bold font. See figure 10.

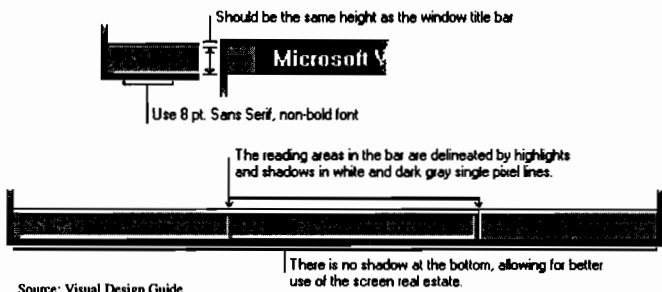


Figure - 10

Cursors/Pointers draw immediate attention to the defined areas of actions and positions on the screen. For example, the normal arrow cursor changes into a dithered box during move and has a plus attached while copying etc. In fact the Microsoft NT system provides many choices to the normal cursor; the arrow with an oscillating tail is my favorite one! Figure 11 shows the

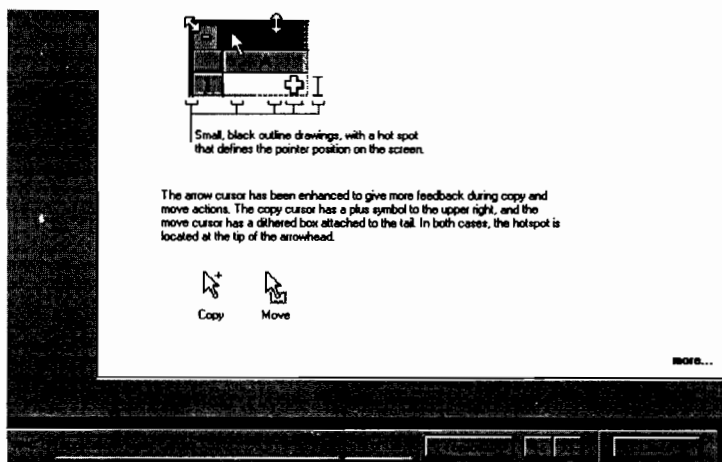


Figure - 11

many cursor shapes along with a status bar at the bottom. Everybody recognizes the hour glass when the system is (busy) in action. Figure 12 shows the cursors for Microsoft NT system. Some of the NT cursors are animated to reflect the real world!

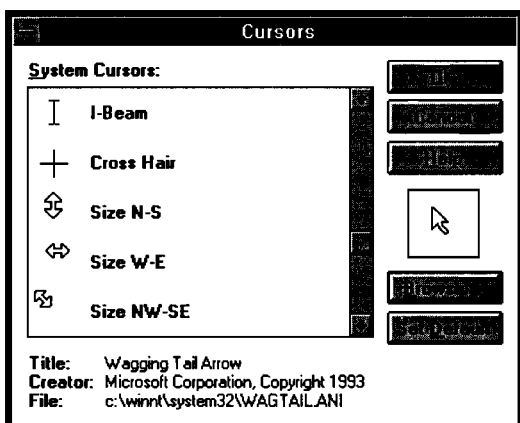


Figure - 12

Other Controls are sliding bars, spin buttons- pressed up, down arrows-, scroll bars etc. Figure-13 shows an example.

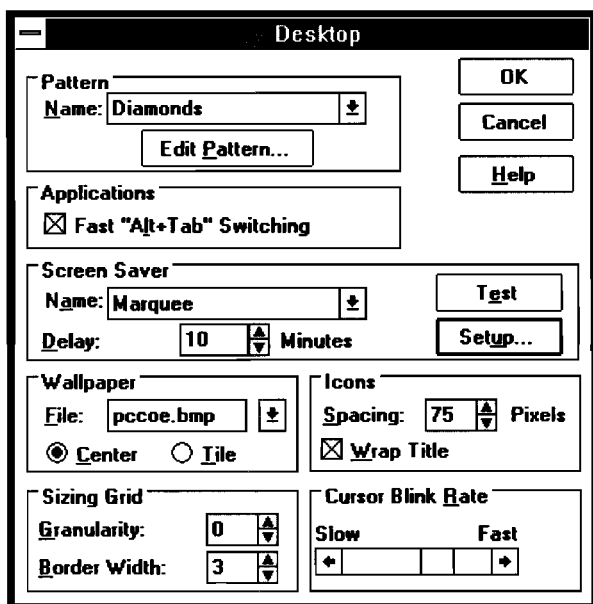


Figure - 13

The Motif style guide is based on IBM's Common User Access (CUA) and Presentation Manager's user specification guide. For the application interfaces Motif provides Widgets and Gadgets; thus ensuring consistency/uniformity.

Motif's Widgets and Gadgets

There are two kinds of widgets: Primitives and Managers; the latter can contain other widgets in it. Gadgets are "*windowless widgets...essentially high performance widgets without the Xwindow system overhead*" [1]. There are six types of Primitive widgets and gadgets: Arrow, Cascade Button, Label, Pushbutton, Separator, and Toggle.

In the widget hierarchy managers are parents. Scrolled windows, Frames are parents/managers. Unlike widgets, gadgets inherit the colors, fonts etc. from their parents. It is a recommended practice even for widgets to follow the colors of their parents and the **traversal highlights** with the standard of 2 pixels thick, for uniform spacing between the widgets so that the widgets can be highlighted one at a time.

Labels

Labels display strings alone or mixed with images. They should use the system fonts - San Serif, 2 pixels width - for size and positions. Inactive labels will be 50-percent dithered. Always align the text in a label, especially while changing the font size. Though the label string is at the center there are top, bottom, left and right margins available along with margin height and width. These resources help with the widget alignment, especially while entering the text.

Pushbuttons

A pushbutton has a rectangular frame with a label in the middle; the label specifies an action. When the action is selected by a mouse button, the label area gets depressed, the top and bottom shadow colors are reversed until the mouse is released. The size of a pushbutton depends on the size of the label. Again there are resources one can set to position the labels. Motif's Standard recommends a horizontal alignment for Pushbuttons in an application for "space efficiency". If placed vertically so as to align the labels, the row-column manager will force all the push-buttons to the same height and width of the widest pushbutton. Since Dialog Boxes have default Push Buttons with action of return or enter, one should be careful. Unlike the regular push buttons (2 pixels for top and bottom thickness), the default ones have 1 pixel thick. The margin settings must be the same for both default and regular ones. Default button must always be placed at the bottom left most position.

Toggles

Toggles, the class name for "n of many" - checkboxes - and "1 of many"- Radiobuttons - have a square or diamond indicator, respectively. Checkboxes can be used alone or in a group when many options need to be selected. When selected the top and bottom shadows will be reversed to be visually depressed. By clicking the same one again, the checkbox can be de-selected, unless it is part of a group then it will be added to the list of selections. There is a 4-pixel space between the indicator and the label. With the Radiobuttons only one selection can be made, "just like the buttons on a car radio from which it got its name". There must be a minimum of two Radiobuttons, so that in order to deselect the one, we need to click on the other. To save space one can use "Option Menus" instead of Radiobuttons or long lists.

Arrows

Arrows are equivalent to the scrollbars and behave like the Pushbuttons with reversed shadows when selected. There are four directions -up, down, right, left- associated with Arrows.

Cascade Buttons

Cascade Buttons are menu bars with attached menu panes or connected to many menus through arrows or ellipsis. These don't reverse the shadows when selected for they are labels or pixmaps. The specific resources help with the positioning of labels, pixmaps and the buttons themselves. See figure 14.

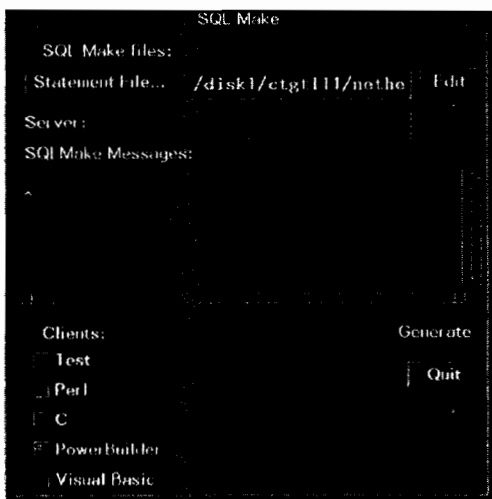


Figure - 14

Separators

These are lines - shadow_etched_in or Shadow_etched_out - providing visual partitioning to specific areas. They are most frequently used in menus, dialogboxes. Shadow_etched_in look is recommended for their "appearance provides a subtle and aesthetically pleasing scribed line". A 2-pixel thickness is recommended, a pixel each for top and bottom shadows. Separators can be horizontal or vertical. For graphics/images there are other less used types of separators: Single and Double, Dashed and Double_Dashed and no_line.

Manager Widgets

Frames

A Frame, as its name suggests, provides a 3-D beveled edge look with top and bottom shadows containing only one child if it is a primitive. With the edge types of "Frame In/Out, Etched In/out", frames can be used to provide a 3-D look to pixmaps/images. The thickness of the shadows is user settable while the area in between the shadows will be of the background color. Refer to

figure -14. The "Frame In" is the recessed look and the "Frame Out" is the raised look if the frame is of the same color as the parent. Normally the shadows are of 2-pixels width. The look of the frame changes dramatically by varying the thickness of the shadows. The "Etched In/Out" look is used when grouping widgets to provide the "pop in/out" look into the panel with a shadow thickness of 1-pixel. See figure 14.

Paned windows can be sized by the users and are used to partition an application windows work area (figure 14). The default distance between any two paned windows is 8-pixels.

Other Widgets (controls)

Scales

Scales are used to dynamically adjust/set some calibration. For example adjusting a mouse speed to reflect the mouse-sprite movement. Visually it appears to be a recessed slide bar. The top of the scrollbar will have the background color while the dynamic part will be in the foreground color. Scales can have horizontal or vertical layout through resource settings.

MenuBars

Menubars play an important role in choosing or making it available until pulled down, the different functional components of an application. The items should be consistent with the other applications in positions, fonts, color and sizes. Menubars are manager widgets that can contain cascade buttons. "The horizontal spacing of the children in a menubar defaults to 0 and should not be changed." The cascade buttons in a menubar must adopt the design principles of Pushbuttons. The recommended level for cascading with menu panes is two so that with the original pulldown menu there are a total of three cascading which will cause less eye strain. The recommended spacing between the items of a menu pane is 0-pixels, the default is 1-pixel, so that the items will appear one below the other. See figure 16a.

Scrollbars, Scrolled Windows

Scrollbars allow the user to view beyond the current screen either one line (vertical) or a few words (horizontal) at a time. The indicator in the scrollbar is proportional to the text/image being viewed to the total and its color defaults to the background color. There are resources one can use in setting the height, the width, the orientation, the colors and the shadow thickness.

Scrolled windows are used to view a portion of the texts, spread sheets, charts, icons etc. They have one or more scrollbars, appear to be recessed windows. To obtain the maximum view area editable or not, certain resources, for example the margin height, width etc., have default value of 0 which should not be altered. The scrollbars in a scrolled window can be placed at Top_Left/Right, Bottom_Left/Right for appropriate viewing convenience. Refer to figure 15.

Text

Text widgets are for inputting data, - digits or letters or any characters - and texts into the application. They can be single or multi lined. They can be directed through appropriate labels. The foreground color is recommended (default) for text along with they are through resource settings and color codings. Like Pushbuttons text widgets height and width are adjusted to the font in them. When placing Pushbuttons and text widgets together in a frame or group they must have the same width, height, margins etc. for a consistent look. See figure 14 and 15.

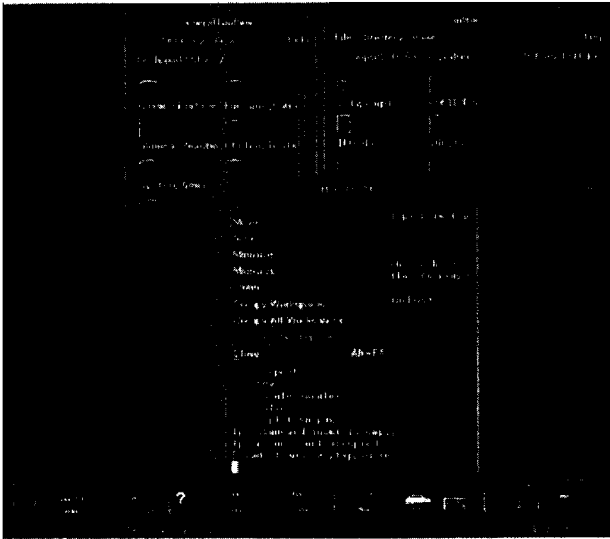


Figure - 15

Scrolled text are scrolled windows restricted to scroll on texts only to maximize the screen usage. Scrolled text can also be set to editable or not. The width, height, border thickness all default to 0 pixels where as the recommended value is 2-pixels so that the texts will be well above the edges. Scrolled text can have the four scrollbar orientation as mentioned above in scrolled windows.

The Motif Style Guide recommends the right and bottom orientation for scrollbars. If the word wrap is available then one may not need horizontal scrollbars. A non editable text or scrolled text must have their parent's background color and an editable one must have the select color which is darker to identify the text area as editable.

Lists, Scrolled Lists

Lists can display output results or provide strings to be selected by the user but not modifiable by the user; it should be changed programmatically or through dialog boxes. The total number of items in a listbox and the scrolled list, the visible number of items, the total length and the spacing between the strings etc. can be determined through resources. The margin height, width should be 2-pixels and the default of 0 must be changed for consistency through out the application. Scrollbars can be added/deleted automatically through resource settings. It is not a good practice to use scroll list over Radio/Push buttons. Scrolling is done over data only.

There are other controls like Bulletin Boards, Forms and RowColumn container widgets which are invisible but have a great say in the children placed on them.

Bulletin Boards are basically for placing the children at x, y coordinates when x, y are specified. The default position is 0,0 and the children will be on top of the others and will not be resized when their parents are!. Bulletin Boards are mainly for dialog boxes. There are four types of

dialog boxes: system_modal, application_modal, modeless and work_area. Modal refers to get confirmation from the user before taking any further action. If you need to resize the children then use Forms. Form is a container widget where children can be placed relative to each other. RowColumn widget is a container used to place an "array of dissimilar-sized Pushbuttons into rows in which each pushbutton retains its unique size or into columns in which each Pushbutton is forced to a single size." [1] In Motif the Window Manager plays an important role in placing, moving, sizing of the widgets and gadgets. One should remember to set the Active Input Focus color to be different from the other windows so that the user can easily identify the window that is receiving the keyboard input because it is "attached" to the keyboard. This is different from an application that has Input Focus exclusively irrespective of other windows. There are special configuration files that allow a user to set the resources of the window manager.

The final topic is Icons design defined by MS windows Style Guide, VisualBasic Visual Design Guide and Motif Style Guide.

Icons

When designing icons one should think of its size, color, image style and 3-D objects depiction. Icons are used to denote the function they execute. Usage of real-life metaphors helps with the purpose and identification and also with their positioning on the screen. For example the folder, the trash can etc. are good "pictorial reminders or visual vocabulary" aids. While grouping icons uniform look will help focusing on the icon function with less distraction. "Windows has adopted an illustrative rather than symbolic icon style. Illustrative icons communicate metaphorical concepts better than abstract symbol." Black outline make the image visible in any background. Use colors with great restraint for "color in icons should enhance and not detract from their meaning." There are standard(32X32) and small(16X16) sizes. The images or pictures on the icons should not only be proportionate to the size but also use contrast colors to reduce the jagged edges. See figure 16 (source:Visualbasic Visual Design Guide) and 5. Figure 16-a shows a cascading menu.

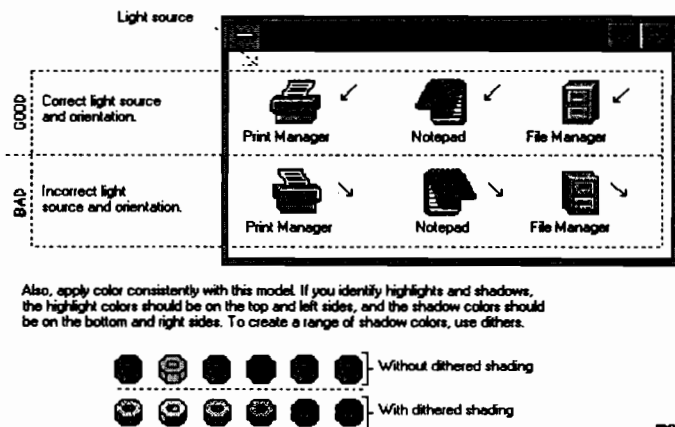


Figure - 16

Motif's icons serve similar purposes. Icons are great for localization because of their symbolic representation that requires no words. They save lots of screen real estate. When I look at the icons I think of the famous words of the poet Milton, "They also serve who stand and wait"!

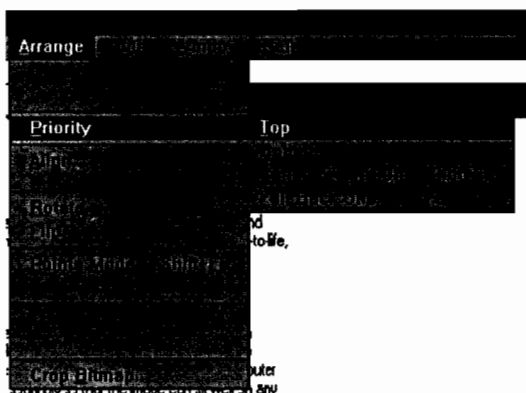
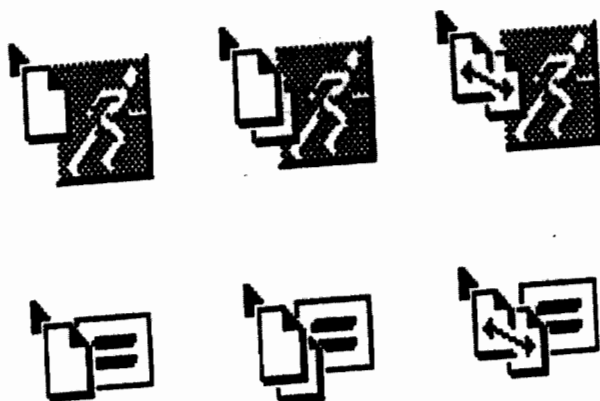


Figure - 16a

According to ISO standards, "Icons are bitmapped images that represent an application or document... must be able to be manipulated, such as moved or deleted". In Motif when a window is minimized the label - the title of the window - seem to be pasted to the background and there is a second kind where the label identifier appears on the outside, known as Pictograms. Refer to figure 14. The use of an icon box is very popular with Motif designs. There are window manager resources one can set in the position of icon boxes, colors, sizes, style etc. Since icons can be "symbolic, whimsical, literal, or abstract" along with 2-D or 3-D looks, the developer must apply caution in not over crowding, confusing the user. Refer to figure 15. The Motif Style Guide recommends, for a drag icon the pointer must change to show the drag operation on the three components of the drag icon, namely: "A source indicator, An operation indicator and A state indicator." See figure 17 which shows the move, copy and link operations for a drag icon.



Source: Motif Style Guide, Release 1.2

Figure - 17

A Case Study

Recently I was involved in creating the client side GUI for an ALLBASE application using the Cambridge Technology Group's (CTG, Boston, Mass) subsidiary Open Environment Corporation's (OEC) 3-tier client/server tool. Client was a PC running VisualBasic and the functionality and the database servers ran on HP-UX 9000/725 system, using HP's Allbase. The application enabled the user to get sales information on a particular part, get a list of part numbers, add or remove part numbers from the inventory etc. Each functionality was represented by a Visualbasic screen providing uniformity in color, fonts, functions etc. Figure 18 is a bad example because it did not use uniformity in color. Figure 18 shows four screens with different background colors.

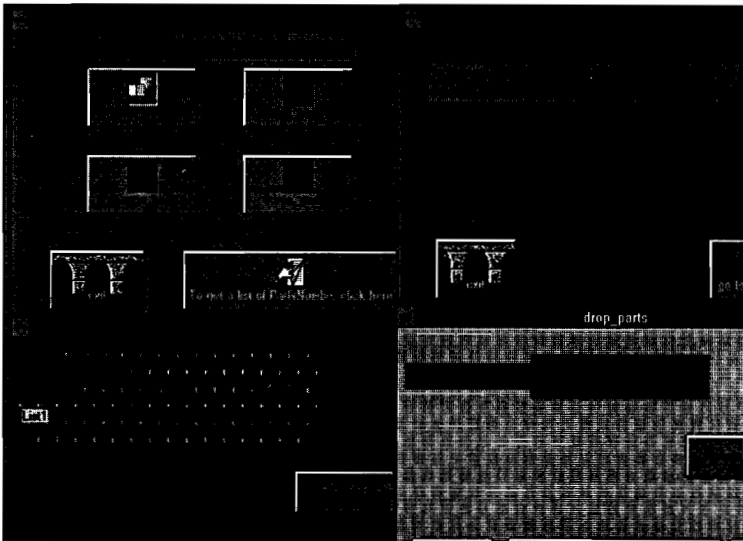


Figure - 18

The application had a standard "about" screen, a security screen and the function screen etc. Figure 19 is a good example for it follows Motif standards for subdued colors, courier fonts of size 10pts, non cluttered screen design, scrollbars to the right etc.

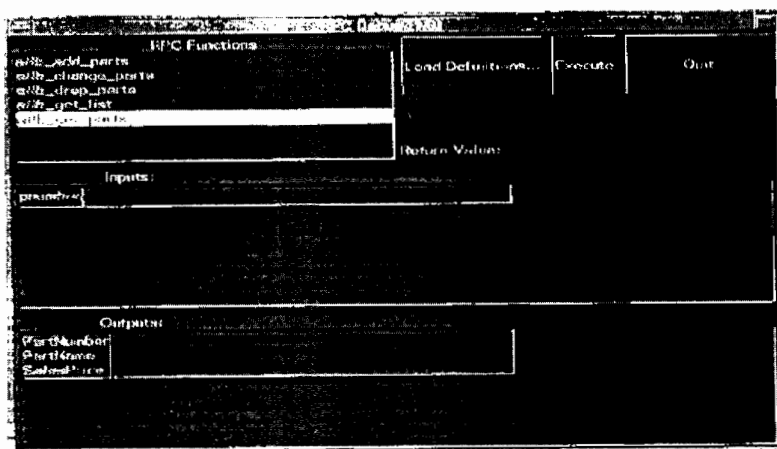


Figure - 19

Conclusion

While designing GUIs one should adopt the following commonsense rules: (MS Windows or Motif)

- Design with a user's ease-of-use and comfort in mind.
- Components should be flexible for changing and manipulating.
- Use the eye pleasing color combination and fonts. Provide consistent look and feel.
- The icon images must reflect real life; Use real life metaphors with proper proportion. "Fingers on hands look better if they are not true_to_life, but true_to_icon."
- Always inform the user what is going on with the application through the appropriate Dialog Boxes, Status Bars, Changing the pointer shapes etc.
- Do not present the Pushbuttons, Radiobuttons etc. in an array of rows and columns.
- Don't use long sentences for labels, menu choices (not more than one line of words).
- Adopt the same size fonts with the same styles and colors for the widgets and gadgets.
- No widget/gadget should be clipped by parents or by the boundaries, especially when resizing, moving etc.
- The separators should be touching the frames on all sides.
- Never present a cluttered look. Always make the best use of the real estate (screen).

By adopting these simple design rules the developer can empower the users.

The Future

The virtual reality being a "real thing", the GUIs will be multi dimensional and "wear and fly" will become a popular jargon. Pen computing and voice activating all add different choices to user interfaces. Multimedia evolution will dominate the user interfaces to be more flexible and adaptable. Computer animation technology will improve the communication between the humans and the machines. As customer demands increase in these "pervasive technologies", the GUIs or any user interfaces will have increasing capabilities, perhaps more than one can dream of, and with their new functionalities they will provide creative controls to overcome the

shortcomings, if any, of the man-machine interactions.

References:

- [1]. Visual Design with OSF/Motif by Shiz Kobara, Page 196. HP Press.
- [2]. UNIXWORLD, June 1992 page 72-74, "Buying Basics of GUI Builders", an article by Allen G. Taylor.
- [3]. GUI Tips & Techniques from Corporate Computing Inc. Volume 1 Issue 1, July - September 1993.
A paper by Bette Palmer "Tips From the Trenches: Effective GUI Design".
- [4]. Visual Design with OSF/Motif by Shiz Kobara, Page 3. HP Press.

Other references:

- IBM's Common User Access(CUA): Advanced Interface Design Guide, Boca Raton, FL IBM, 1989.
- Microsoft's The Windows Interface: An Application Design Guide.
- OSF/Motif Style Guide Release 1.2
- Microsoft's VisualBasic Visual Design Guide.
- HUMAN FACTORS INTERNATIONAL, INC. video on the "Do's and Don'ts of 17 GUI tools". Tel : 1-800-242-4480 or 515-472-4480. Fax: 515-472-5412.
- The X Journal -July -August 1993. Vol2 No 6. Pages 42-44.

Paper # 1015
Getting the Most out of HP-UX
Andrew J. Phillips
Hewlett-Packard Co.
Technology Solutions Lab
2015 South Park Place
Atlanta GA 30339
(404) 850-2937

Abstract

As the usage of HP-UX systems grows, more and more leverage can be taken of the unique programming environments that HP-UX provides. These environments range from the older and better documented command line interface (or shell) through the more recent and complex CDE (Common Desktop Environment) and DCE (Distributed Computing Environment) standards.

Any HP-UX system provides a multi-user, multitasking environment that is, at its' core, a general purpose computing system. That implies a significant amount of flexibility in terms of resources and configuration.

For the purpose of this paper, I will assume a reasonably well-configured system running HP-UX 8.0 or later that is connected to a LAN, since that describes the majority of HP-UX machines.

For conciseness, I will also assume that the reader understands and is familiar with the concepts of the HP-UX operating system, the file system, permissions and security, and knows how to use the man pages. If that is not the case, I would recommend reading an introductory text such as "A UNIX Primer" by Steven Prata. For clarity, the reader should also have some experience using a programming language such as Pascal or C. If you have read a man page in the last month or so, you should have little difficulty understanding this paper.

This paper attempts to provide an overview of some of the principles and practices associated with rapid systems development using the Korn shell. First, an informal survey of some of the available tools for application development is presented. Then, some of the capabilities and access methods of HP-UX systems are examined; next, some basic shell and shell programming concepts are explored; followed by a case study of a real-life utility being used today. I have included some examples that should help you understand the choices that need to be made and help get you started with shell programming.

Survey of Some Popular Tools

Users of HP-UX workstations have a rich programming environment and many tools to choose from. Many high-level languages are available, like C, Pascal and FORTRAN. I will refer to these as third-generation languages (3GLs). There are also many 4GLs available, like Smalltalk, PERL, C++ and so on. To help the novice (or experienced) programmer with using the X Window System, the so-called GUI builders (for Graphical User Interface) can not only help create interfaces, but also generate code for Microsoft Windows too. One example of such a package is UIM/X, from Visual Edge Software, Ltd.

To help the programmer integrate these tools, various programming environments like H-P SoftBench or comprehensive libraries like the MKS Toolkit are also available for a price. Deciding which tools to use in your environment is an important step in empowering programmers for rapid application development (RAD). I suggest the following as an aid to choosing the best tool or tools for the job.

Problem solving naturally depends on which kind of problem it is you wish to solve. If your problem can be solved by manipulating strings or ASCII files, the Korn shell itself is often a suitable choice (see case study later on). On the other hand, if your problem

involves significant number crunching or systematic calculations, one of the 3GLs mentioned earlier may be suitable. For GUI development or database access, either a 3GL or 4GL with appropriate libraries can be very cost-effective.

Network Access Methods

Since applications typically need to access other machines to retrieve data, possibly from a database, or to access other resources not available locally, this requirement is very important to any problem solving strategy. Application programs in general can use a wide variety of methods to access other systems. Some of the more frequently used include remote procedure call, or RPC, developed by Sun Microsystems but implemented by almost all UNIX system vendors; Berkeley sockets (really a library of system calls); and DCE or Distributed Computing Environment, a comprehensive framework of cooperative services from the Open Software Foundation. A relatively simple method to make files on one machine available to processes on another is by utilizing the Network File System or NFS. NFS allows one networked machine to access a directory from another networked machine, and to have it appear much like a local directory. We will see an example of how to do this in the case study.

The HP-UX Korn shell

I have chosen to use the Korn shell rather than the other available shells (C or Bourne) for several reasons. I began using UNIX under the C shell at first, but later experience with functions, aliasing and editing previous commands convinced me to change. Later, the emergence of a windowing Korn shell as part of the Common Desktop Environment (CDE) reinforced to me that choosing the Korn shell was working out. Therefore, I will not mention other shells in this paper, even though they are similar in many ways.

The very first interaction most people have with an HP-UX system is the infamous command line or "shell" program that provides the user interface to many UN*X systems. A typical command line prompt is '\$' (or '#' if you are root), akin to the typical MS-DOS command line prompt 'C>'. The UNIX shell has been described as one of the best 4GLs ever invented, because of its combination of power and flexibility. I hope to lend some credence to that statement. Conceptually, the shell can be thought of as something that reads commands, and then executes them, not unlike some language interpreters. In general, most shell commands follow a quasi-standard format I will refer to as the synopsis. An example with a description follows:

```
          |----- parameters -----|
COMMAND  <options>      <arguments>
```

Where **COMMAND** is the filename of the executable. Parameters are everything except the command itself; **<options>** are 'flags' that typically change the behavior of the command, and the **<arguments>** specify what the command should operate upon. For example, if I were to type 'rm *' ... the rm command will be found and executed, and the shell will 'evaluate' the '*' to represent all the ordinary files. So, this command will simply delete any files present in the current working directory. Assuming, of course, that the user of this rm command has the appropriate privileges. If the '-i' option is added (for example : **rm -i ***), the behavior of the command is modified such that it prompts the user for confirmation before removing each file.

So, the simplest problem solving approaches in HP-UX can first start with single shell commands. Since most problems are more complex than what can be accomplished with a single built-in command, the next step would be to 'blend' several commands together, in order to achieve the desired result. For example, if I wanted to collect all the addresses from my phonebook that were located in say, California, sort them, and mail them to my manager, I could type:

Getting the Most out of HP-UX

```
grep ' CA ' phonebook | sort | mailx jlv
```

To explain this step by step, the `grep` command selects all the records from the file named `phonebook` that contain the abbreviation for California (with blanks before and after). This is done so that any records containing the string 'CA' as part of a name, street name, or company will not be selected. These records are then redirected, or 'piped' as input to the `sort` command, which in turn 'pipes' the sorted records to the `mail` program, which then mails them to user `jlv`, who happens to be my manager. In the terminology of HP-UX, the `grep` command is a FILTER, because it filters out some of the input, and the '|' (pipe) symbol implies redirection, because it re-directs the standard output from the controlling terminal to somewhere else, in this case, an executable command, where it is processed as if it were read from the standard input. Another way of specifying redirection is to use the '>' symbol, which can be used to place output into a file, while the '>>' symbol can be used to append output to a file. We will see more examples of redirection later on.

In this way, we can see that by combining several commands, we are able to accomplish some relatively complex tasks with a minimum of effort. To help make this environment even more powerful, the shell also 'evaluates' the entire command line for you, so that filenames can be selected more easily using a regular expression (something the shell knows how to evaluate). We saw this before with the '`rm *`' example. A regular expression can be as simple as '`*.c`' (which even MS-DOS can handle). This '`*.c`' would be expanded to read all files whose names end with a period followed by a lower-case c. A slightly more complex regular expression is '`[Mm]*[Mm]`'. This would be expanded to include all files whose names begin with an 'm' (either upper or lower case) followed by any number of characters, and ending with an 'm' (also either upper or lower case). These regular expressions allow

the user to specify filenames (and some other things as well) using a flexible and powerful syntax.

The next shell facility I will discuss allows us to passively interact with system settings by reading and writing 'shell variables'. Shell variables are much akin to global variables in other languages, except that they do not have to be declared or initialized before use.

For instance, the shell variable named PATH is used every time a command is typed, since the shell will search the directories represented by the PATH variable for a matching executable file name. This is so, when you type `ls`, the system actually knows to execute `/bin/ls` (providing, of course, that `/bin` is part of your PATH). The actual value of the PATH variable can be determined by just echoing its' value to stdout (using the shell `echo` command). When assigning values to shell variables, one need only use the variables name, as in `PATH=/bin`. To read the current value of a shell variable, one typically will place a `$` before the variable name, as in :

```
echo $PATH
```

On my system, this results in

```
/usr/vue/bin:/usr/bin/X11:/bin:/usr/bin:/usr/local/bin
```

which is a small, but useful path. Other 'standard' shell variables include :

```
USER=ajp  
SHELL=/bin/ksh  
HOME=/users/ajp  
TERM=hpterm  
COLUMNS=80  
DISPLAY=r3125xxx:0.0
```

The 'set' command, with no parameters, will provide a more comprehensive list of the variables and their

values present in your environment. The ones shown above indicate that I log into the system as 'ajp', I use the Korn shell, my login directory is named ajp under /users, I am using the hpterm terminal emulator with a window width of 80 characters, and the name of the display I am using is r3125xxx.

If some of this is beginning to sound like Greek to you, I suggest that you try the "keysh" shell that is provided with HP-UX systems. This shell is helpful for people who are just learning because it uses the old familiar softkeys to help the user issue commands even if he or she isn't familiar with their arguments. The keysh also provides a status line with indicators for mail, the current directory, and other useful information. Also, one can see the resulting command generated by the keysh. This can be a great help to learn the 'standard' parameters for some individual shell commands.

Korn shell scripts

The next step in our list of problem-solving approaches for HP-UX is to combine a series of individual commands into what is known as a script. On the simplest level, a shell script can be just a few standard commands lumped together in a file so that one need only type the filename instead of each command in turn. In fact, the following example is just that. Note that the file containing the shell script should be executable, and a '#' sign indicates a comment. In a file named showme, I have :

```
#
# Example script 1
# filename: showme
#
date      # display the current date and time
who am i  # show user name, login terminal and time
pwd      # show the current directory
```

So, when I type showme ... the three commands in file showme are executed (just as if I had typed them in myself), and the result looks like :

```
Sat Jul  2 12:44:40 EDT 1994
ajp          ttyp2          Jun 30 14:53
/users/ajp
```

I could also have achieved this same result by simply entering " date; who am i; pwd " on a single command line. While this example only illustrates a rather trivial use of shell scripting, shell scripts in general are far more than just a way to save some typing. In fact, I hope to show that shell scripts can provide many of the features of high-level programming languages, while being easier to use.

Well, what features of high-level languages does the Korn shell support, you may ask. For starters, the Korn shell provides the normal if statements and case constructs. Loop control can be accomplished using for, while and until (a while loop that tests at the end). Shell variables can be used much like variables in other languages, except that the Korn shell supports fewer data types.

Parameter substitution is another powerful feature of HP-UX, both when writing shell scripts and using the shell interactively. In simple terms, one can refer to the parameters to one's shell script as \$1, \$2, \$3, etc. (corresponding to the first, second and third parameters) or as \$*, which refers to all the given parameters at once.

For example, to help illustrate the use of control structures, parameters, and looping, the following script will upshift lines (read from stdin) containing the string represented by the first parameter to the script (line numbers have been included only to aid in the discussion) :

EXAMPLE SCRIPT #2

Getting the Most out of HP-UX

```

#
# This script upshifts an entire line from stdin
# when the first parameter occurs within that line.
# filename: example2
#
1 while read line
2 do
3   CHECK=`echo $line | grep $1 `
4   if [ "$CHECK" != "" ]
5   then
6     line=`echo $line | tr "[:lower:]" "[:upper:]"`
7   fi
8   echo $line
9 done

```

line 1: the while here reads from stdin into the variable line until end-of-file (or a line containing only a <RETURN> is entered). Lines 2 and 9 define the statements to be performed inside the while loop.

line 3: this line sets the value of the shell variable CHECK to the value of the first parameter (if that string is present in the shell variable 'line') or to the empty string (if the first parameters' value is not present within the value of line). This is done in order to determine if the line should be upshifted or not.

line 4: this IF statement translates the entire string from lower to upper case if the shell variable CHECK is not empty.

line 6: this statement pipes the current string (in the variable \$line) through the 'tr' command which (using the given parameters) will perform the upshifting and then re-assign this new value to the shell variable 'line'.

line 8: this line simply writes each input line (whether or not it has been upshifted) to stdout.

This script can then be used to capitalize only certain lines within a file, based upon their content. To make this script read from the file '/tmp/article_data', upshift any lines containing 'HP-UX', and place the output into the file called hp-ux_data in my login directory, I would invoke it as follows:

```
example2 HP-UX < /tmp/article_data > $HOME/hp-ux_data
```

In this way, I can use parameters and redirection to control both how the script will behave (which lines it will capitalize) and where the input comes from (and where the output is sent) respectively. I believe these concepts are fundamental in the quest to harness the power of the shell.

Now that we understand the basics, but before we go on the case study, there is one more programming concept that will allow us to modularize our shell programs so they are much more powerful, and also easier to read and understand. This is the concept of functions (or procedures), a common one in many languages. The Korn shell allows us to define functions using the following syntax:

```
my_function_name()
{
# This would be the body of the function
# Each function can access any shell variables
# within its' scope, as well as any parameters
# provided when it is invoked.
echo " This is my_function_name "
}
```

Korn shell scripts are simply ASCII files that contain comments, indicated by a pound sign '#', and shell commands, which are just about anything else. The different styles of braces in this example function are both significant and required. To call a function that has been defined within a shell script, one needs only to type its name (just like any other command), and in fact, the function IS a (user-defined) command in the

shell. In order to pass parameters to a Korn shell function, we just place any parameters after the name of the function, and refer to them inside the function as \$1, \$2 and so on. If we want our function to pass data back however, we must use shell variables rather than parameters.

So, to summarize, we now have a programming environment that has flexible I/O facilities (pipes and redirection), some of the normal control constructs (like if, case, for, while), and subroutine calls in the form of functions. There is an incredible array of string and file processing routines available with HP-UX, and we will look briefly at some of the facilities available. For a more in-depth treatment of these facilities, see references [3] and [4].

Case Study - CDB

This case deals with one of the results of a company-wide consolidation. After 18 data centers were "moved" into a single facility, the computer operations group found that their job abort process was not adequate when literally thousands of jobs were involved. They simply could not keep track of the shifting support assignments and the myriad of notification requests. So, a project was begun to put all the batch jobs, machine names (over 100), support people and their phone numbers into a database that operations could use when jobs aborted. While that was underway, a quick and dirty interim solution was also begun. This paper is the result of the "interim" solution, which has helped HP3000 computer operators in Atlanta for over 2 years now.

It turned out that what the operators really wanted was a quick way to find the right phone number to call, given a particular job. They had the names of the jobs that aborted from an abort printer, and they were responsible for notifying the proper support person. All the operators would need simultaneous access, and there would, of course, need to be a way to make the

inevitable moves, adds, and other changes when the need arose.

I was new in my job and had been experimenting with shell scripts a little. I had recently implemented (yes, just for grins) a 'phonebook' utility that had been provided as an example in a textbook, and it seemed to provide for the basic elements (data file, lookup and modify mechanisms) required for a solution to the job-abort problem. I had recently modified this example to make use of functions. It seemed to me that a shell script could provide a serviceable solution to this type of problem. There was also the significant advantage of already having a working prototype in the form of the example I had already tested.

And so began the project known as 'cdb' (for Contacts Data Base) in June of 1992; which, by its' unfortunate moniker, precluded the easy use of one of the HP-UX C language debuggers.

The requirements actually called for two databases, one for names and phone numbers (like the one I already had), and another one for holding jobnames, machine names, peoples names, and other things like whether or not they preferred to be called in the middle of the night if their job aborted.

What I wanted to be able to do was search an ASCII file for one or two strings (job and system), and then match phone numbers with all the names associated with any qualifying records. That is, once I had the people responsible for the aborted jobs, find and display their phone numbers.

So, I began with a script that displayed a menu, asking whether you would like to look someone up in the database, add or change something, or exit. It also had defined functions for lookup, add, remove, change, display and listall. Let's start with the lookup() function after some modifications :

```

#
# lookup function - checks for both JOB and MACHINE,
#                   and displays all 3 phone numbers.
#
1 lookup()
2 {
3 echo " Enter the job that aborted : \c"
4 read JOB
5 echo " Enter the machine name : \c"
6 read MACHINE
7 grep $JOB $ABORTBOOK | grep $MACHINE > /tmp/recs$$
8
9 while read line
10 do
11     echo $line
12     NAME1=` echo $line | cut -f 4 `
13     NAME2=` echo $line | cut -f 6 `
14     NAME3=` echo $line | cut -f 7 `
15     display_phones $NAME1 $NAME2 $NAME3
16 done < /tmp/recs$$
17 rm /tmp/recs$$
18 }

```

This lookup function first prompts the user for which job and machine combination to find. These values are read into the shell variables JOB and MACHINE, respectively. Line 7 filters the qualifying records from the database and places them in a file called '/tmp/recs\$\$'. The UNIX grep command is used to accomplish this. This blend of two grep commands is, for now, our search mechanism for finding records that contain both the proper job and machine name.

The \$\$ metacharacters used as part of the temporary file name are replaced with the process id number of the current process. This allows more than one lookup to proceed simultaneously without interference. The while loop on line 9 reads through the qualifying records (specified on line 16 as redirected input to the loop), echoes them out, and calls display_phones (another function) once per record to show the phone numbers for the three contacts. The statements on lines

12 through 14 extract fields four, six and seven from the record in `$line` and use these as contact name parameters to the function `display_phones`. The `rm` command on line 17 cleans up the temporary file after processing is complete.

This provides essentially what the operators needed, it lists phone numbers when given a job and machine name. Now that we have lookups working, we must also provide for adding, modifying and deleting records from either the phonebook or abortbook files. Towards that end, let's look at a minimal `add()` function that will work for both files:

```
#
# add function - real simple right now
# just pass the filename to be modified as
# parameter one.
#
1     add()
2     {
3     echo " Enter the record here : "
4     read LINE
5     echo "$LINE" >> $1
6     sort -o $1 $1
7     echo "$LINE has been added to the database "
8     }
```

Here, line 1 begins the syntax for defining the `ksh` function `add`. Everything within the curly braces is the function definition. Lines 3 and 4 prompt the user for input and place that input into a shell variable called `LINE`. Line 5 appends the new data to the database (whose name was passed in `$1`), and line 6 ensures that everything is in the proper sorted order. Confirmation is returned by line 7.

At this point, we have the skeleton of an application that fits the needs of the operators, can be used from almost any HP-UX platform, and only took about a week to write, test, load with data, and debug. That sounds like rapid application development to me.

Access Methods for CDB

Now that all the operators can get phone numbers immediately after a job abort, the next task is to ensure that the proper people can keep this database accurate. To do this means that we must provide all the application support people easy access to our 'abortbook' database. This was accomplished by using NFS to mount the cdb directory onto all the servers used by the application support groups.

To make this work, not only did we have to allow all these people write access to the database, but some method of concurrency control was required. This results from the fact that when two different people want to make changes to the same file, something must be done to keep the second persons' changes from overwriting the changes made by the first. To do this, a simple locking mechanism was introduced that followed this simple algorithm:

```
1   if [ -r $lockfile ]
2   then
3     cat $lockfile
4     exit
5   else
6     date; who am i > $lockfile
7     < OK to MAKE CHANGES to DATABASE HERE >
8     rm $lockfile
9   fi
```

Here, line 1 checks to see if someone has already locked the database. If so, we print the contents of the lockfile (to show who has it locked) and then exit. Otherwise, we create a new lockfile (that shows who we are and when we locked it), and proceed to make our changes. When we are finished, we must remove the lockfile to allow the next person to make updates. While this scheme is simple at best, it does prevent changes getting lost (as long as all edits are done using this script). This algorithm does NOT provide for

concurrent write access, but we were able to live with that since, on average, there was only a single change made per day.

Other Business Controls and Life Cycle

Ultimately, we wound up with two scripts for the abort process. One used by operations to perform lookups, print the entire database, and change phone numbers when needed. The second used by the application support groups to make moves, adds and changes to the abortbook file. The first script has now grown to about seven pages, and the second (requiring more complexity) is almost thirteen pages long. Most of this growth was due to the addition of business controls during the extended life cycle of this temporary solution.

Those business controls included implementing a log file so that changes could be tracked, adding passwords to some of the functions like changing the phonebook file, and making the update script much easier to use. The search mechanism had to be more selective (it is now a 23 line awk program, instead of the original two grep statements) and the modify and add functions also underwent significant enhancements. To date, we have added the use of softkeys instead of the original menu, a GUI-based update module for formatting the data from moves, adds and changes, and provided for the use of 'wildcards' both in the abortbook records and when searching for a jobname. Also, the operators no longer have to type in a jobname and machine name, we have automated the process so that the names, phone numbers, and any job abort instructions print out as part of the job abort sheet.

Overall, implementing and maintaining this application has been relatively simple, since the code-test-debug cycle has been shortened significantly by using Korn shell scripts. In general, I was able to implement most of the requested changes by the next day ! This has resulted in not only greatly streamlining the job abort process for the operations and application

support groups, but also in an up to date phone listing that can be used easily by others.

=====
References

- [1] "HP Interface Architect 2.0 Developer's Guide", HP part number B1164-90003, Third Edition, April 1992.
- [2] "Using HP-UX", HP part number B2910-90001, First Edition, August 1992.
- [3] "UNIX Shell Programming", by Stephen G. Kochan and Patrick H. Wood, Howard W. Sams & Co., Indianapolis IN, 1990.
- [4] "Advanced UNIX - A Programmer's Guide", by Stephen Prata, Howard W. Sams & Co., Indianapolis IN, 1989.
- [5] "Using HP-UX", HP part number B2910-90001, 1st Edition, 1992.
- [6] "UNIX Networking", by Stephen G. Kochan and Patrick H. Wood, Hayden Books, Carmel IN, 1989.

Paper Number 1016
**Visual Basic for the MIS Professional: How to Develop High
Performance Client-Server Applications**

Terry O'Brien
Executive Vice President
DISC
5733 Central Avenue
Boulder, CO 80301
303.444.4000

Handouts will be provided at time of presentation

**THE H.P. INFORMATION SYSTEMS MARKETPLACE TODAY:
WHAT'S HOT AND WHAT'S NOT!**

**Diane Amos, CPC
Amos & Associates, Inc.
633-B Chapel Hill Road
Burlington, NC 27215
910-222-0231**

In today's fast paced, rapidly changing Information Systems market, the lines of technology are blurring. "Open Systems" may literally mean opening the world of technology for companies to meet their needs in a better way.

How does this impact a company's ability to staff its' I.S. department? How can an I.S. professional stay on top of this wave of technology and not be swept out to sea? This presentation will explore the challenges presented by the many changes in Information Systems today.

The biggest request that I receive is for information relating to "what's hot" in the technical arena. I.S. professionals want to stay current with technology so that their careers can progress. From an employment perspective, let's look at what's HOT and what's NOT!

Let me first offer a disclaimer. I am not a technology expert. The information that I'm sharing is what I hear from the unique view that I have as a recruiter. This is information that I have gathered from managers who I conduct searches for and candidates who are seeking new jobs.

I. Skills in Demand...

- A. Relational databases are the way of the future for the Hewlett Packard marketplace and everyone is jumping on this wagon. Companies that develop these relational databases are well represented here at this

**The H.P. Information Systems Marketplace Today:
What's Hot and What's Not!**

conference in the exhibit hall which offers a wonderful opportunity to compare the various vendors. From a recruiting standpoint, we've seen the heaviest requests for people with skills on the following relational databases: Sybase/Oracle/Allbase/Informix/Progress.

- B. Client-server tools such as Powerbuilder and Uniface are also in demand. Interestingly enough, everyone is talking about "client-server," but few companies can really define it. It's like arguing over what is truly a relational database. Everyone has a different opinion about "client-server," as evidenced by the many different talks at the last INTEREX show. But more importantly, few companies are really "there" yet. This causes a real shortage of experienced talent with those skills that are in demand.
- C. PC's are springing up everywhere! With the downsizing of mainframes, more companies are turning to PC's and need highly qualified PC talent. I don't mean computer people who are PC "users." The demand for PC talent is in several different categories. We are seeing an increased demand for Windows' developers. Office automation specialists who can train users on the multitude of PC software are requested frequently. A PC specialist is often times expected to get into the guts of the hardware and diagnose or repair it, especially in small shops. And finally, we see a major demand for those individuals who can hook PC's together into networks. Which brings me to the next skill area in demand...
- D. Networking! Access to information quickly is one of the most vital ingredients to having a competitive edge in today's marketplace. Networking computers together across the world through WAN's, throughout remote offices with LAN's is what will change the future. Networking expertise is definitely in demand with such products as

**The H.P. Information Systems Marketplace Today:
What's Hot and What's Not!**

Novell/Netware leading the way. A "certified network engineer" -CNE is the most in demand and can therefore expect the higher wage.

- E. With Hewlett Packard selling UNIX boxes like hot-cakes (60% of their market), obviously HP-UX operating systems skills are in demand. The staffing problem that can be created by this is that MPE customers who are migrating to UNIX would like to hire someone with both MPE and HP-UX skills. Those few employees that have experience on both operating systems certainly have the edge. But remember, companies on MPE that have been running their business applications on MPE expect to continue to do this on UNIX. They will not hire someone with just UNIX technical skills. They need the business acumen as well.
- F. Systems integration is the hot buzz word now. Those candidates that have multi-platform experience and a know-how to integrate the various systems in a company will have an edge. With open systems technology today, companies may have multiple databases, hardware, and operating systems. Someone who knows how to integrate this all seamlessly will be worth their weight in gold. With HP making inroads into the IBM and VAX markets, many companies are having to migrate from VMS and MVS to HP-UX. Conversion experience is also highly valued.
- G. 4GL's continue to be popular with a demand for the standbys Powerhouse and Speedware but we also see new kids on the block in the HP market: Progress, Powerbuilder and Uniface. In the MPE world, few companies are solely using 4GL's, therefore, there is usually a need for COBOL skills in addition to the 4GL background. C and C++ languages are strong in the UNIX world, which leads us to object oriented technology....

**The H.P. Information Systems Marketplace Today:
What's Hot and What's Not!**

- H. We're already being asked to provide candidates with experience in object oriented programming and design using related products such as Smalltalk. The supply has not yet reached the demand, and I predict the demand will be increasing.
- I. As more companies look to streamline, I see more of them turning to 3rd party application software rather than custom writing their own systems. In the manufacturing arena, ASK is a big draw with some requests for AMAPS and WDS as well. With HP making inroads into the mail order business, several mail order packages are popular. Accounting software is quite prevalent with many choices to be made. For that reason, typically the request is for "some" type of accounting package experience, but not requiring a specific package. The manufacturing industry is the most demanding and specific industry for requiring exact matches on the manufacturing software that is in use.
- J. Surprisingly, there are still companies that are seeking what I call the "vanilla" programmer -those with skills in COBOL/IMAGE/VPLUS. They don't care about 4GL's or relational databases, and run just fine the way they are, thank you. For those of you who enjoy this environment, there is still a place for you. But for how long?

II. Skills not in demand...

- A. HP1000 professionals are having a hard time of it. Those that have not had the opportunity to migrate to HP9000 are stuck in careers and shops that are either going to stay that way or disappear. I haven't seen a request for an HP1000 candidate in 3 years and I have a drawer full of candidates looking for a way out of that market. My only suggestion is to get training fast on other hardware.

**The H.P. Information Systems Marketplace Today:
What's Hot and What's Not!**

- B. Anyone who is just a coder or just an analyst is not in demand. A programmer today is expected to be able to do some analysis, and an analyst is also expected to program. The days of the pure coder cranking out code in the back room are gone. There is also little demand for the pure analyst who designs systems for users without seeing their system through to implementation.
- C. Techies with no interest in business will have a hard time in the marketplace of the 90's. Today, an I.S. professional needs to have business acumen. They need to see the big picture and the role they play in it. They also need to be effective communicators with the end users and understand the business issues that the users are trying to solve.
- D. Certain industries have taken a downturn and I.S. professionals with narrow industry experience may find themselves boxed in a corner. A good example of this is the programmer with defense and aerospace industry experience. With the cuts in defense, those jobs are going away and the experience doesn't translate well to private industries that want business systems experience such as manufacturing or accounting.

III. Soft Skills

Technical skills can be learned. But today more companies are putting emphasis on what I call "soft skills" over technical skills. Their feeling is that "soft skills" are more inherent and not easily taught on the job.

What are these "soft" skills?

They are areas such as:

- customer service orientation
- people skills
- business acumen/savvy
- effective listening and communication skills

**The H.P. Information Systems Marketplace Today:
What's Hot and What's Not!**

2000-5

Let's break this down so we're clear in what we mean.

- A. "Customer service orientation" means an attitude that says give the customer what they want. It's an understanding that Information Technology truly exists to serve the user. An I.S. professional with this expertise can work with a user to clearly understand their needs and translate them into a workable solution. This kind of I.S. professional generally has a lot of patience with the non-technical user, and can talk at their level adapting as necessary. With I.S. becoming more user friendly and users taking a more active role in the integration of business and I.S., this skill is critical to companies in the 90's. Practically every search that I have, discusses the importance of this skill. Gone are the days of separation between user and I.S. professional and the ivory tower has disappeared. Adapt, learn how to do this, or your career may falter.
- B. "People skills" can mean many things and the vagueness troubles those who try to define it. What I understand from my clients who request this is that they want to hire people who get along with other people, are outgoing, dynamic and aggressive, expressive rather than analytical. This area may bode trouble for the typical computer professional who is the "analytical" and "introverted" type. Skills that need to be developed are empathy for the other person, open-mindedness and flexibility for other points of view, an ability to reach out for understanding rather than standing ground in conflict. I know several computer professionals who are analytical and introverted who have learned, that to progress in their career they needed to develop this side of their personality. It can be done and is another skill that companies are stressing in the 90's.

**The H.P. Information Systems Marketplace Today:
What's Hot and What's Not!**

2000-6

- C. "Business acumen" means an ability to see the "big picture" of what a company is trying to accomplish, it's mission, goals and strategies and not just their part in the play. Many I.S. professionals have not been allowed to play a part in business decisions before now, but with companies embracing the "empowerment" concept, they not only want to give this decision making power to their employees but they expect their employees to handle empowerment successfully. I.S. departments are now often on equal footing with other departments in a company and have the President's ear on how to help run the company. A technical person who only cares about the bits and bytes of technology will not be asked to assume this responsibility. This responsibility is not just at the top either - it is filtering down through the ranks with each I.S. professional playing a part. Learning how what you do that impacts the work of others and how it fits into the overall business scheme is crucial. ASK those critical questions - this is how you can learn this "soft" skill.
- D. "Effective listening skills" will not only get you that desired job but help you to be successful in it. This is occasionally a weak area for I.S. professionals while interviewing. They are concentrating so hard on their answers that they don't always listen carefully to the question to make sure they understand it. The same goes for daily communication with peers and users. The best user reference I can get is when they tell me that my candidate took the time to listen to their needs so that they really understood. The trick I believe, is to concentrate on the speaker, repeat your understanding of their statement or question, and through continued feedback, make sure that you are hearing what they are saying. It's an easy procedure and with practice becomes almost commonplace. Feedback is the KEY.

IV. Future trends

People ask me every year what the future holds. This is a fun aspect of my job, and this is my time to play Jeanne Dixon. Sometimes I'm right on the money and sometimes I'm not.

**The H.P. Information Systems Marketplace Today:
What's Hot and What's Not!**

2000-7

Overall, I see a continuance of companies running lean and mean. Because of this, companies are asking for almost perfect matches in both skills and personality.

-I believe that HP-UX will be a major player in the future, however the MPE shops that choose to remain MPE also will do well.

-The most in-demand job will continue to be the 3-5 year application developer.

-Opportunities for managers will continue to diminish due to rightsizing and reengineering. - Object oriented technology should play a major role in the future as well as client-server, networking technology.

Technology will not just be in the workplace, it will be in our homes and play an increasing role in our lives. We're all hearing about the "information highway," which we need to follow closely. Because of these future trends, I.S. professionals need to be open, flexible and adaptable to learning new technology. Those who do not may find themselves out of a job.

V. Importance of training...

With the demand for newer skills, the need for training has never been more important. Sadly, this is where most companies drop the ball. They expect to achieve their goals without investing in the training to get them there. So often, if they would just send their eager, talented people off to be trained on the "latest-greatest" rather than upset their internal salary structure by hiring the outside "guru," they would see their goals achieved and create a loyal staff besides. If they have to hire that GURU, they should utilize the GURU to train the others. Training is the KEY! The keynote speaker at San Francisco's INTEREX stated that a manager explained that he was fearful to train his staff because they might leave his company to better themselves. The speaker's reply was, .. "and what will happen to your company if you don't train them?"

If you need some justification to add to your training budget, try using these statistics to convince management. (Taken from the American Society for Training and Development).

**The H.P. Information Systems Marketplace Today:
What's Hot and What's Not!**

2000-8

1. Over the past 50 years, investments in job-related training have consistently outperformed investments in machine capital. A trained workforce produces more productivity gains than capital investments do.
2. The average 1.4% of payroll that U.S. companies invest in training reaches 10% of the workforce - Japanese and European-owned companies based in the U.S. spend three to five times more on employee training than American companies do.
3. By the year 2000, 75% of all workers currently employed will need retraining - technological and societal changes are reshaping jobs in every industry.
4. 70% of the people who will be working in the year 2000 are already in the workforce - the only way they will have access to training is if companies like yours supply it.
5. The surplus labor supply of a few years ago has dwindled to almost nothing - companies won't be able to buy talent, instead they will need to create it with training.

VI. Issues for companies...

A word of caution is needed in the midst of this frantic rush toward newer technology. Take a good look at your company's needs and how you are able to meet them. If your system does a great job in meeting those needs, is it smart to jump on the bandwagon just because other are? I'm seeing many companies take the leap without looking where they're going, and then having to backtrack because they impulsively jumped into a technology that they weren't prepared to handle or that didn't truly meet their needs. There has to be a balance between a company's needs and new technology.

**The H.P. Information Systems Marketplace Today:
What's Hot and What's Not!
2000-9**

On the other side of the coin, getting behind in technology may cause you to lose your staff. Think through your strategies and objectives carefully to weigh out where technology fits into your company's goals and mission.

VII. Issues for candidates...

Look at how you can become a well-rounded business professional, not just a technically competent computer individual. Remember my comments on companies hiring "business" oriented people who can communicate effectively. Develop your "soft" skills! If you are a technically brilliant I.S. professional, but have some difficulty in relating with end users and your peers, there is help out there for you. The world is loaded with training workshops on "developing better listening skills", "dealing with difficult people", etc. If you can't take the time off of work to attend, there are numerous books and tapes that you can study in your spare time at home. I believe that "soft skills" can be learned, so embark on your own self improvement program. Remember, they hire you 75% because they like you and 25% because they think you can do the job.

Flexibility and adaptability to changing technology will be critical for the future of your career. Those of you that want to stay with the "familiar" may lose out as everyone else moves ahead. Be willing to learn new things, get trained and adapt to change.

In summary, we've discussed the technical skills that are in demand and not in demand, explored the area of "soft skills," looked into our crystal ball for future trends, and recognized the importance of training. If I could give you one piece of advice from this presentation that I believe will help you the most, it's...develop your "soft skills."

**The H.P. Information Systems Marketplace Today:
What's Hot and What's Not!**

2000-10

Re-Examining Re-Engineering

A Guideline for Competitive and Strategic Information Management in the '90s

by

Feyzi Fatehi

*Hewlett-Packard Company
General Systems Division
19111 Pruneridge Ave.
Cupertino, California 95014
Tel: (408)-447-4567*

ABSTRACT

During the past 20 years, the industrialized world has been making the transition from an "industrial economy" to an "information economy," making information rather than land, labor, or capital the basis for creation of wealth and prosperity. Nonetheless, few organizations have demonstrated that they have integrated this knowledge into their strategies in any substantive way.

A possible explanation behind this discrepancy is the fact that using information as a strategic resource, and building structures and processes that will support this focus, is not an activity that lends itself to a simple, mechanistic, or schematic approach. It is clear that traditional information systems modeling approaches are not sufficient for modeling the true complexity and richness of an organization's information assets and environment.

In this paper, the author shares his collective experience gained from working with several major corporations that have been involved in the process of re-engineering their information systems. The paper, while presenting a fresh look on the subject, sorts through the confusion surrounding the relationship between information technology and both productivity and competitiveness.

It is in this context that the reader's attention is focused on differentiating Information Technology from Information itself, and articulating the critical role of strategic information management, leading to a sustainable competitive advantage. The paper also re-examines the inter-relationships and the mutual influence of the information technology on one hand, and the organizational structures, business goals, and management philosophy on the other.

Introduction

About two years ago, the author completed and presented a paper titled: "From Downsizing to Rightsizing"[1]. The main purpose of that paper was to present the reader with an alternative perspective on the subject. The author considers this paper a logical sequel to that one for two reasons. First, it attempts to accomplish the same goal of presenting a fresh perspective on a contemporary subject. Second, this paper attempts to take a step beyond "information technology" and direct, if not divert, the attention of the reader to the strategic role of the I in IT. The paper attempts to clearly demonstrate that the majority of the re-engineering efforts are either *technology-centric* or *process-centric* both of which address the underlying "efficiency" of an organization and proposes a new approach based on *information-centric re-engineering* focused on enhancing the "effectiveness" of an organization.[2]

The Two Commonly Used Approaches to Re-Engineering

A survey of the on-going major re-engineering efforts indicates that most are based on one of the two commonly practiced approaches to re-engineering: technology-centric or process-centric, or various combinations of both. Before exploring any alternatives, let's take a quick look at an overview of each approach.

1) Technology-Centric Re-Engineering

Several forces have created the demand for technology-centric re-engineering during the past few years. The most important of these is the availability of a new generation of enabling and complementary technologies such as:

- Local area networks (LAN)
- Graphical user interface (GUI)
- RISC-based high-performance, cost-effective servers
- Expandable symmetric multiprocessing (SMP) servers
- Gateways between LANs and wide area networks (WAN)
- Network operating systems (NOS).

These individual technologies are synergistically brought together by the emergence of several architectural driving forces, most importantly the client/server architecture to either completely replace the existing "legacy" systems, or in some form co-exist with them. In most cases the result of technology-centric re-engineering efforts can be summarized as reduced *size* of computer processors, oftentimes leading to a reduced *cost* of computer processing.

However, the technology-centric re-engineering efforts have often fallen short of the expectations of business managers.[3] What has been the basis for this shortfall? Basically, business management often cannot obtain *integrated*

information about their business from the information systems they are using. Generally speaking, the world of information systems development is characterized by detail centered around aspects of technology such as functionality, hardware, software, and network configuration details. It is not surprising, then, if the business side is neglected. The main point of using technology to improve the competitive position of the business is usually lost by the emphasis on the technical development issues - on efficiency and the *how*, rather than effectiveness and the *why*. [4]

2) Process-Centric Re-Engineering

Process-centric re-engineering efforts can be divided into two separate but related categories: Business Process Re-engineering (BPR) and Business Process Automation (BPA). BPA is the automation of manual business processes through the use of new technology now available. BPR, on the other hand, is the reshaping of key, core business processes, regardless of whether they have been or will be automated to reflect contemporary, optimized models of doing business. The two go hand in hand because both strive to achieve a common objective: to maximize the *efficiency* of the organization's business processes.

In his 1990 Harvard Business Review article that provided the impetus for much of the discussion on BPR, Michael Hammer wrote of a dramatic change in the handling of accounts payable at Ford, which could be used to effectively illustrate this concept. To avoid the full description of the background scenario, it suffices to say that the used process was lengthy and required hundreds of people to complete the steps. When Ford looked at a competitor, Mazda, a very different process was discovered. They found that less than a dozen people handle the process at Mazda. This provided Ford with a 'benchmark' for time and resources. To achieve this level of efficiency, Ford had to redesign their outdated process (BPR) as well as introduce two new technologies, EDI and barcode readers (BPA) to eliminate most of the cumbersome paperwork. In the new process the traditional purchase orders, invoices, and requests for payment were substituted with electronic transactions that handled the full process with the supplier from order to payment. In business terms, the process was 're-engineered' with increased time and cost efficiencies.

Developing an Information-Centric Perspective

Business decisions may involve choices affecting large-scale investments, strategic directions, and critical organizational goals in which the CEO or COO would be involved. Business decisions can also involve somewhat simpler choices regarding issues related to productions, administration, or any other department. These decisions are handled daily and hourly by line workers, supervisors, specialists, and managers.

The variety of these decisions demonstrates that businesses depend on information at all levels for both daily operations and longer-term strategies. When decisions, specially strategic decisions, are made by information-leveraged individuals the decisions are more effective.[5] If these decisions are implemented properly in subsequent actions, the organization is better positioned to achieve its strategic and competitive goals. This is the basic, underlying premise of an *information-centric re-engineering*--that timely, relevant, high-quality information is crucial to business success, that its critical importance must be recognized, and that information resources must be managed as strategic assets.

However, before we focus more closely on the topic of information-centric re-engineering, it is nessecary to set the stage by briefly exploring an emerging trend in the economic base of competitive landscape. Moreover, we will evaluate the critical role of information in this new economic paradigm.

A Shift in the Economic Base of Competition

Thinking about the nature of competitive strategy, one cannot simply ignore the competitive base of the economy at large. Most importantly, specific attention should be paid to the emerging shifts in the economic patterns.

With the advent and deployment of such concepts as "electronic commerce" and "information super-highways," the emergence of an *information economy* has already acheived the status of common wisdom.

Transitioning from an "industrial economy" to an "information economy" makes information rather than land, labor, or capital the basis for creation of wealth and prosperity. In other words, in this kind of economy, it is what you know, not what you own, that determines success.

In an information economy, businesses compete on the basis of their ability to acquire, manipulate, interpret, and use information effectively. Businesses that master the information-based competition will be the big winners in the future, while the organizations that don't will quickly lose their competitive advantage.

During the last two decades, General Motors learned that controlling the capital base of the largest industrial corporation in the world provided little advantage in the face of organizations that were better informed about shifting customer demands, and better informed about how to organize business and manufacturing processes to serve these demands.

Information-Enabled Vs. Information Technology-Enabled Competition

Information-enabled competition is not synonymous with investments in information technology. General Motors used its massive capital base to make the

single largest investment in information technology in history when it acquired Electronic Data Systems. As the consequent results suggest, that did not make them information-enabled competitors.

The value of information technology depends on information and the role of information in organizations. Information technology investments create no more advantage or productivity, by themselves, than do investments in new machines tools. It is not technology, but effective use of technology that creates value. Information technology is an important *enabler* of better information use, but it can just as easily be useless without the acquisition, manipulation, interpretation, and effective use of relevant, timely, and dynamic information.

Information must Inform!

Before we can continue and expand our discussion to the realm of information-centric re-engineering, we need to fully understand what information is. Information is not just data collected; rather, it is data collected, organized, ordered, and associated with context and meaning. Information must inform, while data has no such mandate. Information must be bounded, while data can be limitless. Information is data minus noise. In order for data to become information, it must become useful to a decision maker and must be presented in such a way that the user can relate to it and act upon it. Information is not free, nor does it flow freely. In an information economy information is a form of wealth, and therefore is a source of power.

Information about the current organization and competitive environment helps managers identify both threats to and opportunities for the company, and sets the stage for the design of a more effective competitive response. Information also functions as a critical resource for the design and evaluation of strategy alternatives. The feedback of performance information is also central to the creation of an adaptive and learning organization that detects and recognizes the early warning signals indicating the need to modify previously set strategies and objectives when they start becoming ineffective.

Structured Vs. Unstructured Business Information

Since the term *decision support systems* (DSS) was coined in the early 70's, organizations have started distinguishing between the bulk of structured data processing activities, such as recording sales orders or managing payroll, and the use of information and information technology to support the less structured tasks of managers. In general, the structured information is mostly associated with operations and both inbound and outbound logistics of an organization, while the less structured information--which is often diffused and qualitative--is the target of executive management in charge of strategy, as well as marketing and sales organizations.[6]

An information-centric approach to re-engineering not only pays close attention to the less structured information needs of an organization, but also provides sufficient means for extracting information out of the bulk of available structured data and *integrating* it with the available unstructured information to maximize the effectiveness of an information-enabled decision-making process.

Sources, Resources, and Users of Organizational Information

Before we further continue this discussion, it is helpful to identify the sources, resources, and users of organizational information.[See Figure 1]

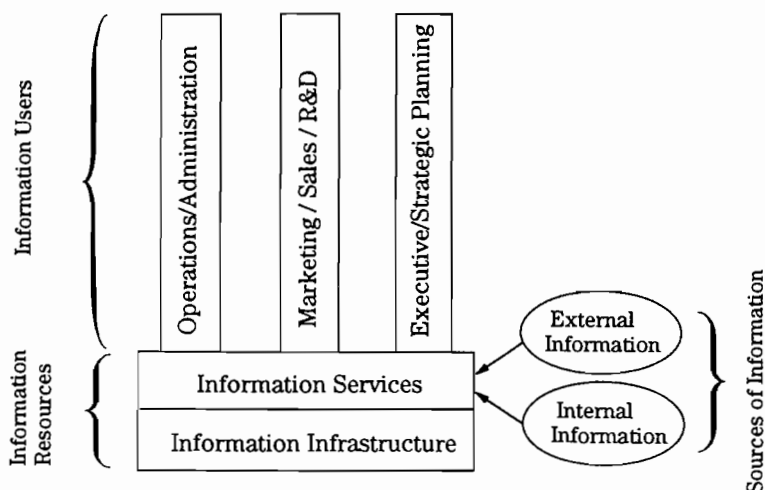


Figure 1. Sources, Resources, and Users of Organizational Information

Information *sources* can be grouped into two main types:

- Internal Sources. Information originated from within the organization, such geographic breakdown of sales information or profitability of a specific product line.
- External Sources. Information originated from outside the organization, such as information about the economic environment or competitive product information.

Information *resources* can also be grouped into two major categories:

- Information Infrastructure. The hardware, software, facilities, staff, and other elements that support the delivery of information services.
- Information Services. The computer applications for the processing, storage, retrieval, transmission, and presentation of information, as well as the services rendered by various Information Resource Centers.

Information *users*, on the other hand, can be generally categorized into three logical groups, each with a distinct set of informational requirement:

- Operations and Administration. This group of users are responsible for planning and implementing the operational processes of an organization. They mostly rely on sets of structured information to carry out their basically structured tasks such as accounts receivable or inventory management.
- Marketing, R&D, and Sales. This group is mostly responsible for tactical plans, requiring actions that satisfy business performance goals, including those that implement strategy. They rely heavily on less structured information, require a significant amount of analysis, and draw heavily from external sources, in addition to using internal sources of information.
- Executive Management. This group is responsible for more strategic decisions, setting the general tone and direction of business. Therefore, they require more integrated view of internal information and rely heavily on external information, such as competitive posturing of major competitors as well as domestic and global environment of business.

A Road Map to Information-Centric Re-Engineering

Strategic and competitive information management requires a dynamic and systematic *process* that is well understood and supported by the key decision makers at all levels of the organization. It starts with the strategic business plan of the organization and results in an information-enabled organization. As illustrated in Figure 2, here are the recommended steps:

- 1) Develop a strategic business plan
- 2) Identify critical information needs
- 3) Develop an enterprise information map
- 4) Conduct information inventory audit
- 5) Identify information availability and *information gaps*
- 6) Define required information products and services

- 7) Re-architect and re-engineer (or engineer in the first palce) information resources to enable the required integrated information products and services.
- 8) Provide customized view and access to each group of information users.
- 9) Leverage the competitive advantage of an information-enabled organization!
- 10) Revisit the entire process with each subsequent cycle of strategic planning.

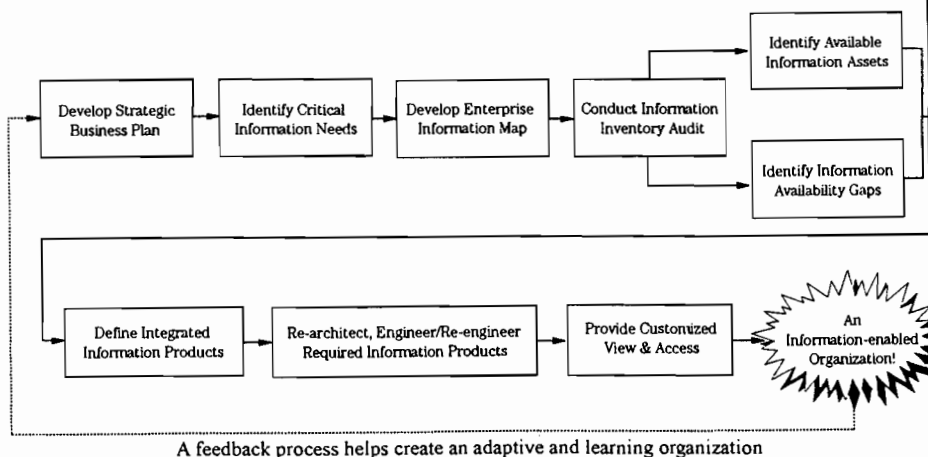


Figure 2. A Road-map for Information-centric Re-engineering

Discussing the development of a strategic business plan is outside the immediate focus of this paper, however, it is important to further explain the identification of information needs, the development of an information map, as well as a few other steps.

Identifying Critical Information Needs

For each main business goal identified in the business plan, the business managers must identify the critical information needed to support the goal. For example, the first-cut, high-level, critical information requirements for a marketing organization in a highly competitive industry might be as follows:

- Detailed reporting on market penetration for the organization and major competitors worldwide
- Graphic presentations of market shares and market trends

- Quantitative and qualitative analyses of shifting customer perceptions and preferences

For the information needs to be well understood, however, these first cut requirements must be expanded into more detailed descriptions.

People Don't Know What They Don't Know

Interviewing managers and key decision makers provide an excellent mechanism for compiling the critical information needs of an organization. However, when interviewing individuals to discern their "true" information needs and requirements, the interviewer could be faced with silence or a baffled stare. Some respondents can't answer many questions concerning their information needs because they haven't a clue whether such information exists, either within or outside the company; or if it exists, whether it can be obtained, put into a system, or delivered in a timely manner.

To resolve this issue, information professionals conducting the interviews need to be conversant with the available sources of information that could be of value to the client or client organization before doing the interview. It is also valuable when discussing information needs to show actual examples, models, or prototypes of information sources already designed.[7]

Organizational Information Maps

The purpose of information map is to help managers as well as others in the organization comprehend the organizations information assets, sources, and users of these information assets, as well as how they relate to the goals and functioning of the organization. Information maps depict the relationships of information sources at various functional levels in the organization and even at a detailed level can be used to answer such questions as what systems or databases are sources of information or services to the various business units. As the metaphor *map* indicates, maps should be as graphic as possible, showing how information relates organizationally, geographically, and operationally, as well as how it relates to the planning and management process.

If automated, information maps can be stored in a repository with online access and can depict the organization's information sources, resources, and users at several levels: high-level (i.e. strategic), mid-level (i.e. tactical), and detail level (i.e. operational). Upper CASE tools that are encyclopedia or repository based probably offer the best technology for automating the information maps. The maps should be updated with each subsequent cycle of strategic planning and information inventory auditing, if not more frequently.[8]

Conducting Information Inventory Audits

An information audit is an assessment of the assets and liabilities of the existing organizational information scenario. To maintain an information-centric focus, the auditors should rely on the information maps and stay focused on the task of *information* auditing rather than getting involved in the information technology arena. The results of this audit should be used to determine the *information gaps*, as well as available and reusable information assets of the organization.

The Architecture Metaphor

The only two items in the process that have something to do with technology are items 7 and 8. Delineating possibilities for information-technology implementation aspects of this process has its own body of knowledge that is clearly outside the immediate realm of this paper. The reader can easily benefit from the wealth of literature and case studies available in the marketplace. However, before we move on, it is important to fundamentally distinguish between the two concepts of architecture and engineering.

James McGee and Laurence Prusak of The Ernst & Young Center for Information Technology and Strategy present an eloquent differentiation: "Information technology architecture is simply more than just information technology engineering in the same way that architecture is more than just mechanical engineering. Engineering is about *realizing* visions with the confines of practical constraints of time, space, budget, and the state of the technologically possible. Architecture is about *articulating* visions that integrate the desires and constraints of clients within the bounds of engineering possibility."

Using this definition, developing new technology solutions, or re-engineering legacy systems, without re-architecting the environment, may be an expedient, but certainly not an elegant and effective approach. Though today's technology systems may meet the needs of yesterday's business requirements, an unarchitected environment does not hold much promise for delineating future systems.

From Chaos to Coordination: The Role of a Chief Information Executive

In most organizations there are at least three categories of employees who are engaged in some of the tasks associated with information management: specialists at Information Resource Centers, information systems professionals, and function-specific knowledge workers. Lack of active coordination and interaction between these three groups, has led to a chaotic and fragmented state of *information management* in most corporations.

For a successful information-centric re-engineering, it is imperative to designate a senior corporate staff with a rich and diverse cross-functional background, to *own*

the tasks of articulating the *information strategy*, planning, coordination, full delivery, and continuous adaptation of an information-enabled organization. Additionally, this individual is commissioned with the primary responsibility of bringing together and clarifying the overall information strategy for the organization, explaining the critical information needs and how they relate to the strategic business goals, and describing the information plans, management processes, and resource assumptions in the organization's business plan.

One would assume that Chief Information Officer (CIO) would naturally fit as a title for the owner of this domain, however, there are a few problems with this candidacy. Most importantly, the title is commonly used for another very important role within the organization. Unfortunately, most of the existing managers with CIO titles, are in reality chief information-technology officers (CITOs), and until recently were selected primarily for their technical, rather than cross-functional, organizational, and political skills. To avoid the existing content/context misalignment with the CIO title, I propose "Chief Information Executive" as a possible title for this role. (Webster's defines *executive* as "Of or responsible for the carrying out of plans or policies.") In order to have executive power, she should have the CIO, and Information Resource Centers directly report to her or at least have significant influence over them. Additionally, in order to be functionally effective, CIE should gain the strong support and cooperation of a representative set of various key functional-specific knowledge workers within the organization.

Since the task of information-centric re-engineering is a dynamic and perpetual task, the CIE position is a permanent one. This is in direct contrast with the typical role of a designated project manager who is commissioned to start and complete the exercise of a technology- or process-centric re-engineering.

Continuous Re-Engineering through Learning and Adaptation

In stable, slowly changing environments, learning and adaptation can proceed slowly. Trial and error is possibly the most appropriate learning strategy. In unstable, rapidly changing environments, however, explicit, systematic learning must be at the core of strategies to keep organizations adapted to their environment.

Learning is both the impetus and engine for change. As the impetus for change, learning is about watching the environment for soft and weak signals that present early warnings of threats or opportunities. It is about detecting changes that threaten to widen the gap between the environmental demands and organization as it currently operates.

As an engine for change, learning provides processes to close the widening gap. Learning is central to analyzing and describing the gap between environment and

current practice, identifying models and examples of other organizations' responses to similar gaps, and designing changes to current organizational practice to bridge the gap.

An information-enabled organization that results from an information-centric re-engineering process is a forward-thinking organization by design, and thus never ceases to learn. The process inherently promotes learning and adaptation through its closed loop mechanism. The last step of the process is designed to feed the cumulative learnings of an information-enabled organization back to the first step of the process in order to appropriately influence and adapt the strategic business plan. This pro-active systematic adaptation step should take place at least at the end of each strategic business planning cycle, if not more frequently.

A Final Note

Although much more can be written about and around this topic, I hope that this introduction to information-centric re-engineering has served to deliver its central message of a paradigm shift.

In closing, I would like to capture the essence of this paper by a simple reminder that "In times of change, learners inherit the earth, while the learned find themselves beautifully equipped to deal with a world that no longer exists." [7]

Acknowledgements

I would like to thank my good friend Khalil Emami for his early encouragements and support, and James Yu for his candid and constructive questions leading to significant improvement of this paper. I also would like to thank a numberless number of friends and colleagues at Hewlett-Packard, Cincom, Dun & Bradstreet, SAP, Wang Laboratories, and other corporations who have directly or indirectly contributed to the formation of the concept of information-centric re-engineering during various interactions over the past few years.

Notes & References

[1] Feyzi Fatehi, (1992), "From Downsizing to Rightsizing: A Guideline for Information Technology Restructuring in the '90s." Interex'92 Proceedings

[2] Much has been written on the subjects of efficiency and effectiveness. My favorite one is the one I have seen in Max DePree's famous book: *Leadership Is An Art*. He in turn, cites Peter Drucker. He writes: "... Peter Drucker has such a great ability to simplify concepts. One of the things he tells us is that efficiency is doing the thing right, but effectiveness is doing the right thing."

[3] See Julia King, (1994), "Re-engineering Slammed." *Computerworld*, June 13 issue, among many recently published write-ups on disappointing experiences with technology-centric re-engineering.

[4] See Kathy Spurr and Linda Hickman (1993), "Software Assistance for Business Re-Engineering." John Wiley & Sons Ltd., for more on this subject.

[5] The word strategy is often confused with long-range plans. A strategy is simply a plan for achieving an objective by focusing on critical goals. Therefore, the objective of strategic planning is to identify the most important goals and determine the most important actions for attaining these goals. An organization can develop a long-range plan that is not strategic or can develop a strategic plan that is not long range. *Strategic plans* focus primarily on goals. In contrast, *tactical plans* focus on actions that satisfy performance requirements including those that implement strategy, and *operational plans* explain in detail the steps needed to make tactics fully implemented.

[6] The heavy emphasis on financial and internally generated data at the expense of non-financial and externally generated information has led, in many instances, to a narrow definition of information. Since early commercial computers were better suited to manipulate numeric data than to work with unstructured "textual data," accounting functions were the first automated functions. For a number of reasons, these functions evolved into "Management Information Systems." While it is

undeniable that accounting and financial information are an important subset of management information, it is equally true that the idea that accounting systems are really Management Information Systems is as distorting as the identification of Information Technology with Information itself. (For a recent analysis of this subject see Sharon M. McKinnon and William J. Burns, Harvard Business School Press, 1992)

[7] James V. McGee and Laurence Prusak (1993), "Managing Information Strategically." John Wiley & Sons

[8] Robert E. Umbaugh (1994), "Quality and Control in IS." Auerbach Publications

Paper Number: 2002

Network Outsourcing: A Key to Corporate Reengineering

Ernest Eugster, Ph.D.

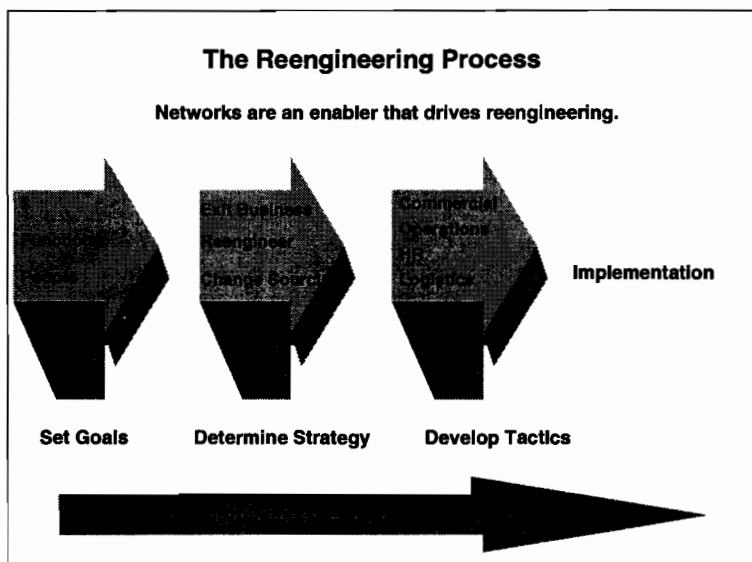
Random Access, Inc., 8000 East Iliff Ave., Denver, CO 80231

(303) 745-9600

In the boardrooms of many corporations, executives are invoking the "R" word: reengineering. They are talking about replacing existing ways of doing business with bold, new processes to be more competitive and profitable. "Reengineering is new, and it has to be done," says Peter F. Drucker, the most eminent of management experts. And many are following his words. *Reengineering the Corporation*, the book written by Michael Hammer, a former professor at MIT, and James Champy, has over 500,000 copies in print and has spent 21 weeks on the *New York Times* bestseller list.

Reengineering is real. Done well, reengineering delivers major gains in speed, productivity and profitability. Any skeptic need only look at the current activities of Chrysler Corp. Four years ago many industry experts agreed that Chrysler Corp. would go out of business due to increased competition and outdated manufacturing processes. Today, after retooling assembly plants and listening to customers, Chrysler is one of the most profitable car companies in the world.

Reengineering is more than reorganizing a few departments. It often involves radically changing the what, how and who of the business. Reengineering starts by



Network Outsourcing: A Key to Corporate Reengineering

looking into the future and working backwards. Reengineers assume that everything can be changed--from people to departments to processes. Once they have a vision of what the new company should look like, they take a saw and sandpaper to do the needed redesign work.

But when reengineers confront the substantive and technical problems, they quickly find that reengineering will not work without networks. Once considered solely as an expense on the corporate balance sheet, the role of networks and other information technology changed almost overnight when executives began to understand that LANs/WANs are an enabler that drives business process reengineering. Networks help companies be more competitive, enhance operations, empower knowledgeable workers and increase the speed and cost-efficiency of information handling. Developing a well-conceived networking strategy is basic to the success of business process reengineering.

Indeed, networks, like other information technology solutions, must be considered from the beginning of the reengineering process; not as an afterthought. A 1993 study by Boston-based market researcher, The Yankee Group, found that reengineering directly drives what type of information systems and networks companies buy. When a company needs to improve its global competitive position, important investments tend to be made in distributed information systems, "any-to-any" LAN/WAN connectivity and centralized network management. And when a company needs to shorten product development times, investments often focus on distributed image processing, workflow software as well as extending the network to customers and suppliers.

Challenges: But integrating reengineering efforts with information technology is not easy. Although the U.S. reengineering market is projected to be a \$27 billion industry by 1997, many efforts at process engineering fail. Seventy percent of all reengineering projects do not successfully reach their goals, according to market researcher CSC Index.

When redesigning a company, reengineers face several daunting challenges. First, reengineers need to define information flows. According to Peter Druker, "Organizations need to learn to ask, 'What information do we need in this company?' 'When?' 'In what form?' and 'Where do we get it.'"

The second challenge is knowing how to wisely use advanced networking and information technologies to achieve major business changes. New technologies such as client/server computing and document image processing offer the means to increase productivity, lower costs and improve customer service. The challenge of corporate reengineers is understanding how to make organizational and process changes to take full advantage of the benefits offered by new technology.

The third challenge is integrating outdated, inefficient information systems. Older systems are proprietary and can consume large amounts of maintenance and support resources. But few companies have the budgets to completely replace older systems. As a result, information flow can be slow and inefficient.

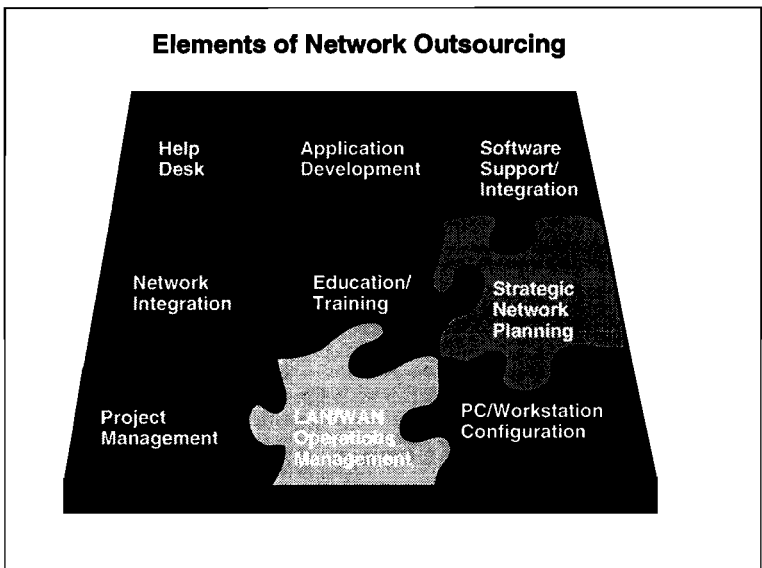
The fourth challenge is using often limited organizational resources. Many companies fail at reengineering because they lack the in-house expertise for implementing technology that is designed to improve the business processes. The

fast moving and increasing complexity of communications technology has overpowered many MIS departments. A 1993 survey by Computerworld of 294 MIS organizations showed the lack of in-house expertise to be the biggest problem facing managers. For many companies, downsizing a mission critical application onto a LAN or client/server environment can be a risky undertaking. Besides someone's job being at stake, the lack of in-house expertise can slow the reengineering implementation process and result in major cost overruns.

Outsourcing for Expertise: Recognizing many of these problems, increasing numbers of reengineers are choosing to turn over the building, integration and operation of all or a portion of their network to an outside vendor or outsourcer.

The word "outsourcing" is new, but the concept has been around for some time. It started with mainframes, where outsourcers managed and operated batch data processing, data storage and disaster recovery.

Today, outsourcing comes in many shapes and sizes. MIS managers are outsourcing complicated or routine parts of the network, freeing them to focus on business issues and technological problems. Others are looking for outsourcers to supplement in-house staff in providing higher service levels, more expertise in distributed networks, remote support and sophisticated network management. According to Dataquest, facilities management, custom application development and network support/integration are the three most frequently outsourced activities.



Outsourcing also can result in cost savings. Outsourcing can lower costs because some outsourcing vendors are willing to acquire staff and capital equipment from the customer. Cost savings can be especially great when a company is disorganized and wasteful in its data and networking operations. A good outsourcer can correct the situation and reduce costs.

It's no secret that outsourcing is the fastest growing segment in the information technology industry. According to Dataquest, the worldwide market for outsourcing, which in 1992 was estimated at \$45.5 billion, is expected to reach \$108.1 billion by 1997. And the industry is still in its infancy. In the United States, the number of contracts over \$500 million doubled from seven in 1992 to 14 in 1993.

Large computer and consulting firms have added outsourcing to their portfolio of services. Computer-services companies like EDS, Cap Gemini Sogeti and Anderson Consulting responded to outsourcing demands years ago. But now, major computer makers like IBM/ISSC, DEC and Hewlett-Packard are offering to help customers build, maintain and run large networks.

A number of smaller companies have also successfully targeted specific technical niches or vertical markets. For example, Concept Dynamics specializes in financial services and has a worldwide clientele. Others are creating consortiums such as the RightSource Group to target client/server computing. Random Access is specializing on LAN/WAN integration and outsourcing.

Admittedly, outsourcing is not a solution for every company. Swiss-based Ciba-Geigy decided to keep the management of its global multiplexer network in-house to lower risks and because no major cost savings could be identified.

But every company should evaluate outsourcing. That's how Eastman Kodak looked at its computer and telecommunication operations, which connects businesses in over 150 countries. Facing competitive threats, Kodak decided it had to reduce costs of its data center while maintaining increasing functionality of its global network. Rather than take a piecemeal approach, the company examined its operations from the outside in. It concluded that it wanted to transfer its facilities and people to an outsourcer in an effort to provide more efficient service and at a cost that would be less than if they managed the networks by themselves. Kodak also wanted outsourcing to result in a higher quality of work life for effected employees and to remain on the forefront of new technology developments.

Kodak's decision to outsource shows most of the important traits of using outsourcing to reengineer business processes. It was occurring in a dramatically competitive landscape; it involves major change, with big results; it cuts across departmental lines; it requires lengthy negotiations; and requires a new organization and mindset. "The hard part of the process," says Claude Millot, former manager of IS Operations and Software at Kodak's Paris office, "was to relinquish tactical control, while keeping strategic control and sharing decision making."

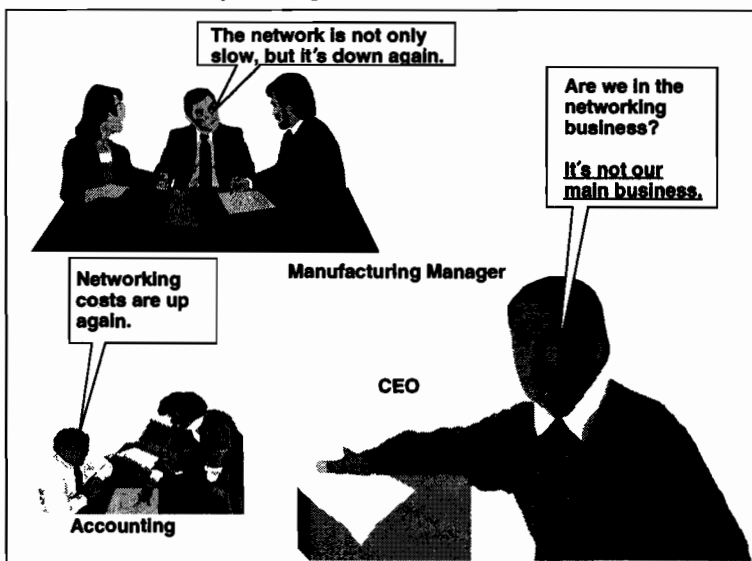
In the end, Kodak outsourced numerous functions to different outsourcers. In the United States, IBM was hired to take over the data center and for running the SNA network. DEC was hired to run the telecommunications operations. Entex (formerly JWP/Businessland) was hired to take over Kodak's PC operations. Kodak also hired Tigon to run the voice messaging system nationwide. In Europe, IBM is responsible for managing data centers, although other parts of the network are still run by Kodak. But Kodak is keeping a close eye on outsourcing opportunities resulting from the increasing deregulation and the move to high speed voice and data networks.

Was it worthwhile? "The capital budget was reduced and the annual expense budget was decreased," says Millot. "In addition nearly 700 Kodak employees were transferred to the outsourcers, while 1,000 employees were redeployed to other Kodak departments. Also the company experienced a rapid infusion of new technologies as outsourcers introduced technology based on their needs to support Kodak, not on what Kodak could afford."

The Emergence of the MIS Reengineer: While the rewards can be significant, many MIS managers fear the word "outsourcing," and will resist it. Some fear that their operation will be turned over to an outsider, leaving them without a job. Others fear that they will lose control of the network. This fear is especially strong in companies where the MIS department has just gained control of departmental local areas networks.

But it's the duty of today's MIS managers to understand, not fear outsourcing. MIS managers play a key role in corporate reengineering. Because information technology is an enabler of that drives business reengineering, it is essential that the MIS manager be a member of the reengineering team. In addition, when an MIS manager can unconsciously tie business strategy to technology, he can deal with issues *before* top level management imposes a solution. Furthermore, this skill can result in career advancement outside computer and networking operations. Here are a few tips:

- **Realize that CEOs/ CFOs are sold on the promises of reengineering and outsourcing.** Top executives know that if they are to succeed in a world of lower margins and increased competition, they must increase productivity, cut costs and be more focused. And that is exactly what they are doing. Companies are spinning off non-profitable businesses, and focusing on businesses where they have a competitive advantage, where manufacturing costs are low and where added value in product or service delivery can be provided.



Network Outsourcing: A Key to Corporate Reengineering

Although executives are still skeptical about information technology costs, they do drive the decision to outsource. Outsourcers are often told to deal strictly with the MIS executives empowered to make the final decisions. But, more often than not, the vendor who seemed to have gotten through most effectively at the CEO level won the business.

Thus, if MIS managers do not want solutions imposed on them, they need to understand the reengineering process and the strategies set by the top management. Attend a management seminar and start reading some of the leading books on reengineering. MIS managers also need to understand change management, which involves countering negative user reaction to new technology or outsourcing through strong training, education and special rewards. Proactive MIS managers can shape the reengineering process and affect the selection of the outsourcing arrangement.

- **Look for new technology and services that can result in business advantages.** Although the level of understanding of information technology issues is growing among the ranks of top executives, the MIS manager is still considered to be the “guardian” of the company’s information; controlling the use and destiny of technology. This requires the MIS manager to constantly keep up with potential breakthrough technologies that bring business solutions to your company. Of course, this involves attending seminars and reading the technical press. But it also involves developing an information strategy. More specifically, develop a technology assessment plan that contains descriptions of emerging technologies, their forecasted business benefits for the company, costs and expected payback periods.

Also include outsourcing. Outsourcing client/server applications development, for instance, can result in quicker and less costly deployment of solutions than if the work was done internally. When the document is complete, present it to top management. This helps educate management and users. It also positions the MIS manager as the leader in setting the strategic direction of an important company resource: **information**.

- **Understand your company’s key business.** To be a major player in the reengineering process, MIS managers need to know as much as possible about their company’s inner workings. If the firm manufactures cars or washing machines, then the MIS manager must understand the best way to support production logistics such as ordering supplies, inventory and order processing. MIS managers can not understand or start to define new ways of doing things if they do not understand the basic practices of the business.

- **Set an example.** If you are a believer of the reengineering process, use the MIS department to show how it can be done. Find ways to streamline operations, cut costs or improve productivity. Network operations centers are good candidates for quick reengineering efforts. Outsourcing a software help desk, for example, can open staff career opportunities, reduce down time and avoid job burnout. Such activities will increase the creditability of the MIS department within the company.

- **Be in control.** Look at outsourcing as a way of gaining greater control; not losing it. In a proper outsourcing agreement with an experienced service provider, the MIS manager should control the relationship from the start. Begin by requesting a written evaluation and business plan of the function to be outsourced from the

prospective vendor. The information you get about service levels and pricing will put you in a better position to determine if outsourcing is needed.

Once you choose a vendor and begin to work together, assign a senior executive to be responsible for the outsourcing arrangement. This will help facilitate communication as well as provide streamlined decision making when things go wrong or change.

Also, constantly monitor service levels. Insist that the outsourcer provide accurate and timely updates on the service, perhaps through daily, weekly and monthly reports. This will keep you aware of trends and changes and allow you to see things you might not have noticed when the function was handled in-house. For example, if you are outsourcing your data network, the arrangement should result in recommendations or suggestions to correct or enhance certain areas that actually improve network performance, staff productivity or costs. In fact, outsourcing may give you more information and, therefore, more control than you had before.

- **Look for the “fit”:** Just as any relationship between people, an amicable working relationship is key to the successful outsourcing. It is in everyone's interest that a “good chemistry” exist between you and the outsourcer--from account management to the technicians.

To ensure a good fit, insist on the right to interview and approve key outsourcing personnel. Also look at the culture of the outsourcer. If your company tends to be conservative with new technology, an outsourcing arrangement with a vendor that prides itself on technical innovation may not be successful.

Successful Outsourcing Tips

- Realize that management is sold on reengineering.**
- Look for technologies that promote business goals.**
- Know your company's business in detail.**
- Set an example.**
- Be in control.**
- Look for the right chemistry.**

In conclusion, network outsourcing is a key to successful reengineering. Although outsourcing is not for every company, every company should look at it. They will

find that outsourcing provides instant expertise in building, integrating and running a network. But the two biggest benefits will turn out to be improved productivity and profitability. What is an MIS manager to do? Although executives drive the decision to outsource, MIS managers have a unique opportunity to be on the reengineering team and make decisions on how information technology will drive reengineering. By clearly explaining to upper management the benefits that outsourcing can bring to reengineering, they can shape processes, affect the selection of solutions and advance their careers.

**Paper Number 2003
Legal Issues in EDI**

**Christopher Seline
Attorney
Formosa Transnational Attorneys at Law
136 Jen Ai Road
Section 3, 15th Floor
Taipei, TAIWAN
886.27.557.366**

Handouts will be provided at time of presentation

Source Code Escrow: A Critical Business & Legal Issue

Bea Strickland

SourceFile
50 Crisp Plaza, Suite 700
San Francisco, CA 94124-2924
1-800-2-ESCROW (800-237-2769)

Interex '94 Conference & Expo
September 21, 1994
Denver, Colorado

Source Code Escrow: A Critical Business & Legal Issue
2004-1

I. Introduction

A. Definition, source code escrow

Source code escrow is the act of depositing with a neutral third party (escrow firm), the source code and other necessary proprietary information for the benefit of the licensee. The escrow firm maintains the source codes and is able to release them to the licensee in the event the developer defaults on support, goes out of business, is involved in a merger or acquisition, etc. This enables the licensee to utilize the software uninterrupted.

B. State of the software industry

The software industry is ever changing and continually gaining complexity. In the past, source code escrow was particularly necessary for large, expensive, highly customized main frame system. Today, as we move to interconnectivity and client server technology, there are more interdependent relationships between software developers. Source code escrow agreements are often between the developer and an OEM, VAR, distributor, joint venture partner, and even lender.

II. Agreement Types

A. Three party. For highly customized software, the three party agreement amongst the escrow firm, the licensee, and the developer is still common.

B. Two party with developer. The a two-party agreement between the escrow firm and the developer. Additional beneficiaries may sign on to this agreement as they license the software from the developer. This agreement provides the opportunity to serve all licensees of a particular software developer for one or more software systems.

C. Two party with licensee. The two-party agreement between the escrow firm and licensee. As the licensee licenses technology from additional developers, these developers will acknowledge acceptance of and be added to the agreement. The acknowledgement identifies the developer, describes the software deposit, and, if necessary, provides an opportunity for the developer to request specific changes to the terms and conditions of the agreement.

III. Drafting the Agreement

A. Events which could trigger release of the source code

1. Developer defaults on support
2. Bankruptcy
3. Specific Events (ie. key personnel leaves)

B. Release conditions

1. An immediate release of the source code is suggested for accounts where the licensee actually has the right to the source code but elects

not to maintain it at their facility in order to avoid unnecessary legal complications and to ensure proper storage and maintenance of the source code.

2. Consultant release is suggested for critical applications which cannot wait for a dispute resolution in the event one arises. In this situation, the source codes are released to a predetermined party which maintains the software until other arrangements are settled.

C. Bankruptcy Code 365 (n)

In 1988 in the United State, Congress passed the Intellectual Property Act of 1988. Section 365 (n) of the Act changed the Bankruptcy Code in relation to source code escrow agreements. It was made clear that software license agreements are executory contracts and, as such, are terminated upon the filing of Bankruptcy of one or more party. However, the Act states that in the event the developer rejects the license agreement, the licensee has the right to reject or continue the license. Additionally, the Act states that the source code escrow agreement is "supplemental" to the license agreement and thus the trustee in bankruptcy may not interfere with the transfer of technology or the release of the source code to the licensee pursuant to the escrow agreement.

D. Technical Verification

Technical verification is critical to ensure that the deposit is complete and accurate. If the deposit is not complete, it is rarely due to malicious intent, but more commonly the result of assigning the important duty of preparing deposit materials to an already overworked programmer. For these reasons, various levels of verifications are offered and suggested:

A. File Listing

The loading of the source material deposit and running a file listing of the source code. At this level of verification we are able to confirm that the magnetic media on which the source material is stored, is in working order. We are also able to compare the size of the source material deposit to the expected size of the source code.

B. Compilation

The compiling of the source material deposit to make certain that the deposit may be compiled and that the source material deposit will generate working object code.

C. Size Comparison

The comparison of the size of the object code compiled from the source material deposit to the size of the licensee's object code. At this level we are able to test whether the source material deposit is complete.

D. Function Comparison

To execute on 1) the newly compiled object and on 2) the

licensee's object code any and all specific functions as defined by the licensee. Here, we are able to test whether the source material is accurate.

E. Economic Online Comparison

Once we have compiled the source code, we will make the newly compiled object code available to the licensee via modem. The licensee will be able to run test functions from the licensee's facility.

E. Deposit contents (as suggested by David Klausner)

Core Deliverables

<u>Description</u>	<u>Priority</u>	<u>When</u>
Code/executable/object/ binary tools such as compilers	10	At and between official release
Product assembly tools (Makefiles)	10	At and between official release
Executable machines/ descriptions and release	10	At and between official
Documentation	10	At and between official release
Ancillary Deliverables		
Lists of key personnel and their responsibilities	09	At initial deposit, updated with personnel changes or responsibility changes
Bug lists, fixed/unfixed	08	At official releases
Special modifications made for customers	08	At release to customers
Design documents, operational description	08	With initial deposit, updates internal changes
Source code control system	07	With initial deposit, updated changes
Forecasts cycle, Product indirection/plans, marketing information	05	With initial deposit, updated at every product cycle or redirection

Project tracking records,	05	With initial deposit, updated time plans, line at every project milestone
Bug tracking system and history	05	With Alpha release, updated with all releases
Wish lists/enhancements lists	02	At every project milestone
Customer feedback/survey	01	With initial deposit, updated with every milestone
Software development policies and procedures	01	With initial deposit, updated with changes

IV. Maintaining the Escrow

A. Secure facility

It is important that the facility where the source code is maintained offers structural and procedural security. For the proper maintenance of magnetic media, it must be temperature and humidity controlled, as well as properly built to protect against natural disasters such as fires and earthquakes. Additionally, it is important that the vault is owned and operated by the firm which has signed your escrow agreement and that it is run with strict procedures controlling access to the deposit.

B. Update maintenance

Make sure that your escrow firm provides a tickler system to encourage regular updates to the deposit materials.

C. Notification

The licensee should have direct contact with the escrow firm concerning deposits contents, updates to the deposit, and status of the agreement.

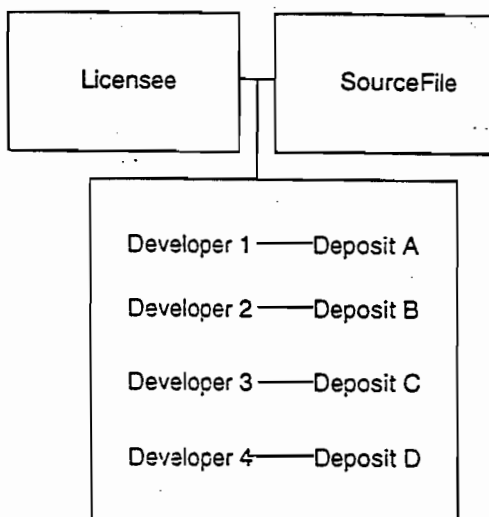
V. Conclusion

A. Successful agreements for successful arrangements. Drafting the agreement is just as important as carrying out the responsibilities of the arrangement. It is most important that the escrow agreement is signed at the same time the license agreement is signed because this is when the dependency begins and the terms are set.

B. Failing to plan is planning to fail. We encourage you to plan for the worst but to hope for the best.

VI. Sample Agreements

**CORPORATE SOURCEFLEX
SOFTWARE SOURCE CODE ESCROW AGREEMENT**



This contract is a two-party agreement between SourceFile and the Licensee. Developers will acknowledge acceptance and be added to this agreement as the Licensee licenses their technology. The Corporate SourceFlex provides the opportunity to serve all developers of one or more software systems licensed by a particular Licensee.

CORPORATE SOURCEFILEX
SOFTWARE SOURCE CODE ESCROW AGREEMENT
SOURCEFILE NUMBER: _____

This Software Source Code Escrow Agreement, dated as of _____, 1994, by and between FileSafe, Inc., a California corporation, doing business as SourceFile ("SourceFile") located at 50 Crisp Plaza, Suite 700, San Francisco, CA 94124-2924 and _____, a _____ located at _____ ("Beneficiary"), and each Depositor identified by Beneficiary to SourceFile as provided for in Section 3 hereof (each a "Depositor", collectively the "Depositors").

RECITALS:

A. Pursuant to certain software license agreements (each a "License Agreement", collectively the "License Agreements"), Depositor licenses, to certain licensees, certain software in object code form (the "Software"). A description of the Software effective as of the date hereof, is provided in Addenda as in Exhibit "C".

B. The Software is the proprietary and confidential information of Depositor, and Depositor desires to protect such ownership and confidentiality.

C. Depositor desires to ensure the availability to Beneficiary of the source code and all necessary proprietary information related to the Software (the "Source Material") in the event any of the conditions set forth in Section 4 of this Agreement should occur.

AGREEMENT:

1. Delivery of Source Material to SourceFile. Depositor shall deliver to SourceFile a parcel (the "Parcel") sealed by Depositor, which Depositor represents and warrants is the Source Material.

2. Acknowledgement of Receipt by SourceFile. SourceFile shall promptly acknowledge to Beneficiary, in writing, the receipt of the Parcel and of any supplements to the Source Material which are added to the Parcel.

3. Acknowledgement by Depositors. For purposes of this Agreement, a developer the Software shall be a Depositor hereunder with such rights of a Beneficiary as set forth herein, only if (i) such developer sent to SourceFile a fully executed copy of the form of acknowledgement attached hereto as Exhibit "C", in which the developer accepts the terms of this Agreement. The name and addresses of the Depositors shall be described in one or more schedules of Depositors to be presented to SourceFile from time to time by Beneficiary. A schedule of Depositors effective as of the date of this Agreement is attached hereto as Exhibit "B". All other developers of the Software shall have no rights hereunder and SourceFile shall have no duties to such developers.

4. Terms and Conditions of the Source Code Escrow. The Parcel shall be held by SourceFile upon the following terms and conditions:

(i) In the event that SourceFile is notified by a Beneficiary that Depositor is unwilling or unable to support or maintain the Software, in breach of the License Agreement with such Beneficiary and that such Beneficiary has given Depositor written notice of such breach (the "Release Condition"), SourceFile shall immediately notify Depositor of its receipt of the Beneficiary's notice and shall provide a copy of such notice to Depositor promptly thereafter.

(ii) If SourceFile does not receive Contrary Instructions from Depositor, as defined below, within ten (10) days of the giving of such notice to Depositor, SourceFile shall deliver a copy of the Source Material to the Beneficiary demanding delivery within sixty (60) days after the date of the Beneficiary's original notice to SourceFile. "Contrary Instructions" for the purposes of this Section 3 shall mean the filing of a notice with SourceFile by Depositor, with

a copy to the Beneficiary demanding delivery, stating that the Release Condition has not occurred or has been cured.

(iii) If SourceFile does receive Contrary Instructions from Depositor within ten (10) days of the giving of such notice to Depositor, SourceFile shall not deliver a copy of the Source Material to the Beneficiary, but shall continue to store the Parcel until: (1) otherwise directed by the Depositor and Beneficiary jointly, (2) SourceFile has received notice of the resolution of the dispute by a court of competent jurisdiction, or (3) SourceFile has deposited the Parcel with a trustee selected by a court of competent jurisdiction for the purpose of determination of its obligations under this Agreement.

(iv) Upon receipt of Contrary Instructions from Depositor, SourceFile shall have the absolute right, at SourceFile's election to file an action in interpleader requiring the Depositor and Beneficiary to answer and litigate their several claims and rights among themselves. SourceFile is hereby authorized to comply with the applicable interpleader statutes of the State of California in this regard.

5. Term of Agreement. This Agreement shall have an initial term of three (3) years. The term shall be automatically renewed on a yearly basis thereafter, unless Beneficiary or SourceFile notifies the other party at least forty-five (45) days in advance of the end of the current term of its election to terminate this Agreement.

6. Compensation of SourceFile. Beneficiary agrees to pay SourceFile reasonable compensation for the services to be rendered hereunder, in accordance with SourceFile's then current schedule of fees, and will pay or reimburse SourceFile upon request for all reasonable expenses, disbursements and advances, including software duplication charges and reasonable attorneys' fees, incurred or made by it in connection with carrying out its duties hereunder. SourceFile's schedule of fees for the initial term of this Agreement is attached hereto as EXHIBIT "A".

7. Limitation of Duties of SourceFile. SourceFile undertakes to perform only such duties as are expressly set forth herein. SourceFile has no knowledge of, and makes no representations with respect to, the contents or substance of the Parcel, the Software or the Source Material.

8. Limitation of Liability of SourceFile. SourceFile may rely on and shall sustain no liability as a result of acting or refraining from acting upon any written notice, instruction or request furnished to SourceFile hereunder which are reasonably believed by SourceFile to be genuine and to have been signed or presented by a personal reasonably believed by SourceFile to be authorized to act on behalf of the parties hereto. SourceFile shall not be liable for any action taken by it in good faith and believed by it to be authorized or within the rights or powers conferred upon it by this Agreement. SourceFile may consult with counsel of its own choice, and shall have full and complete authorization and protection for any action taken or suffered by it hereunder, in good faith and in accordance with the opinion of such counsel.

9. Indemnification of SourceFile. Depositor and Beneficiary jointly agree to indemnify and defend SourceFile and to hold it harmless from and against, any loss, liability or expense including reasonable attorneys' fees and costs incurred by SourceFile arising out of or in connection with this Agreement, carrying out its duties hereunder, any other claim of liability with respect to the Source Material. In the event suit is brought by any party to this Agreement, or any other party, as against any other party, including SourceFile, claiming any right they may have as against each other or against SourceFile, then in that event the parties hereto, agree to pay to SourceFile any attorney's fees and cost incurred by SourceFile in connection therewith.

10. Record Keeping and Inspection of Software. SourceFile shall maintain complete written records of all materials deposited by Depositor pursuant to this Agreement. During the term of this Agreement, Depositor shall be entitled at reasonable times during normal business hours and upon reasonable notice to SourceFile to inspect the records of SourceFile maintained pursuant to this Agreement and to inspect the facilities of SourceFile and the physical condition of the Source Material.

11. Technical Verification. Beneficiary reserves the option to request SourceFile to verify the Source Material for completeness and accuracy. SourceFile may elect to perform the verification at its sight or at the developers sight. Depositor agrees to cooperate with SourceFile in the verification process by providing its facilities and computer systems and by permitting

SourceFile and at least one employee of Beneficiary to be present during the verification of Source Material.

12. Restriction on Access to Software. Except as required to carry out its duties hereunder, SourceFile shall not permit any SourceFile employee, Beneficiary or any other person access to the Software, unless consented to in writing by Depositor. SourceFile shall use its best efforts to avoid unauthorized access to Software by its employees or any other person.

13. Updates of the Source Material.

(i) Depositor shall provide from time to time, supplements of the Source Material for each version of the Software.

(ii) A representative of SourceFile shall place such supplements into the Parcel containing the Source Material without removing the Source Material.

(iii) Depositor shall send to SourceFile a duplicate of the Source Material within three (3) days after receiving written notice from SourceFile that the Source Material has been destroyed or damaged.

14. Bankruptcy. All supplements to Source Material and duplicate Source Material shall be subject to the terms and provisions of this Agreement.

15. Notices. Any notice or other communication required or permitted under this Agreement shall be in writing and shall be deemed to have been duly given on the date service is served personally, sent by overnight courier, or five (5) days after the date of mailing if sent registered mail, postage prepaid, return receipt required, and addressed as follows or to such other address or telefax number as either party may, from time to time, designate in a written notice given in like manner:

TO DEPOSITOR: AS SET FORTH ON EXHIBIT "B" SCHEDULE OF DEPOSITORS

TO BENEFICIARY:

TO SOURCEFILE: SOURCEFILE

50 Crisp Plaza

Suite 700

San Francisco, California 94124-2924

Attn: Ms. Bea Strickland

Facsimile: (415) 715-2733

15. Miscellaneous Provisions.

(a) Waiver. Any term of this Agreement may be waived by the party entitled to the benefits thereof, provided that any such waiver must be in writing and signed by the party against whom the enforcement of the waiver is sought. No waiver of any condition, or of the breach of any provision of this Agreement, in any one or more instances, shall be deemed to be a further or continuing waiver of such condition or breach. Delay or failure to exercise any right or remedy shall not be deemed the waiver of that right or remedy.

(b) Modification or Amendment. Any modification or amendment of any provision of this Agreement must be in writing, signed by the parties hereto and dated subsequent to the date hereof.

(c) Governing Law. This Agreement shall be governed by and construed in accordance with the laws of the State of California.

(d) Headings; Severability. The headings appearing at the beginning of the sections contained in this Agreement have been inserted for identification and reference purposes only and shall not be used to determine the construction or interpretation of this Agreement. If any provision of this Agreement is held to be invalid, illegal or unenforceable, the validity, legality and enforceability of the remaining provisions shall not in any way be affected or impaired thereby.

(e) Further Assurances. The parties agree to perform all acts and execute all supplementary instruments or documents which may be reasonably necessary to carry out the provisions of this Agreement.

(f) Entire Agreement. This Agreement, including the attachments hereto, contains the entire understanding between the parties and supersedes all previous communications, representations and contracts, oral or written, between the parties, with respect to the subject matter thereof. It is agreed and understood that this document and agreement shall be the whole and only agreement between the parties hereto, with regard to these escrow instructions and the obligations of SourceFile herein, in connection with this Agreement, and shall supersede and cancel any prior instructions. SourceFile is specifically directed to follow these instructions only and SourceFile shall have no responsibility to follow the terms of any prior agreements or understandings.

IN WITNESS WHEREOF, the parties have executed this Agreement as of the date first above written.

BENEFICIARY

By: _____
Name: _____
Title: _____

[FILESAFE\B\CORPSOURCE-2.RDL]

Document Version Date: July 8, 1994

Document Print Date: July 7, 1994

SOURCEFILE

FileSafe, Inc.,
a California corporation

By: _____
Name: Bea Strickland
Title: Director

EXHIBIT "A"

SOURCEFILE COMPENSATION SCHEDULE

EXHIBIT "B"

SCHEDULE OF DEPOSITORS

EXHIBIT "C-1"

ACKNOWLEDGEMENT BY DEPOSITOR

The undersigned hereby acknowledge, accepts, and agrees to be bound by the terms of the attached Corporate SourceFlex Software Source Code Escrow Agreement by and between FileSafe, Inc., a California corporation doing business as SourceFile, as Escrow Agent and _____, as Beneficiary, dated _____, 199_.

DEPOSITOR:

BENEFICIARY:

Once executed, send original by CERTIFIED OR REGISTERED MAIL to:

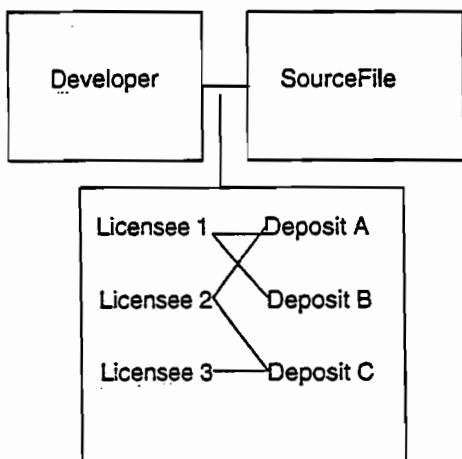
SOURCEFILE:

SOURCEFILE
50 Crisp Plaza
Suite 700
San Francisco, California 94124-2924
Attn: Ms. Bea Strickland
Facsimile (415) 822-2570

DESCRIPTION OF SOURCE MATERIAL DEPOSIT

Source Code Escrow: A Critical Business & Legal Issue
2004-10

SOURCEFLEX SOFTWARE SOURCE CODE ESCROW AGREEMENT



This contract is a two-party agreement between SourceFile and the Developer. End Users may sign on to this agreement as they license the technology from the Developer. The SourceFlex contract provides the opportunity to serve all licensees of a particular software Developer for one or more systems.

SOURCEFLEX
SOFTWARE SOURCE CODE ESCROW AGREEMENT
SOURCEFILE NUMBER: _____

This Software Source Code Escrow Agreement, dated as of _____, 1994, by and between FileSafe, Inc., a California corporation, doing business as SourceFile ("SourceFile") located at 50 Crisp Plaza, Suite 700, San Francisco, CA 94124-2924 and _____, located at _____ ("Depositor"), and each Beneficiary identified by Depositor to SourceFile as provided for in Paragraph 3 hereof (each a "Beneficiary", collectively the "Beneficiaries").

RECITALS:

A. Pursuant to certain software license agreements (each a "License Agreement", collectively the "License Agreements"), Depositor licenses to certain licensees certain software in object code form (the "Software"). A description of each Software effective as of the date hereof, is attached hereto as Exhibit "A".

B. The Software is the proprietary and confidential information of Depositor, and Depositor desires to protect such ownership and confidentiality.

C. Depositor desires to ensure the availability to its Beneficiaries of the source code and all necessary proprietary information related to the Software (the "Source Material") in the event certain conditions set forth in Paragraph 4 of this Agreement should occur.

AGREEMENT:

1. Delivery of Source Code to SourceFile. Depositor shall deliver to SourceFile a parcel (the "Parcel") sealed by Depositor, which Depositor represents and warrants is two copies of the Source Material. SourceFile has no knowledge of, and makes no representations with respect to, the contents or substance of the Parcel, the Software or the Source Material.

2. Acknowledgement of Receipt by SourceFile. Promptly after receipt of the Parcel and of any supplements to the Source Material, SourceFile shall notify in writing such Beneficiaries for which Depositor has paid SourceFile the fee for such notice. Depositor shall provide supplements to the Source Material for each version of the Software. Depositor shall send to SourceFile a duplicate of the Source Material within three (3) days after receiving written notice from SourceFile that the Source Material has been destroyed or damaged. All supplements shall be subject to the terms and provisions of this Agreement. SourceFile will notify Beneficiary and Depositor of each update to the Source Material. Such notification will be sent via certified mail, return receipt required.

3. Acknowledgement by Beneficiaries. For purposes of this Agreement, a licensee of the Software shall be a Beneficiary hereunder with such rights of a Beneficiary as set forth herein, only if (i) such licensee is identified on the current schedule of Software licensees delivered to SourceFile by Depositor from time to time and (ii) such licensee has sent to SourceFile a fully executed copy of the form of acknowledgement attached hereto as Exhibit "B", in which such licensee accepts the terms of this Agreement. The name and addresses of the Beneficiaries shall be described in one or more schedules of Beneficiaries to be presented to SourceFile from time to time by Depositor. A schedule of Beneficiaries effective as of the date of this Agreement is attached hereto as Exhibit "C". All other licensees of the Software shall have no rights hereunder and SourceFile shall have no duties to such licensees.

4. Terms and Conditions of the Source Material Escrow. The Parcel shall be held by SourceFile upon the following terms and conditions:

(i) In the event that (1) SourceFile is notified by Beneficiary that Depositor is unwilling or unable to support or maintain the Software in breach of its License Agreement with Beneficiary and that the Beneficiary has

given Depositor written notice of such breach (the "Release Condition") and (2) Beneficiary has paid to SourceFile all fees and charges then due and owing, SourceFile shall follow the following procedures set forth in this Section 4, parts (ii), (iii), (iv) and (v).

(ii) SourceFile shall promptly notify Depositor of the occurrence of the Release Condition and shall provide to Depositor a copy of Beneficiary's notice to SourceFile.

(iii) If SourceFile does not receive Contrary Instructions, as defined below, from Depositor within thirty (30) days following SourceFile's delivery of a copy of such notice to Depositor, SourceFile shall deliver a copy of the Source Material to Beneficiary. "Contrary Instructions" for the purposes of this Section 3 shall mean the filing of written notice with SourceFile by Depositor, with a copy to the Beneficiary demanding delivery, stating that the Release Condition has not occurred or has been cured.

(iv) If SourceFile receives Contrary Instructions from Depositor within thirty (30) days of the giving of such notice to Depositor, SourceFile shall not deliver a copy of the Source Material to the Beneficiary, but shall continue to store the Parcel until: (1) otherwise directed by the Depositor and Beneficiary jointly; (2) SourceFile has received a copy of an order of a court of competent jurisdiction directing SourceFile as to the disposition of the Source Material; or (3) SourceFile has deposited the Parcel with a court of competent jurisdiction or a Trustee or receiver selected by such court pursuant to this Section 4, part (v) below.

(v) Upon receipt of Contrary Instructions from Depositor, SourceFile shall have the absolute right, at SourceFile's election to file an action in interpleader requiring the Depositor and Beneficiary to answer and litigate their several claims and rights amongst themselves. SourceFile is hereby authorized to comply with the applicable interpleader statutes of the State of California in this regard.

5. Term of Agreement. This Agreement shall have an initial term of three (3) years. The term shall be automatically renewed on a yearly basis thereafter, unless Depositor or SourceFile notifies the other party in writing at least forty-five (45) days prior to the end of the then current term of its intention to terminate this Agreement.

6. Compensation of SourceFile. Depositor agrees to pay SourceFile reasonable compensation for the services to be rendered hereunder in accordance with SourceFile's then current schedule of fees, and will pay or reimburse SourceFile upon request for all reasonable expenses, disbursements and advances, including software duplication charges and reasonable attorneys' fees, incurred or made by it in connection with carrying out its duties hereunder. SourceFile's schedule of fees for the initial term of this Agreement is attached to this Agreement as Exhibit "D".

7. Limitation of Duties of SourceFile. SourceFile undertakes to perform only such duties as are expressly set forth herein.

8. Limitation of Liability of SourceFile. SourceFile may rely on and shall suffer no liability as a result of acting or refraining from acting upon any written notice, instruction or request furnished to SourceFile hereunder which is reasonably believed by SourceFile to be genuine and to have been signed or presented by a person reasonably believed by SourceFile to be authorized to act on behalf of the parties hereto. SourceFile shall not be liable for any action taken by it in good faith and believed by it to be authorized or within the rights or powers conferred upon it by this Agreement. SourceFile may consult with counsel of its own choice, and shall have full and complete authorization and protection for any action taken or suffered by it hereunder in good faith and in accordance with the opinion of such counsel.

9. Indemnification of SourceFile. Depositor and Beneficiary jointly agree to indemnify and defend SourceFile and to hold it harmless from and against, any loss, liability or expense incurred by SourceFile, arising out of or in connection with this Agreement, carrying out its duties hereunder, any other claim of liability with respect to the Source Material. In the event suit is brought by any party to this Agreement, or any other party, as against any other party, including SourceFile, claiming any right they may have as against each other or against SourceFile, then in that event the parties hereto, agree to pay to SourceFile any attorney's fees and cost incurred by SourceFile in connection therewith.

10. Record Keeping and Inspection of Software. SourceFile shall maintain complete written records of all materials deposited by Depositor pursuant to this Agreement. During the term of this Agreement, Depositor shall be entitled at reasonable times during normal business hours and upon reasonable notice to SourceFile to inspect the records of SourceFile maintained pursuant to this Agreement and to inspect the facilities of SourceFile and the physical condition of the Source Material.

11. Technical Verification. Beneficiary reserves the option to request SourceFile to verify the Source Material for completeness and accuracy. SourceFile may elect to perform the verification at its site or at the developers site. Depositor agrees to cooperate with SourceFile in the verification process by providing its facilities and computer systems and by permitting SourceFile and at least one employee of Beneficiary to be present during the verification of Source Material.

12. Restriction on Access to Software. Except as required to carry out its duties hereunder, SourceFile shall not permit any SourceFile employee, Beneficiary or any other person access to the Software except as provided herein, unless consented to in writing by Depositor. SourceFile shall use its best efforts to avoid unauthorized access to the Source Material by its employees or any other person.

13. Bankruptcy. Depositor and Beneficiary acknowledge that this Agreement is an "agreement supplementary to" the License Agreement as provided in Section 365 (n) of Title 11, United State Code (the "Bankruptcy Code"). Depositor acknowledges that if Depositor, as a debtor in possession or a trustee in Bankruptcy in a case under the Bankruptcy Code, rejects the License Agreement or this Agreement, Beneficiary may elect to retain its rights under the License Agreement and this Agreement as provided in Section 365 (n) of the Bankruptcy Code. Upon written request of Beneficiary to Depositor or the Bankruptcy Trustee, Depositor or such Bankruptcy Trustee shall not interfere with the rights of Beneficiary as provided in the License Agreement and this Agreement, including the right to obtain the Source Material from SourceFile.

14. Notice. Any notice or other communication required or permitted under this Agreement shall be in writing and shall be deemed to have been duly given on the date service is served personally, sent by overnight courier, or five (5) days after the date of mailing if sent registered mail, postage prepaid, return receipt required, and addressed as follows or to such other address or facsimile number as either party may, from time to time, designate in a written notice given in like manner:

TO DEPOSITOR:

TO SOURCEFILE: SourceFile
50 Crisp Plaza
Suite 700
San Francisco, California 94124-2924
Attn: Ms. Bea Strickland
Telephone: (415) 715-2733
Facsimile: (415) 822-7322

TO BENEFICIARY: As set forth in Exhibit "C" Schedule of Beneficiaries.

15. Miscellaneous Provisions.

(a) **Waiver.** Any term of this Agreement may be waived by the party entitled to the benefits thereof, provided that any such waiver must be in writing and signed by the party against whom the enforcement of the waiver is sought. No waiver of any condition, or of the breach of any provision of this Agreement, in any one or more instances, shall be deemed to be a further or continuing waiver of such condition or breach. Delay or failure to exercise any right or remedy shall not be deemed the waiver of that right or remedy.

(b) Modification or Amendment. Any modification or amendment of any provision of this Agreement must be in writing, signed by the parties hereto and dated subsequent to the date hereof.

(c) Governing Law. This Agreement shall be governed by and construed in accordance with the laws of the State of California.

(d) Headings; Severability. The headings appearing at the beginning of the sections contained in this Agreement have been inserted for identification and reference purposes only and shall not be used to determine the construction or interpretation of this Agreement. If any provision of this Agreement is held to be invalid, illegal or unenforceable, the validity, legality and enforceability of the remaining provisions shall not in any way be affected or impaired thereby.

(e) Further Assurances. The parties agree to perform all acts and execute all supplementary instruments or documents which may be reasonably necessary to carry out the provisions of this Agreement.

(f) Entire Agreement. This Agreement, including the attachments hereto, contains the entire understanding between the parties' and supersedes all previous communications, representations and contracts, oral or written, between the parties, with respect to the subject matter thereof. It is agreed and understood that this document and agreement shall be the whole and only agreement between the parties hereto with regard to these escrow instructions and the obligations of SourceFile herein in connection with this Source Code escrow, and shall supersede and cancel any prior instructions. SourceFile is specifically directed to follow these instructions only and SourceFile shall have no responsibility to follow the terms of any prior agreements or oral understandings.

IN WITNESS WHEREOF, the parties have executed this Agreement as of the date first above written.

DEPOSITOR

SOURCEFILE

FileSafe, Inc.,
a California corporation

By: _____

By: _____

Name: _____

Name: Bea Strickland

Title: _____

Title: Director

[SOURCEFILE\B\SOURCE.FLX]

Document Version Date: July 8, 1994

Document Print Date: July 7, 1994

EXHIBIT "A"

DESCRIPTION OF SOURCE MATERIAL

Source Material Deposit A: _____

Source Material Deposit B: _____

Source Material Deposit C: _____

Source Material Deposit D: _____

EXHIBIT "B"

FORM OF ACKNOWLEDGEMENT BY BENEFICIARY

The undersigned hereby acknowledge, accepts and agrees to be bound by the terms of the attached SourceFlex Software Source Code Escrow Agreement by and between SourceFile, Inc., a California corporation, as Escrow Agent and _____, as Licensor, dated _____, 199__.

BENEFICIARY: _____

Signature: _____

Name: _____

Title: _____

Address: _____

Telephone: _____

Facsimile: _____

DEPOSITOR: _____

Telephone: _____

Facsimile: _____

Please send CERTIFIED OR REGISTERED MAIL to:

SOURCEFILE:

SOURCEFILE

50 Crisp Plaza

Suite 700

San Francisco, California 94124-2924

Attn: Ms. Bea Strickland

Telephone (415) 715-2733

Facsimile (415) 822-4302

The following is a list of the Source Material Deposits placed in escrow with SourceFile for Beneficiary: _____

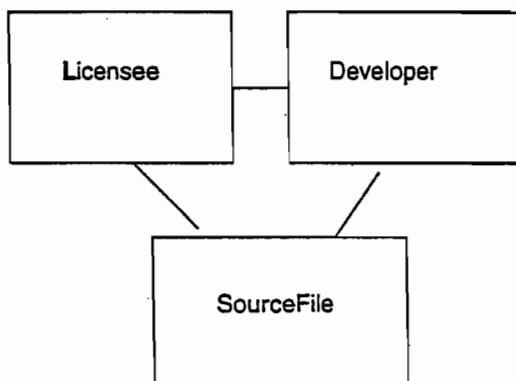
EXHIBIT "C"

SCHEDULE OF BENEFICIARIES OF THE SOFTWARE

EXHIBIT "D"

SOURCEFILE COMPENSATION SCHEDULE

**SOURCEPRO
SOFTWARE SOURCE CODE ESCROW AGREEMENT**



This contract is a three-party agreement amongst the Developer, the Licensee and SourceFile.

Source Code Escrow: A Critical Business & Legal Issue
2004-17

SOURCEPRO
SOFTWARE SOURCE CODE ESCROW AGREEMENT
SOURCEFILE NUMBER: _____

This Software Source Code Escrow Agreement, dated as of _____, 1994, by and among FileSafe, Inc., a California corporation, doing business as SourceFile located at 50 Crisp Plaza, Suite 700, San Francisco, CA, 94124-2924 ("SourceFile"), _____
a _____ located at _____, ("Depositor"), and _____ a
_____ located at _____, ("Beneficiary").

RECITALS:

- A. Pursuant to a certain software license agreement ("License Agreement"), Depositor has licensed certain software in object code form (the "Software") to Beneficiary.
- B. The Software is the proprietary and confidential information of Depositor and Depositor desires to protect such ownership and confidentiality.
- C. Depositor desires to ensure the availability to Beneficiary of the source code and all necessary proprietary information related to the Software, (collectively the "Source Material"), in the event certain conditions set forth in Section 3 of this Agreement should occur.

AGREEMENT:

1. Delivery of Source Material to SourceFile. Depositor shall deliver to SourceFile a parcel (the "Parcel"), sealed by Depositor, which Depositor represents and warrants contains the Source Material and any necessary proprietary information. SourceFile undertakes to perform only such duties as are expressly set forth herein. The Source Material shall be held by SourceFile. As of the date of this Agreement, Depositor shall complete a detailed description of the Source Material which is attached to this Agreement as Exhibit "A". 2. Acknowledgement of Receipt by SourceFile. SourceFile shall promptly acknowledge Depositor and to Beneficiary the receipt of the Parcel and any supplements to the Source Material which are added to the Parcel. Depositor shall provide to SourceFile the latest version of the Software at least every six months as supplements to the Source Material without removing the Source Material. Depositor shall send to SourceFile a duplicate of the Source Material within three (3) days after receiving written notice from SourceFile that the Source Material has been destroyed or damaged. All supplements shall be subject to the terms and provisions of this Agreement. SourceFile will notify Beneficiary and Depositor of each update to the Source Material. Such notification will be sent via certified mail, return receipt required.

3. Terms and Conditions of the Source Material Escrow. The Source Material shall be held by SourceFile (in the FileSafe underground vault, specifically designed for magnetic media storage), upon the following terms and conditions:

(i) In the event that (1) SourceFile is notified by Beneficiary that Depositor is unwilling or unable to support or maintain the Software in breach of its License Agreement with Beneficiary and that Beneficiary has given Depositor written notice of such breach (the "Release Condition") and (2) Beneficiary has paid to SourceFile all fees and charges then due and owing, SourceFile shall follow the following procedures set forth in Section 3, parts (ii), (iii), (iv) and (v):

(ii) SourceFile shall promptly notify Depositor of the occurrence of the Release Condition and shall provide to Depositor a copy of Beneficiary's notice to SourceFile;

(iii) If SourceFile does not receive Contrary Instructions, as defined below, from Depositor within thirty (30) days following SourceFile's delivery of a copy of such notice to Depositor, SourceFile shall deliver the Source Material to Beneficiary within 45 days of the date of Beneficiary's original notice to Depositor. "Contrary

Instructions* for the purposes of this Section 3 shall mean the filing of written notice with SourceFile by Depositor, with a copy to Beneficiary demanding delivery, stating that the Release Condition has not occurred or has been cured;

(iv) If SourceFile receives Contrary Instructions from Depositor within thirty (30) days of the giving of such notice to Depositor, SourceFile shall not deliver the Source Material to Beneficiary, but shall continue to store the Source Material until: (1) otherwise directed by the Depositor and Beneficiary jointly; (2) SourceFile has received a copy of an order of a court of competent jurisdiction directing SourceFile as to the disposition of the Source Material; or (3) SourceFile has deposited the Parcel with a court of competent jurisdiction or a Trustee or receiver selected by such court pursuant to Section 3, subpart (v) below;

(v) Upon receipt of Contrary Instructions from Depositor, SourceFile shall have the absolute right, at SourceFile's election to file an action in interpleader requiring the Depositor and Beneficiary to answer and litigate their several claims and rights among themselves. SourceFile is hereby authorized to comply with the applicable interpleader statutes of the State of California in this regard.

4. Term of Agreement. This Agreement shall have an initial term of three (3) years. The term shall be automatically renewed on a yearly basis thereafter, unless Beneficiary, Depositor or SourceFile notifies the other parties in writing at least forty-five (45) days prior to the end of the then current term of its intention to terminate this Agreement.

5. Compensation of SourceFile. Beneficiary agrees to pay SourceFile reasonable compensation for the services to be rendered hereunder in accordance with SourceFile's then current schedule of fees, and will pay or reimburse SourceFile upon request for all reasonable expenses, disbursements and advances, including software duplication charges and reasonable attorneys' fees, incurred or made by it in connection with carrying out its duties hereunder. SourceFile's schedule of fees for the initial term of this Agreement is attached to this Agreement as Exhibit "B".

6. Limitation of Liability of SourceFile. SourceFile may rely on and shall suffer no liability as a result of acting or refraining from acting upon any written notice, instruction or request furnished to SourceFile hereunder which is reasonably believed by SourceFile to be genuine and to have been signed or presented by a person reasonably believed by SourceFile to be authorized to act on behalf of the parties hereto. SourceFile shall not be liable for any action taken by it in good faith and believed by it to be authorized or within the rights or powers conferred upon it by this Agreement. SourceFile may consult with counsel of its own choice, and shall have full and complete authorization and protection for any action taken or suffered by it hereunder in good faith and in accordance with the opinion of such counsel.

7. Indemnification of SourceFile. Depositor and Beneficiary jointly agree to indemnify and defend SourceFile and to hold it harmless from and against, any loss, liability or expense incurred by SourceFile, arising out of or in connection with this Agreement, carrying out its duties hereunder, and any other claim of liability with respect to the Source Material. In the event suit is brought by any party to this Agreement, or any other party, as against any other party, including SourceFile, claiming any right they may have as against each other or against SourceFile, then in that event the parties hereto, agree to pay to SourceFile any attorney's fees and cost incurred by SourceFile in connection therewith.

8. Record Keeping and Inspection of Software. SourceFile shall maintain complete written records of all materials deposited by Depositor pursuant to this Agreement. During the term of this Agreement, Depositor shall be entitled at reasonable times during normal business hours and upon reasonable notice to SourceFile to inspect the records of SourceFile maintained pursuant to this Agreement and to inspect the facilities of SourceFile and the physical condition of the Source Material.

9. Technical Verification. Beneficiary reserves the option to request SourceFile to verify the Source Material for completeness and accuracy. SourceFile may elect to perform the verification at its site or at Depositor's site. Depositor agrees to cooperate with SourceFile in the verification process by providing its facilities and computer systems and by permitting SourceFile and at least one employee of Beneficiary to be present during the verification of Source Material.

10. Restriction on Access to Software. Except as required to carry out its duties hereunder, SourceFile shall not permit any SourceFile employee, Beneficiary or any other person access to the Source Material unless consented to in writing by Depositor. SourceFile shall use its best efforts to avoid unauthorized access to the Source Material by its employees or any other person.

11. Bankruptcy. Depositor and Beneficiary acknowledge that this Agreement is an "agreement supplementary to" the License Agreement as provided in Section 365 (n) of Title 11, United States Code (the "Bankruptcy Code"). Depositor acknowledges that if Depositor, as a debtor in possession or a trustee in Bankruptcy in a case under the Bankruptcy Code, rejects the License Agreement or this Agreement, Beneficiary may elect to retain its rights under the License Agreement and this Agreement as provided in Section 365 (n) of the Bankruptcy Code. Upon written request of Beneficiary to Depositor or the Bankruptcy Trustee, Depositor or such Bankruptcy Trustee shall not interfere with the rights of Beneficiary as provided in the License Agreement and this Agreement, including the right to obtain the Source Material from SourceFile.

12. Notices. Any notice or other communication required or permitted under this Agreement shall be in writing and shall be deemed to have been duly given on the date service is served personally, one day after the date of sending by overnight courier, or five (5) days after the date of mailing if sent registered mail, postage prepaid, return receipt required, and addressed as follows or to such other address or telefax number as either party may, from time to time, designate in a written notice given in like manner:

TO BENEFICIARY:

TO SOURCEFILE:

SourceFile
50 Crisp Plaza
Suite 700
San Francisco, California 94124-2924
Attn: Ms. Bea Strickland
Telephone (415) 715-2733
Facsimile (415) 822-7322

13. Miscellaneous Provisions.

(a) **Waiver.** Any term of this Agreement may be waived by the party entitled to the benefits thereof, provided that any such waiver must be in writing and signed by the party against whom the enforcement of the waiver is sought. No waiver of any condition, or of the breach of any provision of this Agreement, in any one or more instances, shall be deemed to be a further or continuing waiver of such condition or breach. Delay or failure to exercise any right or remedy shall not be deemed the waiver of that right or remedy.

(b) **Modification or Amendment.** Any modification or amendment of any provision of this Agreement must be in writing, signed by the parties hereto and dated subsequent to the date hereof.

(c) **Governing Law.** This Agreement shall be governed by and construed in accordance with the laws of the State of California.

(d) **Headings; Severability.** The headings appearing at the beginning of the sections contained in this Agreement have been inserted for identification and reference purposes only and shall not be used to determine the construction or interpretation of this Agreement. If any provision of this Agreement is held to be invalid, illegal or unenforceable, the validity, legality and enforceability of the remaining provisions shall not in any way be affected or impaired thereby.

(e) **Further Assurances.** The parties agree to perform all acts and execute all supplementary

instruments or documents which may be reasonably necessary to carry out the provisions of this Agreement.

(f) Entire Agreement. This Agreement, including the attachments hereto, contains the entire understanding among the parties and supersedes all previous communications, representations and contracts, oral or written, among the parties, with respect to the subject matter thereof. It is agreed and understood that this document and agreement shall be the whole and only agreement among the parties hereto with regard to these escrow instructions and the obligations of SourceFile herein in connection with this Agreement, and shall supersede and cancel any prior instructions. SourceFile is specifically directed to follow these instructions only and SourceFile shall have no responsibility to follow the terms of any prior agreements or understandings.

IN WITNESS WHEREOF, the parties have executed this Agreement as of the date first above written.

DEPOSITOR

By: _____

Name: _____

Title: _____

BENEFICIARY

By: _____

Name: _____

Title: _____

SOURCEFILE

FileSafe, a California Corporation

By: _____

Name: Bea Strickland

Title: Director

[FILESAFE\B\SOURCE.pro]

Document Version Date: July 8, 1994

Document Print Date: July 7, 1994

EXHIBIT "A"
DESCRIPTION OF DEPOSIT

EXHIBIT "B"
SOURCEFILE COMPENSATION



TELEPHONE RESPONSIBILITIES: HAS
YOUR MIS TAKEN OVER YET?

Paper# 2005
By: Rick E. Hupe
From: Telenomics Inc.
31566 Railroad Canyon Road
Canyon Lake, Ca. 92587
800-328-1177

HISTORY OF RESPONSIBILITIES

In the late 1980(s), several court rulings from Judge Green's attempt to break up the Bell Operating Company monopolies began implementation. It changed the way many companies addressed their telephone responsibilities. Our paper will address the 'trickle down effect' from these court rulings and the uses of a Telephone Management Systems.

Background

For many years the telephone company took care of both home and private business telephone needs. When the breakup of the telephone monopolies began with the Judge Green court rulings in 1981, companies began picking up their own responsibilities previously handled by the telephone companies. The individuals assigned these new telephone responsibilities varied from company to company. They ranged from the telephone switchboard operator who may have received a promotion to become the Communications Supervisor to the Facility Engineer who was responsibility for the building(s).

With the telephone company break-up many new opportunities were created for new small companies. These type of services included the new coming of age for such products as voice mail, long distance traffic (such as Sprint and MCI), PBX manufactures such as (MITEL, Northern Telecom, etc.) and Call Accounting vendors.

In recent years, companies began looking at more complex systems then the Communication Supervisors and Facility Engineers were use to seeing. They were required to perform analysis on such things as least cost routing, traffic patterns, T-1 interfaces, data and voice integration, new procedures, proposals, etc., etc..

CURRENT TREND

Many companies today have given MIS the telephone responsibilities. It only made sense, the staff usually employed Analysts, Programmers, performed proposals and worked on major projects. They are experienced on installing new applications and the training of company personnel.

The logic of the PBX also being a computer becomes another factor in consideration of who should be responsible for the company telephone activities. In fact, if you put a PBX cabinet and a CPU cabinet together (contingent on manufacture), the boxes look almost identical.

When you typically think of PBX's, you picture a dedicated room with specialized air conditioning (generally locked up) with a PBX containing hundreds of wires running through the wall. Not so anymore. We are seeing new PBX's the size of tower PC's setting in the computer room controlling thousands of lines/trunks through fiber optics. Many companies even interface the PBX directly into the HP computer.

Other Considerations

The MIS departments were already responsible for the financial data on the companies main computer. This includes financial's such as general ledgers, receivables and sales automation. All modules which can benefit by the integration of telephone summarized data. Since their job was already defined to maintain systems within the various departments, logic became strong in assigning the MIS Department with the telephone responsibilities.

Problem

The biggest problem with this new scenario was while MIS now had this new responsibility, they most likely didn't want it. MIS personnel are use to responsibilities for data they can control. By utilizing the systems provided by the telephone vendor, they were completely dependent on the outside vendor. Additional problems included how to provide for high speed printing, back-up's, access by more than one user, interfacing to the financial modules and control of new requests. Retraining into a different operating and data structure also presented problems.

In situations where MIS inherited the companies telephone responsibilities, it became very frustrating with the small stand alone systems furnished by the telephone vendors. They were now in charge of a system that was inherent with problems and required much of their manual labor.

Another frustration associated with the small stand alone systems was the realization the company had invested large sums of money for their mainframe systems. These systems were already equipped with high speed printers, backed up every night, maintained high processing speed associated with large volumes of transactions, allowed for multiple users and were readily manageable by the existing staff. All the functionality required in a successful telephone management system. Having the telephone data on the mainframe had a lot of merit.

We have come to the realization over the past ten years, it's not really important if the MIS Department or the Telecommunications Department has authority over the telephone system, providing they work together and share resources. In a shrinking economy, this is becoming more commonplace. We have seen many instances where the MIS shares this responsibility with other departments.

What are Telephone Responsibilities and Who Cares?

There are a lot of departments which care about telephone responsibilities. The company telephone expenses generally makes up one of the top 3-5 expenses. It's a great area for potential savings and new applications.

A good place to become introduced to telephones expense is in Accounts Payable. Look at the number of telephone bills, their totals and try to figure out what all the numbers mean. It can be quite an eye opener.

SO WHO CARES

Accountant's

In most companies you will find at least three telephone traffic vendors or more. Consequently the company will receive at least three different telephone bills during different times of the month. In most cases, the bill will only consolidate to the assigned company telephone number, not the actual extension which originated the call (there are some exceptions when access codes are used). In addition to regular traffic/bills, there are credit card bills, cellular bills, person to person, off premises bills and all sorts of special item bills.

The Accountant (in most companies) have responsibility to verify telephone bills, consolidate them from the various vendors and charge back costs to the originating department. After the breakdown and consolidation has taken place the accountant will then make the appropriate journal entries into the General Ledger.

If you started with the various raw telephone bills in Accounts Payable, you can appreciate what a manual time consuming job this is for accounting each month.

The telephone company sometimes makes errors in their billing. Without an independent telephone bill, the company has little room to challenge the telephone vendor. It's usually, "send me a bill and we'll mail you a check".

In a company we recently came in contact with, the telephone company mailed a bill in error by approximately \$40,000. The company at first thought they were pirated (for the second time within a few months). It turned out to be a telephone company billing error which was caught because of the large variance. If the error was smaller, it most likely would have gone undetected and paid. In fact, it may have been going on for a long time.

With an automated telephone management system the bill can easily be audited. By doing such a company can challenge the telephone company, and most likely win.

Controller

The controllers job (among other things), is to control costs. The companies telephone expense is a good place to look. For example:

- The variance to budget by department is an excellent tool to manage department expenses during the month before costs get out of hand.
- The extension usage by department is another tool to manage the department managers in making sure they are managing their employees.
- The average cost by Carrier is an opportunity to see if there are financial advantages in changing long distance companies based on rates and discounts.

Department Managers

The department managers are the first line of defense in the opportunity to save company dollars. If either detail or exception call data is provided to the department managers on a regular basis (monthly) and discussed with the employees, saving will occur in two areas.

- #1. Productivity.. Let's assume the employee averages approximately 8 hours a week on the telephone. That's 20% of their payable time each week. If you think most employees don't spend that much time on the telephone, you may be kidding yourself. Some spend much more.

These types of calls can come in two forms, incoming or outgoing. If it's an incoming call you may not think it's important because the calling party is paying for the telephone call. However, your paying the employees salary, insurance, office expense, etc. to participate in that call.

It is also possible if the calls are incoming, the calling party is using your 800 number. If that's the case, your paying both the overhead, cost of the call and tying up the 800 line. **Do you know where the 800 calls go within your company?**

- #2. Hard dollars..

How many calls do you think are business vs personal?

- 1 out of every 4?
- 1 out of every 3?
- 1 out of every 2?

Let's assume it's 1 out of 3. What do you think that adds up to over a year.

Annual Bill	Potential Savings
\$80,000	\$26,400
\$100,000	\$33,333
\$150,000	\$50,000
\$300,000	\$100,000
\$500,000	\$166,666
\$750,000	\$250,000
\$1000,000	\$333,000

In many companies, the employee monthly telephone report is used as a tool in areas such as annual reviews. We've seen where employees with large volume of telephone calls are generally the same employees which are constantly behind on projects. We once discovered a high priced contract programmer (who was behind on his project) which came in and ran his Real Estate business out of the companies office. At their expense!

Receptionist

In many companies you will find a Receptionist responsible for greeting visitors, vendors, etc.. It's also commonplace to utilize these individuals as the switchboard operator.

Associated with incoming calls and visitors, is the requirement to identify and find company personnel. At times the Receptionist is only given a last name, first name or department for identification.

On-Line Directories have been created for the Receptionist(s) with short name searches by first name, last name and department. Many companies use this module to produce their in-house hard copy telephone directories and identify company personnel.

Sales Department

The Sales Department uses Telephone Management Systems to make sure enough calls are being made to support the sales activity. We know companies which use the system to determine who makes the most calls in conjunction with the most sales and the time of day these calls are being made.

One of the key features the Sales Managers utilizes in the system is assignment of a telephone number to key accounts. When the report is generated, it is sorted on company, illustrating the number of times the customer has been called and by which agent(s) on which day's.

Human Resources

It's uncommon for employee termination based on telephone calls, even though it does happen on occasion.

The Human Resources Department requests an up-to-date employee telephone report at times of termination. While the employee is being terminated for reasons other than telephone abuse, it's a good company policy to gather all information possible before a termination, including the record of telephone calls which may illustrate a lack of productivity on the employee part.

Telecommunications Manager

The Telecommunications Manager responsibilities may vary, based on how the various job descriptions are broken out.

We typically find the Telcom Manager responsible for dealing with the telephone carriers for local and long distance services, the PBX vendor in identifying calling patterns for the programming of the least cost routing software, monitoring trunk and trunk groups looking for broken trunks, under/over utilized trunks and potential toll fraud.

Toll Fraud

Toll Fraud is estimated for be a four billion dollar a year business. Many articles have been written on the subject, it was even a featured segment on the CBS 60 minutes program. The majority of the following information on this subject is taken from a Forbes magazine article in August of 1992.

The most common type of toll fraud calls are a result of a break-in through the company Phone-Mail System or 800 telephone numbers. Once in, the calls are routed through the company PBX system to places such as the Caribbean or Columbia. Most commonly, these calls are placed by organized crime and/or drug dealers. The break-in number is even sold on the streets. It's a big business.

The telephone bill comes in once a month. The Toll Fraud is generally discovered at this point because the telephone bill can be thousands of dollars higher than it's suppose to be, for example:

United Nations	\$1,000,000
Drug Enforcement	\$2,000,000
NASA	\$12,000,000

We know of companies at this convention who have been hit by toll fraud within the past couple of years. The theory is, either you have been hit by toll fraud, or you will be. If your not doing anything to protect yourself, you will most likely be responsible for the costs if/when toll fraud hits your company.

There are many ways companies are installing toll fraud capabilities. These vary from on-line alarms searching for short frequent calls coming into the system which will in turn set off an alarm, to the printing of reports based on set parameters should toll fraud patterns occur.

Toll Fraud capabilities are now being installed as a standard feature on some of the newer PBX's and can also be incorporated as part of a good Telephone Management Software System.

**With all this Sophistication
are we becoming Big Brother?**

The company has a right to manage it's resources. The Telephone Accounting Systems automates the Accounting Department, gives the Telcom Manager the tools to do his/her job, helps the Controller manage costs, provides the Receptionist with on-line directories, enables Sales to track important calls which can assist in sales and provides data for the Department Manager and Human Resources in their management responsibilities.

It's true the system does monitor calls to reduce costs. Let's compare this to an employee who wants to obtain a office calculator. First, they have to fill out a requisition form, then they have to get their manager to sign off, then it's sent to purchasing where it go's through paper work and then through the procurement cycle. Finally it's purchased and may go to receiving.

What if that same employee wants to make a \$100.00 telephone call and no system is in place to identify the call back to the originating party. Basically, all they have to do is pick up the phone and make the call. The bill will be embedded with thousands of other calls back to the main company telephone number, not the party who originated the call. It's easy.

How's all this done

The PBX and/or CENTREX system puts out a raw data record. It contains some basic information, such as:

- Extension
- Duration
- Trunk
- Time of call
- Date of call (maybe)
- Authorization Code (if used)

The system captures this data via an RS232 connection to a buffer devise which is 'POLLED' into the HP/3000 for editing, costing and record standardization.

The data base contains the company chart of accounts including employee, extension, account code, department, division, mail stop, etc..

The data base is also built to contain a trunk roster defining the local and long distance carriers, their rates, discounts, local dialable points, and significant costing data required for the costing algorithms.

V&H is another table within the system which is used for milage calculations for costing and the identification of city and state for each call destination. This file will contain over 60,000 entries. It will go through many changes during the year, especially when a prefix code split occurs.

The call detail record is determined to either be valid or rejected. If rejected, it will go into an error recycling file for clean up or termination. The valid records (almost all records on new systems) will then be costed, combined with the personnel data, identified to city and state and placed in various data sets within the data base for future processing.

**Who Typically uses
these system**

- Newspaper Clients

Manage their network and pay bonus to sales for number of calls placed for classified ad's and overdue accounts

- Marketing Companies

Need to utilize account codes and charge back costs to projects/clients

- Universities and Colleges

Manage their network, control administrative costs and perform student billing

- City School Districts

Charge back telephone costs to individual schools from the various telephone carriers

- Convention Centers

Produce telephone bills within 15 minutes of a vendor terminating the booth

- Attorney Firms

Bill back clients

- Any type of company which wants to manage their network and control costs

Why not bills from the Long Distance Companies?

Some Long Distance Carriers are now supplying a partial bill. However the features they most likely cannot provide are:

- Reports on Demand
- Long Distance Overflow
- Incoming
- On-Line Directories
- PBX Traffic Reports
- On-Line Telephone Bill
- Personnel Name and Department
- Consolidated Telephone Bill
- General Ledger or Sales Automation Interface
- Ad-Hoc Reporting Capabilities
- Control of Your Own Data
- Variance to Budget
- Toll Fraud Capabilities
- Ability to Easily Change Vendors

In summary, the long distance and/or local carriers will supply a bill for the traffic they carry on a monthly basis with very few options.

Why Do We Have PBX's and CENTREX Systems



One option available is to have individual lines into the office like we do at home. These lines are known as trunks and have monthly charge of \$20-\$25 per line. They do not have the capability to dial from extension to extension as we do with our PBX or CENTREX systems.

In order to make internal calls with individual lines, we would have to make an outside call in order to make an inside call. This would not be cost effective. We could not transfer incoming calls or easily interface data via the network. Consider 500 lines/trunks at \$20 mo. each. That would be a service charge of \$10,000 per month!

The PBX and CENTREX offerings do more than just reduce costs and offer internal telephone dialing. However, we're just addressing the basics.

The company with a PBX of 500 telephones would more than likely have 100 - 125 trunks to service the 500 extensions. The monthly savings on dedicated trunks will more than pay for the PBX in a few short months.

Internal calls, voicemail, least cost routing, data links, toll fraud, directories, DID Lines and a host of other features are offered with new PBX Systems. Most of these features are not feasible without a PBX or CENTREX System. Being able to identify and modify long distance carriers can also generate great future savings.

Why all the Current Activity with PBX Vendors

We are seeing a lot of activity within the PBX market because of the new North American Dialing Plan, the reduction of size, features, maintenance and price in the newer PBX's. Some of the existing older PBX's will not handle the changes going on in the industry today.

**Who's got the Best
PBX on the Market**

Good question and one we would not get into. We work with all of them and most certainly have our favorites. Some of the top vendors we work with include:

- MITEL
- ROLM
- NEC
- NORTHERN
- FUJITSU
- SIEMENS
- AT&T

Summary:

Telephone Management offers a variety of choices and can effect many departments if correctly implemented. It's one area MIS has tried unsuccessfully to keep away from.

We are finding MIS inheriting telephone responsibilities in companies all over the United States. We are also finding a new approach being taken when MIS and/or Telcom work together on the project. It's being treated analytically as an application, and it's working.

Presentation Number 2006
Imaging and Document Management
By David W. Sborov
NSD, Inc.
1400 Fashion Island Blvd.
San Mateo, California 94404
415-573-5923

Electronic Document Management software is necessary new software for the efficient and productive organizational environment. The paperless office is becoming more of a reality. And, studies have shown that up to 80% of the labor, operating and capital outlay expenditures associated with the storage and retrieval of paper and microforms can be eliminated with electronic document management systems.

However, most people do not really understand what electronic document management is, the difference between document management and imaging, the key technical requirements and management considerations, or how to implement document management or imaging within their organizational environment. Further, most people do not know what capabilities will be required in their electronic document management software for it to really be an effective, cost saving, information tool.

Any of the above items could become a paper of its own. Electronic document management is a big topic, and may affect, in a positive way, every aspect of your organization. This presentation will outline many of the basics as well as key requirements that must be considered.

THE PAPER PROBLEM

To understand the need for electronic document management, it is first necessary to understand the paper and information problem. Although desktop computing has simplified and made business transactions and communications more efficient, this new technology has significantly increased the use of paper. As we have moved closer each year to the paperless office, the use and amount of paper used has actually accelerated each year.

Electronic information distribution technology continues to expand, with e-mail, multi-media, groupware, and application programs for generating and viewing data base reports, etc.. However most documents and almost all of the computer generated information is still being printed, (usually multiple times) before being distributed within the organization. It seems like everybody either has or wants a printer.

U. S. business annually use more paper now, as compared to any other year. Because most managers are concerned with meeting their business objectives, they do not pay much attention to the paper problem. They must rely on the existing printing and information distribution infrastructure which has traditionally been paper intensive and lacks any information and print distribution strategy.

Think about the amount of paper your organization consumes, and its attendant cost and burden. Printing and information distribution costs are the silent, growing expense within your organization.

As an example, according to TENEX CONSULTING, "Over 324 billion paper documents are handled by U.S. businesses each year, at an average cost of 25 cents per document." Stated in another way, according to NETWORK WORLD, U.S. Businesses created 92 billion original paper documents. And the average business document is copied 19 times during its life. TENEX CONSULTING says " Information management consumes up to 8% of a company's revenues, up to 40% of an office's labor costs, and up to 60% of the average white-collar employee's workday." For most businesses, document management is their largest expense.

WHAT IS ELECTRONIC DOCUMENT MANAGEMENT

Electronic document management is a information distribution and archival strategy to reduce your organization's reliance on paper and other microfilms and to continue increases in productivity. All organizations will embrace electronic document management at some time in the relatively near future. The motivating factors will be to drive down the cost of doing business, manage the increasing flow of internally generated and externally generated information, increase efficiency, and meet current ISO 9000 requirements.

Electronic document management software is an integrated information management system that combines existing, separate information processing systems into a powerful electronic tool for storing, retrieving, transmitting, managing and sharing information in an electronic format using desktop PC's and servers.

Although reduction of paper is one of the most obvious benefits of electronic document management, there are many others. These include fast document retrieval rates; sequencing, tracking and reporting of documents; extensive and automatic indexing and cross-referencing, excellent document reproduction capabilities; document integrity; and reliable security.

The hardware components of an electronic document management system include a server or CPU, optical storage device or autochanger, scanner, and desktop PC's or workstations. The server and desktop PC's are networked together. Depending upon the document management vendors software, the environment can be closed or open. A closed environment is associated with stand alone hardware and the vendor's proprietary software. Open systems tend to be client server based systems able to run on different vendor platforms.

Electronic document management benefits are realized through fulfilling end user application needs. However, user application needs can be vastly different. For example, the application needs of the corporate finance department are going to be different from the application needs of the corporate engineering department. Thus, a certain amount of system flexibility must be incorporated into the document management software

Obviously with such a broad requirement to satisfy end user application needs, a document management system must have a number of different capabilities. These capabilities include Document Archiving, Compound Document Control, Document Version Control, Text Indexing and Retrieval, and Work Flow. Unfortunately most people who perceive the need for electronic document management do not fully understand what these terms mean especially in light of their end user application needs.

What are these terms and what do they mean?

Document Archiving - This function is an organized and electronic method of storing and retrieving documents within an organization. Because documents are in an electronic format and are easily reproduced and moved around without the possibility of being damaged or misplaced, most organizations store their archive in one place as opposed to having duplicate copies in filing cabinets in multiple locations. Some organizations will however, create back-up optical disks or tapes to be stored at some other remote location for disaster recovery purposes.

Compound Document Control - Allows users to control a document or a file that may consist of many different elements, such as a spread sheet, a graphic, and text. As changes or additions are made to the original element, in its original program, those changes flow through to the final document.

Document Version Control - This function allows users to track the status of documents that a number of users may alter to ensure that necessary updates are not made to older versions of the document or that older versions of the document can be retrieved for historical reasons. This capability is especially useful in proposal generation activities where more than one person or group is working on a document at the same time.

Text Indexing and Retrieval - This is the most important function of a document management system. It not only allows you the ability to find and retrieve a file or document, it also allows information contained in a document to be read and indexed. Indexing and retrieval enables end users to find and access information that pertains to a particular subject, regardless of the document type. The more functionality and extensive indexing and retrieval capability that is included, the more flexible and useful the document management system will be to the end users. The more advanced systems will have an external indexing feature that will allow you to find a document or part of a document without having to enter the system and manually searching index fields.

Work Flow - This function pertains to the specific control of files or documents as they are used by a group or department within an organization. Workflow is based on routing and dependencies. It can be simple or complex. Workflow can be used to control the routing of documents to specific persons or groups for specific actions. It may also control the pace at which documents move through the organization to ensure that they are processed within certain time guidelines.

ELECTRONIC DOCUMENT MANAGEMENT AND IMAGING

Most people think that electronic document management is imaging. However, imaging is only a small part of document management. It is an important technology, but may not be required for every application. In fact in a narrow definition, imaging refers only to the act of scanning and inputting a document into an electronic document management system.

Imaging has received the largest amount of attention in the electronic document marketplace. This is because it becomes easier to cost justify the document management investment if one can demonstrate the reduction in paperflow. Thus, those companies with huge in-bound flows of paper such as insurance companies and banks, found making a document management decision easy, based on turning the in-bound flow of paper (non-computer generated documents) into electronic documents.

However, more than two thirds of all the paper in an organization is created as an out-bound document or as a computer generated document which was printed, probably multiple times, before being distributed. Although this paper flow is easily recognizable within the organization, it's cost is spread across multiple budgets in multiple departments. Thus no one person, department, or function owns all the paper. The Information Managers or Data Processing Managers most times know about the burden of out-bound or computer generated paper, but are powerless to do anything about it.

In certain instances, imaging can be a very important part of document management. But it should only be viewed as a way to input information into a document management system.

For example, in a insurance claims processing department, or in a mortgage origination environment, where large numbers of incoming documents (usually paper) must be associated with computer generated output, imaging is a requirement. Or in other cases, where there is a filing cabinet being used on a day to day basis, imaging might be very appropriate so that the contents of that filing cabinet can be incorporated into an electronic document management system.

In many applications, imaging is not a requirement for electronic document management. In those industries heavy into transaction processing, or those that rely heavily on internally generated computer documents or that generate large volumes of COM (computer output microfiche) output, electronic imaging is not important to end user applications.

Work flow applications in electronic document management may or may not rely heavily on electronic imaging. In those applications where there is a heavy in-bound concentration or flow of paper documents, and a strong dependence on that information, such as mortgage loan processing or insurance claims, imaging becomes very important to the workflow process. But in industries where internally produced (i.e. computer generated) documentation is mission critical, imaging becomes a secondary priority and may not be important at all.

Electronic imaging can be a relatively expensive addition to an electronic document management system. Depending upon how elaborate the system, per seat incremental costs can be as high as 80% of the base document management software price. Thus, one needs to be sure that the in-bound paper flow (non-computer generated documents) is truly significant as opposed to the internally generated documents, which most likely were computer generated and at one time already in an electronic format.

ELECTRONIC DOCUMENT MANAGEMENT SYSTEM REQUIREMENTS

The key requirement for fully functional electronic document management systems is that they must be able to integrate both computer generated and non-computer generated documents into one cohesive solution. Only the non-computer generated documents must be imaged. And, a versatile document management system should be able to import images and data from more than one electronic imaging system or computing environment.

Electronic document management systems must be flexible and easy to use. They must reflect organizational diversity and recognize different application needs. They must utilize or at least co-exist with information processing and distribution systems that currently exist within an organization.

Such existing information systems may be composed of one or more of the following:

Legacy or host computing systems

Client/Server platforms

PC's and Workstations

The mail room and other existing systems

There is also an emerging requirement for electronic document management systems to have open architecture and interoperability. These concepts are especially important given the pace of new technology and the growth and power of client/server computing. The issues that must be taken into account when evaluating an electronic document management system include the following:

Cross-platform availability - Will your chosen computing environment be fully supported by your present hardware or software vendor, or be supported by your electronic document management vendor, or will it be supported at all?

Integration with e-mail, work group & workflow systems - Will new workflow, work generation, and messaging products be able to integrate into your electronic document management system transparently?

Backup - The use of electronic document management software can complicate the back-up procedures due to information being stored at potentially many disparate sites. Do you need to simultaneously create backup for disaster recovery or security purposes?

Scalability - Will your electronic document management system have the ability to grow with the addition of new clients, have the additional indexing and storage capacities that may be needed in the future, have the necessary performance or the ability to grow across your organizational enterprise?

Interoperability between document management systems - Interoperability is becoming a big issue. In the future the need for electronic document management systems to communicate with one another will be essential. Will your system be able to communicate with other archival and storage systems?

Standards - Along with interoperability comes the standards question. What standards is your document management vendor using. Is your vendor a member of the Shamrock Group or plan to adhere to their proposed standards?

Links to billing - Documents and information equate to dollar revenue for many organizations. Will your electronic document management system be able to communicate and import billing information to your accounting or other systems?

OTHER MANAGEMENT CONSIDERATIONS

In addition to the above technical functionality and requirements, there are also management considerations which must be considered. Some of these considerations include:

Applicability to your Environment - How is information generated now in your environment. Is there a heavy concentration of in-bound or non-computer generated documents that need to be imaged? How many documents are now being computer generated on either PC's or in a host computing environment? Will workflow be a necessary requirement or will simple on-line viewing satisfy your needs? Is your environment now networked? How many PC's do you have as opposed to terminals?

Cost Justification - How much paper is being used on a weekly or monthly basis. How many printers exist in the organization. How many file cabinets are there? How many file cabinets contain duplicate files and documents? How many people are involved with handling paper?

Legal Issues - Can all of your information be stored in an electronic format? Legal concept of "only and best copy".

Workflow procedures - Is the document management system flexible enough to satisfy the workflow requirements of different organizational functions? Will a tool kit or the writing of additional script be necessary to customize workflow from one department to another? Should a separate workflow software package be considered?

Vendor Selection - Does the vendor offer a standalone or open electronic document management system? Is the vendor image oriented, or computer output oriented? How long does it take to install and configure the vendor's document management system. How much customization will be needed. Does the vendor talk about "Business Process Re-Automation" and propose a services contract along with the document management software? Is the vendor's software compatible with the existing computer environment?

It is important to fully investigate past implementations of your selected vendor's document management software. How long did it take to install and then implement. How much will this phase cost? Is the cost of installation included with the software? How much training will be needed for both the users and system administrators? The answers to these questions are important and may surprise you.

The implementation of an electronic document management system requires careful planning and clearly stated system objectives. Your document management software vendor will help you in the planning and implementation of the system. They should have questionnaires, guidelines, procedures, and manuals that will help you through this process.

The timing as to when an electronic document management system is implemented is also important. The success or failure of this technology, and the return on your investment is totally dependent on the end user's acceptance and use of the system. Thus, the timing of implementation becomes an important step to success.

The best time to implement electronic document management is during a migration from one system to another. This may happen when moving from a host or legacy system into a client server environment. This is an ideal time to teach users new habits and the benefits of document management while adjusting to new computing systems.

Another implementation concept is the use of pilot programs. Pilots are valuable tools to introduce new technology into an organization. Usually there is a group or department that is anxious and willing to try the new technology. The successful implementation of electronic document management means that virtually every person, document and work process in the organization will be affected. Changes in work habits, processes and even communication will be affected. Therefore, access and information regarding the pilot must be made available throughout the organization.

WHAT DOES ALL THIS MEAN

What does all this mean to a potential electronic document management system user. With electronic document management you will save money and your organization will become more efficient. Information and documents will be located and accessed in seconds. Expensive printers and copiers will be eliminated because the information will be available on your desk top PC. And, filing cabinets, especially file cabinets containing duplicate documents will be a fixture of the past.

Most organizations who have implemented electronic document management systems have realized significant returns on their investment. You will also, if you methodically analyze your needs, and critically evaluate your software vendor.

Before you make a document management software decision, be sure you have validated your real needs and have the ability to critically analyze the vendor's advertised capabilities. Know the intended user applications. Are you evaluating a imaging system or a true document management system? Do the capabilities really match my application needs. Are documents scanned into electronic format the highest priority?

Many companies in the last few years have purchased imaging systems, only to discover after implementation that what they really needed was a full blown document management system.

Paper Number: 2009

FUTURE STRATEGIES FOR D.P. MANAGERS

Roger Lawson

Proactive Systems Inc., 4 Main St, Los Altos,
CA 94022 (Tel:415-949-9100)

Abstract:

The author gave a number of presentations a couple of years ago on strategic issues facing computer system managers. The issues included: whether to move from proprietary systems to UNIX or other open operating systems, whether to convert to SQL databases and associated 4 GLs, whether to downsize from mainframes and minis to PCs, whether to move to client/server from host based systems, which GUI to select, etc.

This paper will focus on the current status of some of those issues, and consider new topics such as the impact of object databases, the NT operating system, PC and host integration, etc.

The aim of the paper is to assist D.P. Managers to make decisions about the directions of their own companies MIS operations in the context of minimizing costs, while maximizing service. Migration paths for existing legacy systems will also be discussed.

Biographical note:

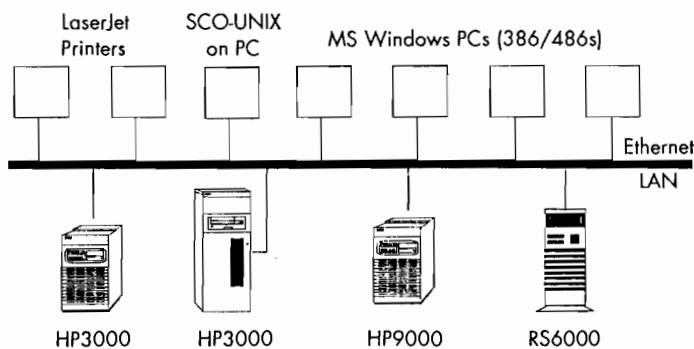
The author has a Masters Degree in Business Administration and over 20 year experience in the data processing industry as programmer, systems analyst and D.P. Manager.

Future Strategies for DP Managers 2009-1

A couple of years ago, I gave a number of presentations concerning the strategic issues facing M.I.S. managers at that time. It covered issues such as whether to move from proprietary operating systems (such as MPE) to UNIX, whether to downsize or not, future database strategies, client/server implementations and other topical matters. This document re-visits some of those issues and looks at some new ones.

My background is that I started my DP career as a programmer on IBM mainframes, worked for a couple of software companies on both mini and mainframe systems, then acted as MIS Manager for an HP3000 installation for a few years, and am now President of a computer software firm which specializes in printing solutions and database tools for mid-range computer users. We develop software on both HP9000 and HP3000 systems. We also have IBM AIX and SCO-UNIX systems in-house and have ported to other environments such as Sun Solaris, IBM AS400, etc. See Figure 1 for our current network configuration in one of our offices (and we have similar set-ups in our other offices worldwide). As you can see, we are well on the road to distributed, open systems.

Figure 1: in-house network



As a \$5 million revenue business, we also have significant operational systems (financial accounting, order processing, marketing databases, etc) on HP3000 systems so what to do with these "legacy" systems is not solely of academic interest to me. As you may have guessed, these systems are mainly

written in COBOL, IMAGE and V-PLUS, but most of our new development work is in "C" on UNIX.

Compatibility and Supportability is Everything.

The only companies I know of who ever decided to scrap their existing application software and start from scratch either had very poor systems, or suicidal DP Managers. Even in a relatively small business such as ours, the investment in existing software is enormous, and even if we could find suitable software packages to replace existing systems the cost of implementing new systems, data translation, staff retraining, and associated work, would be very expensive and require a large amount of management effort (the least available commodity, as in most companies). Some of our existing systems are almost ten years old, but the HP3000 and the MPE operating system have been wonderful in terms of maintaining compatibility and minimizing non-essential rewrites (we have had 4 hardware changes in that period). The systems are also easy to support - it's still easy to hire HP3000 COBOL programmers even if we didn't have the expertise in-house. IMAGE has also been a very reliable, stable and high performance database. As we hold key operational and financial data in our databases, this is a prime requirement.

However our systems are looking somewhat old fashioned. Even though almost everyone has a WINDOWS PC on their desk, they emulate dumb terminals to access our host operational systems, a situation that I would bet is not uncommon in your own companies! Users have become familiar with using PC software applications under MS-WINDOWS (MS-WORKS, MS-WORD, COREL DRAW and our own FANTASIA and FLEXIBASE/SQL products are some of the PC applications we run). They are looking for an improved user interface on existing host systems at the same time as Hewlett Packard and other major vendors are trying to sell us the concept of client/server. How do we move into the future and where do we aim for?

HP3000 Database Issues

(Skip this section if you only use HP-UX). Twenty years ago one of the reasons I bought an HP3000 was because it was more compliant with industry standards than most of the mid-range competition (standard COBOL development language, etc). I'm basically lazy - I don't like to have to learn a new language and nor do I want to take the risk of having to rewrite existing systems unnecessarily at some future date because the supplier has gone out of business or the software tools I chose have gone out of fashion. At that time, there was no industry standard database system, at least not on minicomputer platforms, and the standard there was for network type databases never really took hold on mainframes even. We are living in a

different age now. Relational databases based on SQL are the industry standard backed by ANSI, ISO and all significant other standards organizations. They are also available on every single computer hardware platform you can name.

IMAGE on HP3000s has been a great product but it ran into an evolutionary dead-end a few years ago. There was no way it could support the distributed database and client/server capabilities that were becoming available for SQL databases without a major investment from H/P (which they were unwilling to do because of their commitment to open, standards-based, solutions).

Thank god that H/P came up with a solution which is called IMAGE/SQL and have effectively given it to everyone for free. With one enormous leap, those 30,000 IMAGE database users worldwide now have the ability to treat their IMAGE databases as SQL ones so far as data access is concerned. Also you can now gradually migrate your software to using SQL calls rather than IMAGE calls, and you can write new software that uses those SQL calls which can later easily be moved onto "real" relational SQL databases such as HPs ALLBASE/SQL, ORACLE, or SYBASE. In addition, because H/P have also given you HP ALLBASE PC API and ALLBASE/NET, you can use any PC product that supports the GUPTA database API (or ODBC soon) so all those user-friendly WINDOWS products for decision support, report writing, graphics, spreadsheets, and even complete 4GL systems become available for use! You also get the distributed database capabilities of ALLBASE/NET to give you transparent access for both ALLBASE/SQL and IMAGE databases across HP3000 and HP9000 systems.

Now everybody sees this as a "good thing". The IMAGE die-hards can continue to use IMAGE and if you can't justify changing application code, you can leave it as is. However where I disagree with some pundits is whether you should continue to develop new systems in IMAGE/SQL and how soon you should migrate to relational databases.

There are three problems with IMAGE/SQL:

- It's not an industry standard SQL implementation. For example, it only contains the data access and not the data definition language (even if this was fixed, there are other differences).

- There is a performance overhead in mapping IMAGE onto ALLBASE/SQL. You would get significantly better performance by using ALLBASE/SQL as a native SQL database. Don't believe those folks who say IMAGE is faster than relational databases, the opposite is true in my experience if you use ALLBASE/SQL wisely.

■ You have to know both SQL and IMAGE to be able to optimize performance and manage your IMAGE/SQL databases, ie. you need two types of expertise rather than just one, so it's more expensive to maintain and manage.

For those reasons I see IMAGE/SQL as a migration path and a way of supporting older systems, but I would write new applications using ALLBASE/SQL or other proper relational database management systems.

HP ALLBASE/SQL?

ALLBASE/SQL in its latest incarnation is a reliable, high performance, high quality SQL implementation. After some initial uncertainty about H/Ps commitment to the product, they are now much more positive and have a good number of successful users on real operational applications on both HP3000 and HP9000 systems. Should I use ALLBASE/SQL rather than some other proprietary DBMS such as ORACLE, SYBASE, INFORMIX, INGRES, etc? The same arguments apply whether you are using HP3000 or HP9000 hardware and can be summarized as follows:

■ HP ALLBASE/SQL is low cost, and is in fact a fraction of the price of some other DBMS products.

■ H/P has, and probably can continue to optimize its performance on PA RISC architecture, and MPE-IX and HP-UX operating systems, better than can the other database suppliers.

■ If you go to a proprietary database company and use their 4GL products on a UNIX box you will be "locked in" more effectively than you were by selecting an HP3000 from Hewlett Packard. At least with H/P their ethics have generally been unquestionable and you may not get the same treatment from other suppliers.

Incidentally if you don't use the database vendors 4GL (and more on this later) you can achieve good application portability across different SQL databases, so buying ALLBASE/SQL doesn't necessarily lock you into H/P. However, as with COBOL, you have to avoid those vendor specific extensions that everyone likes to add.

■ HP3000 users who want to use IMAGE/SQL will find it easier to "mix and match" IMAGE/SQL databases with ALLBASE/SQL databases than with other databases and will minimize expertise needs by going that route.

■ Software application package vendors may find it easier on the basis of multiple hardware platform support to implement on ORACLE, SYBASE,

INFORMIX, INGRES, etc, but it may result in a more expensive and less efficient solution than using ALLBASE/SQL on HP systems.

Database Vendors 4GLs or Client Server?

I covered the issues of host database selection above, but do you purchase the database vendors 4GL, or some other solution? I would go the latter route, for the reason that data has a longer life than software development languages. You can, and often do, write an application program, throw it away later and replace it by a new program (which may be written in a different language and even run under a different operating system). If the core business remains the same, data is never thrown away and totally re-entered. Therefore the application database tends to have a longer life, even if its form adapts over the years.

In practice SQL databases are likely to be the commercial standard for storing data for at least the next 20 years (personally I don't expect object orientated databases to be a significant proportion for reasons that can't be covered in the limitations of this document, and there are no other candidates on the horizon - in any case object databases such as H/PS OpenODB are based on core SQL databases).

However the development tools are likely to change significantly in that period. There are lots of good, client-server orientated development tools available or coming soon. For example, GUPTA SQL-WINDOWS, POWERSOFT POWERBUILDER, COGNOS POWERHOUSE CLIENT/SERVER, etc. Other solutions if you want more control (and lower per seat run-time cost) are to develop in MICROSOFT VISUAL BASIC/C++ or some other PC software toolkit and connect to the host system via various database APIs, or via the DCE software, or via Berkeley Sockets (soon to be available on both HP3000 and HP9000) - there are also "middleware" products that can simplify connection to many different host systems.

Development tools are changing and improving rapidly so the best tool now may not be the best tool in 5 years time. Also the choice of which development tool to use is much more dependent on application complexity, data volumes, development lead times, development staff expertise, cost constraints, etc, than is the choice of database. I would therefore choose a separate client/server front-end development tool from the the host database product.

In my view SQL DBMSs are a commodity and the lowest cost, reliable product on the chosen hardware platform is the best choice (if the selection is inappropriate longer term, or I want to move to another hardware platform, I can change my choice later).

Client/Server Issues

So far as end-users are concerned "the user interface is the system". Most users are now familiar with an MS-WINDOWS user interface, so any client/server product we acquire must use that interface. It seems to me that Microsoft have won the desktop war already and that COSE, H/P, et al, will not succeed in persuading people to run UNIX on the desktop or replace those cheap PCs with expensive workstations. However with a bit of luck we won't all be in hock to Microsoft in future, because of possible anti-trust action against Microsoft in both the USA and Europe, and also because WABI support and other similar products will enable us to run our MS-WINDOWS compliant software under other operating systems and on other hardware. There are also some software products that enable you to produce cross-platform front-ends to run under MS-WINDOWS, X-WINDOWS, OSF-MOTIF, OS/2 PRESENTATION MANAGER, etc (eg. VISIXs GALAXY or NEURON DATA's OPEN INTERFACE), but these are relatively complex to use and I think they are more relevant to software package developers than end-user application developers.

By moving to client/server based systems in a LAN environment, I can build very flexible, enterprise wide systems, and I couldn't care whether the host databases are on HP3000, HP9000 or other hardware. I can use SQL to glue all the systems together.

I would still choose one of the traditional minicomputer systems for the host server because of better reliability, recoverability and back-up systems and better top-end transaction performance than most other PC network servers. In fact the HP3000 is somewhat better than the HP9000 in that regard (the MPE-iX transaction monitor is much better at protecting and recovering my data files than HP-UX although most SQL implementations provide in-built recovery anyway). Note that implementation of ENCINA or IBM's CICS as standard transaction monitors will provide distributed transaction integrity if you are fully distributing your processing.

Will I save money by moving to client/server systems from conventional host based ones? Probably not from what we have seen and what has been reported by other users. The software is cheaper but the complexity of supporting LAN environments results in higher technical support overheads.

Will I get the performance I traditionally get from host based applications (such as IMAGE/COBOL/V-PLUS on HP3000s)? The jury is still out on this. I have only come across a few high volume, real commercial, transaction processing client/server implementations. Most implementations that H/P and the software vendors talk about are low transaction volumes, small data base sizes or specialized applications (eg. supporting managers doing

occasional activities rather than a mission critical application such as order processing). You need to look at any software vendors offerings very carefully in this regard.

UNIX versus MPE and Client/Server

UNIX is primarily a character orientated I/O operating system, whereas MPE has been optimized for block-mode transactions. Block data handling is much more efficient and enables you to handle many more transactions per hour in a commercial environment!

Most of the UNIX TPC benchmarks you see published are totally unreal because they are set up using I/O generators which do not match the way host applications are typically developed on UNIX. Comparisons of HP3000 and UNIX systems as host transaction processing systems are therefore fraught with difficulty. Also putting character mode, client/server front-ends onto an HP3000 system is not likely to result in high throughput!

The real transaction throughput of applications developed in your selected development system and the selected network environment is therefore one issue that you have to look at very carefully. Your aim should be to do as much as possible at the client level, and only use the host as a database server. The objective is to use those low-cost PC MIPS as much as possible. However you should use SQL stored procedures on the host where possible to maximize activity for each SQL I/O transaction and to ensure integrity and consistency within the host database.

Re-engineering Terminal Interfaces

If you want to take your old HP3000/V-PLUS programs and convert them to MS-WINDOWS, can you do that? Well apart from V-PLUS WINDOWS (which H/P seems to have stopped talking about), there are at least three products (one from Hewlett Packard, two from other vendors) that claim to enable you to re-engineer your front-end. The result tends to look prettier than a V-PLUS screen but they don't operate like most WINDOWS applications without a lot of customization work. As at the time of writing, I find it difficult to recommend any of them as a quick and easy solution, but that may change rapidly.

There are also products aimed at converting character mode I/O or screens (eg. UNIX CURSES menus) but I am not very impressed by those either. It's even more difficult to re-engineer such software (at least with V-PLUS you have a limited menu domain).

Host Processing and Development

On most commercial systems, you not only have on-line updating, you also have some "job" or "background" processing requirements, ie. weekly statement runs, volume invoice printing, etc. These activities need to be developed in a suitable high performance language, the jobs need scheduling and managing, and the output print handled and routed to a suitable printer. These processes should run on the host system and not on a client (although they may be initiated on a client) because of the problem of excessive network traffic if you attempt to shift all the data down to the client and do it there. A conventional host 3GL may still be a good development choice here.

How does HP-UX compare to MPE in these areas? Certainly UNIX provides a more flexible, richer and more productive development environment. However it is poor at job management and spool file management in comparison with MPE, although you can buy add-on software on both systems to improve them as neither matches the facilities available on mainframes. The support of network connected printers is much better on UNIX as H/P haven't yet figured out that HP3000 users want to plug their LaserJets into a LAN and print to them from the HP3000 spooler. Despite many people telling them this was required some time ago, they still leave it to third parties like us to provide a solution to this need.

Printing will no doubt become a LAN based, distributed solution, apart from very high volume applications. In either case, laser, inkjet and other non-impact technologies will become the norm.

Although HP-UX has a good reputation for reliability and stability in comparison with other UNIX flavors, HP-UX is still however not as stable and consistent as users of MPE have come to expect. For example, we have had problems with running programs compiled under one version of HP-UX on another, and where there was no workaround at all, ie. the program had to be re-linked on the operating system version it was going to run on. The only time I can remember this kind of issue coming up on MPE was when H/P changed the COBOL run-time libraries and there was a simple workaround in that case. In addition, H/P have recently decided to remove NetIPC support from HP-UX, which is rather like removing part of FOS from MPE and then expecting the users to rewrite their software to get around the resulting problems! However it does appear that HP-UX is becoming more stable.

PC/Host Integration

UNIX can provide a much better level of integration between host and PC file systems (including printer device sharing). There are several PC products on the market that extend the UNIX NFS (Network File System) to the PC level so you can share disc files, back up files, load PC software from the UNIX system and share network printers. We purchased and use the PC/TCP software from FTP which I can recommend for this purpose.

In comparison the HP3000 is really backward in data communications and open file systems. No NFS and you have to buy NS to get a limited version of TCP-IP networking support!

For the reasons mentioned above, and for purely fashion/political reasons, I would tend to favor HP-UX over HP3000 if I was buying a new system from scratch. However it does depend on transaction volumes and application characteristics as to which is the better solution. Also if you already have an HP3000 system, the effort to migrate may be more than it's worth (having mixed MPE and UNIX systems is not difficult if you have a large enough site to justify expertise in both operating systems, and HP9000s and HP3000s are object code compatible at the link level, plus use the same compilers so moving code between them is relatively easy).

Where do LAN Operating Systems Fit In?

To move to client/server you probably need to be LAN based for performance and transparency reasons, and even then you have to be careful about the LAN loading which is why higher capacity standards are being discussed. Should you look at NOVELL or LAN-MANAGER in addition, so as to provide integration of PCs and printers? We tried this (by installing NOVELL on both our HP3000 and HP9000 systems) and found it complex, messy, expensive and unnecessary bearing in mind the aforementioned solution.

Where does that leave NT and MPE?

MICROSOFT NT is claimed to be a "coming" operating system but I just don't see that happening on the mid-range. Who needs another proprietary operating system when nobody has explained why it will be any better than MPE, VMS, OS/400 or UNIX, and it will certainly take at least another 10 years to become as reliable and fully featured as those operating systems?

MPE's future is only assured if it retains its current strengths as a host database server operating system, while overcoming its weaknesses. The easiest way for it to do the latter is to become more like UNIX (eg. in the

data communications area) and that is exactly what H/P have been working on. With POSIX on MPE we are partly there but it's taking a long time to arrive unfortunately, and there is much more to do yet.

A Practical Client/Server Implementation

As you may have gathered, I personally feel that client/server is the route to go, and my company has decided on this as a strategic direction for its software development. For the reasons explained above, when we decided to enter the market for SQL database management tools (we already had a product for IMAGE database management now called FLEXIBASE/IMAGE) we went for a client/server architecture with an MS-WINDOWS user interface. After some false starts and networking problems, we now have a software product called FLEXIBASE/SQL that has a common client user interface for both HP3000 and HP9000 users so you can concurrently manage multiple databases on different hosts from multiple windows on the same PC. You can't tell which hardware platform is being used on the host, nor which type of SQL DBMS is being accessed. This is a step forward, even for old-fashioned software guys like me. Incidentally you can use FLEXIBASE/SQL on IMAGE databases via the IMAGE/SQL software of course.

What H/P Has To Do To Make It All Happen

I hope that my vision of the future is a lot clearer to you. It is based on distributed systems and client/server. I don't believe that client/server is just a new way for computer vendors to sell you more hardware and software, and I have a commitment to open systems standards where possible.

However you will soon learn when you take this path that this solution is not easy. Installing and supporting client/server systems is complex and requires new expertise. It doesn't help that you can't buy a single vendor solution. H/P won't even support your network software nor your PC clients in the traditional way, so you're on your own. Even though you can buy most of the components you need from H/P (they do sell PCs, networks, host systems and software), actually trying to do so is exceedingly difficult. I gather even the CEO recognizes they have a problem in this area which is probably an understatement in my experience.

H/P needs to resolve both these issues if they really want to satisfy the commercial mid-range users, and even more so if they want to capture a proportion of the mainframe market as they are aiming to do.

Unfortunately H/P management seems to think that just because the computer system products it sells are becoming cheaper, it should abandon

customer contact and leave it to third parties with lower quality customer service as a result. Just because I am buying a lower price product does not mean I expect to get poorer service. If you walk into MacDonalds the product is low cost, and not necessarily to my taste, but the service is wonderful.

If H/P can get these issues fixed, with their high quality products being sold and supported properly, they will be a good choice for the future.

PAPER #: 2010
"TURNING MULTIMEDIA DATA INTO INFORMATION"

Mike Kennedy
Excalibur Technologies Corporation
9255 Towne Centre Drive
San Diego, California 92121
619-625-7900

The challenge today and in the future is to locate the data in local and wide area networks and then determine the informational value that you are looking for.

With the proper information retrieval tools, there exists the opportunity to develop a new class of application by providing data elements such as voice, text, images and full motion video natively. This has the potential of adding incredible value to the application.

The concept is fairly simple. People have been collecting data for years and storing it in flat files and RDBM's with little or no analysis being done and with a limited set of tools to access it.

People are badly in need of tools that can excavate the data and a new generation of software and hardware is needed that provides users with the capabilities to find and correlate data.

There are emerging technologies and tools in the area of parallel processors, text retrieval systems, pattern matching technology, fractal compression and natural languages that address these needs.

Information retrieval today stands on the threshold of becoming a strategic tool for information management. With the prolific digitization of information, the development of technical standards which cut across databases, networks and platforms, and the merging of computers and telecommunications to create an environment of virtually unlimited information distribution, there is growing need for

intelligent software that organizes, adapts, searches and retrieves information on demand.

I would like to begin with a few observations made by Richard Saul Wurman in his best-selling book, *Information Anxiety*.

“More new information has been produced in the last 30 years than in the previous 5,000 . . . Today, a weekday edition of *The New York Times* contains more information than the average person was likely to come across in a lifetime in seventeenth-century England. . . . The English language now contains roughly 500,000 usable words, five times more than during the time of Shakespeare . . . The amount of available information now doubles every five years; soon it will be doubling every four.”

Today we mass-produce information the way we once mass-produced automobiles, and information has become an international currency upon which fortunes will rise and fall. This paper will explore some of the key factors shaping the business of information, particularly as they relate to information retrieval.

First, with respect to fortunes rising and falling, allow me to make two points:

- all gold may glitter, but
- all that glitters is not necessarily gold.

Today, the glitter of information is digital. Virtually every means we have of comprehending information — words, numbers, pictures, and sounds — can now be transmitted, stored and processed as digital data. The word “multimedia,” though much in vogue these days, may prove to have a remarkably short half-life. In point of fact, any and all media are, or can be, digital. Today, there is really only one primary information medium, and it is digital. To quote Nicholas Negroponte, the head of MIT’s Media Laboratory, from a recent *Byte* magazine article: “Bits are bits.”

But we must remember that everything that glitters isn’t necessarily gold. It is curious that the word “information” has come to define our modern age but has itself become

ambiguous and overused almost to the point of senselessness. Once commonly defined as instruction, teaching, and meaningful communication relating to the formation of ideas and knowledge, the word "information" has been increasingly applied as a technological term referring to anything that can be sent over wires, beamed through fiber, bounced off a satellite, or stored in a computer's memory.

This technological meaning has become so pervasive that the critical distinctions between data, information, and knowledge have been largely — and mistakenly — forgotten. The heart of the matter is this: raw data can be, but is not necessarily, information. Data are facts, information is the meaning that human beings assign to these facts. Further, meaning, or useful knowledge, is the inherent value of information. As one observer of the information age stated: "Information is not knowledge. You can mass-produce raw data and incredible quantities of facts and figures. You cannot mass-produce knowledge."

But we may be on the verge of a new era in information retrieval technology that will, in fact, allow us to extract knowledge from data using computers in new ways.

Information retrieval, like every other aspect of the information industry, has been and will continue to be shaped by the twin forces of technological innovation and market demands. Computers entered our workplaces some forty years ago as machines with tremendous power to perform complex numerical computation and analysis at great speed. Today, they process and store not only our numbers, but our words, pictures, sounds, and signals as well. The evolution of information technology — powerful workstations on the desktop, networks, intelligent software — has brought us to the threshold of seemingly limitless data and computing power. For decades we have been asking for more, faster. Now that we have it, the real challenges of information management are about to begin.

Broadly speaking, within the information industry there are four kinds of businesses.

These businesses require a non-restrictive, unstructured means to access, distribute, manage, store and retrieve information. First, and most obvious, is the creation business. Today, we can include in this category any organization — large or small — that has anything to do with the media, education, manufacturing, consulting, financial services, health care, legal services, insurance, law enforcement, government, research . . . the list goes on and on. Suffice it to say that virtually all of us, to some degree, are in the business of creating information. In large part, the information creation business represents the marketplace for the other three.

The second information business is easily recognized. It is the transmission business. Powered by brilliant technology, insatiable demand, and handsome profits, this business is virtually the exclusive domain of Fortune 500 companies: the ones who crowd the headlines with news of multi-billion dollar mergers; the general contractors who are competing to build the “information superhighway.”

The third business of information, still speaking broadly, is the storage business, and this domain includes companies of every size. This is the province of computer hardware and software, and it is the area where some of the most innovative technological breakthroughs have taken place. Changes of scale in power and performance have become routine in this business. Devices that once filled entire rooms now sit on our desktops, powered by microprocessors the size of a fingertip. Yet while physically shrinking, storage technology has at the same time increased in capacity in inverse proportion to its size. Information that once filled four file cabinets now fits on a disk no bigger than our hand.

There is a fourth domain in the information industry, and it is largely unexplored. This is the new frontier of seemingly infinite information and limitless computing power. It offers untapped potential for new kinds of businesses that enable “understanding” by helping customers bridge the gap between data and knowledge, making information meaningful, applicable, and approachable. These are the businesses that will build the off-ramps of the information superhighway.

The information technology of the 90s offers us the promise to access, distribute, manage, store, and retrieve information in real-time, regardless of database, network or platform. The information challenge of the 90s, driven by market demand, will be to liberate knowledge from data, empowering human skills and enabling productive action.

Traditionally, the information retrieval business has been viewed as a supporting player in the data storage business, and most retrieval companies and technologies have been content to focus on objectives defined by this limited view.

Legacy retrieval technologies, which search strings of text, depend almost exclusively on methods of organizing information by manually analyzing and classifying its content. Most early text retrieval systems, for example, only allowed searching for documents based on a few key words or phrases selected in advance by a librarian when the document was archived.

Legacy retrieval systems for structured database information have also failed to extract all of the value that can be gained from historical data. One industry observer has pointed out that "companies have spent billions of dollars over the past 10 years to create relational databases in the hopes of "extracting a broad range of information from data . . . (however) only 10% are really delivering benefits of the original investment . . . People want to know trends and patterns, you can't get that from the current implementation of SQL."

When we consider applications for multimedia information retrieval, the need for innovative technology and new methods becomes abundantly clear. Two basic challenges confront conventional retrieval technologies when we try to apply them to multimedia information:

First, multimedia information can present daunting volumes of data. For example, a typical 90-minute movie, at only marginal resolution and color quality as compared to broadcast video, may require close to 60 billion bytes of storage. Even the most powerful compression would leave us with megabytes of data.

How can we expect to index, search, and retrieve on the full-contents of such information? Unless we are willing to invest in impractical levels of brute clerical force for hand-labeling every separate frame, the best that conventional SQL-based approaches can offer us is to treat the entire movie as a database BLOB. But that still leaves the problem of access to content unresolved.

The second major challenge of multimedia content-based retrieval is how to label and categorize difficult to describe data types. How do you label a sound? How do you describe a color, a shape, or a texture? Conventional models, which layer database structures or other predetermined organizational maps on top of data in order to retrieve its content, simply cannot handle the volume, or the complexity of multimedia information.

In contrast, Adaptive Pattern Recognition Technology offers capabilities for efficiently, flexibly, and accurately managing many types of unstructured, multimedia data and for retrieving it based on its contents. This innovative technology has proven itself in text retrieval and document imaging applications and shows enormous potential for meeting the multimedia challenges of the future.

Adaptive Pattern Recognition Processing (APRP) is based on neural network models that recognize and index binary patterns in all types of digital data. These models consist of a set of algorithms or pattern processors which self-organize and self-adjust to the data being processed. Unlike traditional models which must be manually "trained" by programming, APRP systems automatically "learn" by processing data.

Because Pattern Recognition indexes are data-directed and self-organizing, they readily adapt to dynamic, unstructured information. Further, APRP enables automatic indexing of data, eliminating the cost, the inherent biases, and the inefficiencies of subjective, manual indexing methods.

Pattern recognition indexes also enable full content-based retrieval of digital information. APRP indexes contain the pattern ID number and a linked list of the locations in the

original data where those patterns are found. Users simply present the system with a sample, or a clue, of the information they want to retrieve, and the search engine matches patterns in the clue with those in its index. In effect, the system matches the content of the sample with content in the data. Also the match does not have to be exact because of the fuzzy search which results from pattern recognition. In a text retrieval application within a document management system, fuzzy search allows users to find documents that contain the always present OCR errors and also does not prevent successful search results even with keystroke errors.

Pattern Recognition indexes are memory efficient, typically requiring only about one-third the space of keyword indexes, for example. And APRP algorithms are executable in parallel, allowing for dramatically reduced processing time using multiple processors.

APRP's integrated architecture enables content-based retrieval for any digital data, regardless of its type. As far as APRP is concerned, "bits are bits" and "patterns are patterns." The technology has proven successful in text retrieval and document imaging application products in a wide range of markets.

As an example, Pattern Recognition can learn a series of handwritten Kanji images and then when presented a Kanji character clue written by someone other than ones who created the original database, an APRP based application can find the closest match.

Just as the bit is the basic unit of binary information, and the database field the basic unit of structured alpha-numeric information, the electronic document will become the basic unit of unstructured multimedia information; and its dynamic character will shape the document management systems through which it flows.

- Legacy systems of the 70s and 80s created "continents of information"
- In contrast, today's IS infrastructure is a heterogeneous, highly distributed, user-centric environment.

- Organizations are rich in information only if they can bring together the diversity of platforms, data repositories, and applications under one roof.

- The growing popularity of imaging has come in large part for its obvious benefit of reducing paper.

- Imaging has played a greater role in opening the door to a common electronic metaphor by which to join otherwise separate and distinct elements of the organization — the electronic document. This shift to user-centric, document-based information is the foundation of technologies such as workflow and groupware.

New consumer markets are emerging as well. Perhaps no more compelling metaphor has emerged in the information industry in recent memory than that of the “information superhighway.” For the past year, or more, it has been difficult to pick up a newspaper or a magazine, or turn on the television, without encountering another story of another mega-billion dollar merger involving some of the largest companies in the world, vying to build the information superhighway. And then the subsequent derailment of the deal because of Judge Greene.

America loves “big ideas” and the information superhighway is one of the biggest to come along in some time. Most people aren’t sure exactly what it’s going to be. Ask ten different people and you’re likely to get ten different answers — particularly if they happen to be in the information business.

But in the area of on-line information services, one thing that millions of people are agreeing on is the Internet, which one observer has described as “. . . a remarkable worldwide computer cooperative, a government-subsidized experiment in distributed computing, electronic community, and controlled chaos.” Total subscribers on the Internet have exceeded 15 million, and are growing by as many as 150,000 new users per month. Add to this the commercial on-line services — CompuServe, America Online, etc. — and you can see just how big the potential consumer market is.

In the area of broadcast and cable television, the promise of a 500-channel universe is causing a radical rethinking of how this industry is going to do business in the future. Freed from the control of supply-side programming and scheduling,

consumer television is heading into an age of microcasting, "open sea" channel-surfing, and on-demand programming. The market for helping users navigate those 500 channels is already starting to attract the attention of a number of companies, ranging from start-ups to some of the largest enterprises in the world.

But, whether you connect to the data highway by copper, coaxial cable, fiber or radio, the key unanswered questions are how you will interact with the giant network — whatever it is and whatever it's called — and what you will find there. Being linked to everybody and everything in the world won't do much good if you can't use the system or locate services you need — or if there's no data on-line you care about.

Another area of immediate concern is the area of document and information management. An early promise of computing was rapid access to information, but that benefit is not necessarily delivered by traditional methods of manual searches or subjective index references upon which many "advanced" offices still rely.

Quick access to correct information enables responsiveness and competitiveness. Some organizations have implemented and expanded the use of databases and various document management software packages to meet the demand for information access. However, these approaches can't always respond to the specificity and depth required by advanced information requests. Full-featured imaging systems can provide the missing elements to immediate information access, and also complement installed database solutions.

Imaging products provide the strategic software solution that completes the suite of tools required for competitive information management.

Product requirements for choosing an imaging system include the need for:

- Versatile document handling: Organization, storage and preparation.
- Fast and easy document retrieval
- Multiple platform support: Availability on a variety of platforms, environments and compatible with multiple

databases and other integral systems software. Long-term commitment to open systems and client/server architecture.

-Ease of use: Intuitive operating environment and GUI interfaces that shorten training time. Natural query processes, and a variety of file formats and data capture methods.

-Integrated imaging: Integrates scanning, OCR, display and printing, all working together seamlessly.

Once we've gone to the expense to load an image database, simple prudence says we should be able to find the information we need, whether we expected to need it or not. We believe that the future of imaging will be based on pattern-based retrieval of the actual digital information in the images. Pattern-based retrieval offers significantly greater flexibility than conventional methods, and we avoid re-cataloguing costs over the life of the information.

APRP permits the retrieval of digital information based on patterns in the digital information. With appropriate human interface software, a user could, for example, circle a customer name or a part number on an image of an invoice, and retrieve all of the invoices for that customer or part. If the organization later discovers a business need to locate all of the shipments to a geographical area, the system is able to respond without re-programming.

The potential applications for pattern-based retrieval technology are enormous. Digital information is expensive to store today, but not too many years ago we were concerned about storing abstracts of text documents instead of their full text for the same reasons. Ten years from now, most of our newly created information will be digital.

Imaging systems are already available with an underlying architecture that supports digital information retrieval even though the user still works with what appears to be a more conventional system. As we need to migrate to more flexible applications over time, the capabilities are already there to do so.

Information technology has already transformed our society, our economy, and our culture to such a degree that it's easy to forget that we are only at the beginning of the

Information Age — and we can only speculate on the changes it will make in our lives over the long term.

The near term, however, is clear.

- As standards become widely adopted, technology borders will begin to disappear.

- As computer power and storage become even more inexpensive and widespread, and as computers and telecommunications combine to create unlimited information distribution, universal access to information will become a practical reality.

- Many industry analysts believe that in the next few years, electronic documents will become containers for many types of multimedia data, including text, still images, full motion video, voices and other sounds. While the technology is well in hand for converting these data types to digital form, application tools for managing, categorizing, filing and retrieving multimedia information are only in the earliest stages of development but absolutely required for users to retrieve the information that they need.

- In the final analysis, a world of electronic multimedia documents dispersed across global networks will create a virtually unlimited market for intelligent software that organizes, adapts, searches and retrieves the right information on demand. Companies that succeed in this market will be “understanding” businesses that add value by bridging the gap between data and knowledge, making information meaningful and useful.

Paper Number: 2011

Leveraging Support Alliances

**Kumar Gollabinnie
Bashinder Grewal**

Hewlett-Packard Company
100 Mayfield Avenue
Mountain View, CA 94043
(415)691-5824/5071

Introduction:

The advent of open systems and the decline of proprietary solutions in the computer industry has led to a new phenomenon of strategic alliance formation among vendors. Such strategic alliances among vendors provides unique opportunities for their customers. Customers are able to significantly leverage the combined core competencies of the allied vendors. This is especially true in the computer services and the support industry when vendors ally to provide support solutions in open systems environments. By understanding the core competencies that result out of support alliances customers can make well informed decisions in selecting their support providers.

This paper will discuss the benefits that accrue to the customer as a result of vendor support alliances. At the end of the discussion a practical tool has been provided that can be used during the selection of support vendors in a multivendor environment. It is important to first understand the support challenges that customers have to face in multivendor environments before looking at the motivation behind the formation of such support alliances among vendors. Finally, a brief overview of some of the different types of support alliances is also provided.

Support challenges in multivendor environments:

1. **Expensive Support Infrastructure.** The infrastructure required to support multivendor environments is very expensive especially if customers opt to support such environments using in-house capabilities. The investment to support an open systems environment is higher than that required to support a proprietary solutions environment. Due to the diverse nature of multivendor products, self maintainers now have to invest in training, parts, test equipment, and diagnostic tools to support their products. It is a well known

fact that even under normal circumstances the cost of self maintenance far exceeds the cost of purchasing support from the OEM.

2. Up-to-date technical expertise. The constant challenge to keep up to date with the technical expertise required to support these products in an ongoing manner is also a very important factor. Vendors introduce new products incorporating newer technologies more frequently. Therefore, keeping up with the support expertise is very essential. Needless to say that training and maintaining such highly skilled personnel is a challenge in itself.
3. Management of multiple support vendors. One of the challenges for customers who contract support services from a multitude of OEMs is to manage the multiple support providers. The terms and conditions of support are diverse. Customers have to deal with various ways in which support is delivered on products. The administrative procedures are different and varied. For instance, the maintenance fees may be on a monthly, yearly, or a per visit basis. For networked environments managing the fault isolation and problem resolution involving multiple vendors could become a chore in itself.
4. Global support delivery capabilities. Customer organizations may be globally or nationally dispersed. OEMs may not be able to provide consistent support across the geography of such customer organizations. This issue becomes important when dealing with mission critical environments. For instance, the support provider may not have some or all of the support elements such as parts, support personnel, technical expertise, and response center capabilities available across the geography of the customer's organization.

Why support alliances?

The ideal solution in overcoming the above mentioned challenges in supporting multivendor environments is to outsource support to a single vendor. However, in actual practice, we see a trend by major corporations to choose a few vendors and manage them effectively in supporting their multivendor environments. Thus the few chosen vendors will now have to grapple with the support challenges in such environments.

Support providers address opportunities in the multivendor support environment by forming cooperative support alliances with OEMs and leading independent support providers. Such alliances are strategic to support providers since they leverage the core competencies of each other in providing support solutions to their customers. It is important for the customer to know the details of these alliances in order to understand the actual delivery of support by these vendors. Thus the benefits that vendors achieve by forming such alliances directly impact the quality and nature of support delivery.



Alliance motivators & support delivery benefits:

Looking at the reasons that motivate support providers to develop support alliances will help customers to understand how these motivations translate to support delivery benefits. Shown below are some of the key motivators and support delivery benefits:

<u>Alliance Motivation</u>	<u>Support delivery benefits</u>
Leverage field support personnel	* Availability of support at all customer locations * Reduced time for on-site support
Reduce investment on support infrastructure	* Lower support costs
Access OEM factory assistance	* OEM quality support
Avail OEM spare parts	* Reduced down time
Access factory training	* OEM quality support
Utilize warranty authorization	* Single vendor support during warranty and post-warranty periods
Meet mission critical demands of customers	* Higher standard of support
Joint marketing opportunities	* One stop pre-sales & post-sales support

Types of Support Alliances:

Customers can be empowered to make intelligent assessments of the value added by support providers by understanding the nature of these partnerships. Support Alliances may be classified based on the levels of commitment and involvement between support partners. The level of commitment and involvement is once again dependent upon the extent to which partners can complement each others core competencies.

The following discussion provides a brief overview of the various types of commonly established support relationships.

Authorized support provider relationship: The OEM authorizes another vendor to perform support both during warranty and post-warranty periods. This relationship provides maximum value to customers since the vendor obtains the critical support elements directly from the OEM. In other words, such a relationship provides customers OEM level or better support via their support

vendor. This is because the vendor leverages the following core competencies of the OEM:

- * technical expertise
- * factory technical assistance
- * Spare parts
- * Documentation
- * Software patches & Updates
- * Warranty support
- * Diagnostic tools

Subcontracting relationship: The primary support provider subcontracts a portion of the customer's support requirements to an OEM, an independent third party or an authorized support provider of the OEM. The customer is relieved of dealing directly with the subcontractor since the primary support provider may manage the following support elements:

- * terms and conditions of support
- * quality of delivery
- * fault isolation and problem resolution
- * on-site repair
- * billing and invoicing

Agency relationship: The support vendor uses the concept of an agent to manage support on behalf of the customer. In such a situation the customer is required to purchase a support contract with the OEM. Customers legally appoint the support vendor as their agent. In essence the support vendor legally manages the customer's support contract with the OEM. Customers empower the agent to insure proper delivery of support.

Affiliate relationship: Affiliate relationships are unique to networked environments. For instance, affiliate agreements allow the primary support provider to work with other vendors to isolate and resolve customer's network problems. The primary support provider leverages the resources of other vendors to effectively isolate and resolve network problems.

Primarily, affiliate relationships provide proactive processes and structure for interaction between the primary support provider and other vendors while delivering network support. Customers benefit in dealing with such support providers because the support provider manages not only the fault isolation and problem resolution but also provides a functional perspective of the customer's network.

Resulting customer benefits:

1. **One stop shopping center.** By choosing a primary support provider that has strong support relationships with other vendors customers benefit from the one stop shopping concept.
2. **Reduced support investment.** The in-house investment in establishing and maintaining a support infrastructure is minimized or eliminated if support is outsourced.
3. **High quality support.** When a leading high quality primary support provider teams up with other OEM vendors the service rendered by the primary support provider on OEM products is also of high quality. This is especially true in situations where leading system vendors may team with internetworking or desktop device vendors. For instance, Hewlett-Packard Company provides HP quality support for SynOptics hubs and Cisco routers.
4. **Consistent support across customer organizations.** Normally, large system vendors provide consistent support for multivendor products globally or nationally across customer organizations. For instance, IBM delivers consistent support for Wellfleet products in the US.
5. **Reduced support costs.** The customer realizes the benefits that accrue as a result of economies of scale achieved by utilizing the infrastructure of large support vendors over that of smaller vendors. These economies are translated as competitive support prices without sacrificing the technical backup assistance from the OEM.
6. **Custom support solutions.** The support provider leverages core competencies among partners to provide custom solutions that meet special support requirements of individual customer organizations. For instance, a customer organization may require a two (2) hour response time at a location where the primary support provider does not have an office. The primary support provider may partner with a local support vendor to deliver such a response, thus leveraging the competency of the local partner company.

A tool to evaluate primary support vendors:

The following discussion will help in evaluating support vendors that will meet customer-specific support requirements. Knowledge of different types of relationships may not be sufficient in evaluating vendors, especially because each vendor may define the support relationship differently. Customer support requirements may be unique. Ultimately, it is the customers support requirements that need to be satisfied. It is useful for customers to understand how their

support requirements map to the capabilities that the vendor has to offer. Thus an attempt has been made to provide a process tool that aids in evaluating customers' support requirements with vendor capabilities that are directly a result of the vendor support alliances.

1. List products. The first logical step is to identify and list the products that make up the customer environment. For example, desktop PCs, peripherals, networking devices, and communication media etc.
2. Identify mission critical components. Functionality of certain key components may be critical for normal operation of a business environment. Therefore, it is useful to identify these components. For instance, a server in an office LAN environment is critical for the normal operation of the entire work group. Whereas a router for a stock trading company is critical since most of the trading is accomplished electronically. When the router is down the trading company business comes to a grinding halt.
3. Identify support requirements. Each component in the business environment may require different levels of support. Therefore, identify the minimum support requirements for the various components. For instance, not all components require a two hour telephone response time or on-site repair.
4. Map support requirements with vendor capabilities. List the support delivery capabilities offered by various vendors and map them to individual support requirements of the components.
5. Shortlist vendors. Select a few top vendors that meet the majority of the support requirements.
6. Compare prices. Perform the price to service tradeoff analysis.
7. Consider other competitive considerations. It may also be useful to consider other competitive considerations such as:
 - a) vendor reputation
 - b) anticipated future support requirements
 - c) global infrastructure
 - d) competency for the business environment under consideration
8. Select the vendor of choice. Compare, rank, and choose the most suited support vendor for the business environment.

Businesses must select the support solution that helps their business grow in the face of rapid change. There is no right or wrong answer - nevertheless, the trend

seems to be outsourcing support to a few leading multivendor support providers who can leverage OEM core competencies by a plethora of support relationships.



PAPER # 2012

RIGHTSIZING TO CLIENT/SERVER APPLICATIONS

JOHN M. WOOLSONCROFT
CONCEPTS DYNAMIC, INC.
1821 WALDEN OFFICE SQUARE, SUITE 500
SCHAUMBURG, ILLINOIS 60173
708-397-4400

INTRODUCTION

The business environment of today -- fast-paced, competitive, customer-driven -- has forced companies to find ways to improve their decision making. In a marketplace that changes rapidly, companies that make quick, well-informed decisions become market leaders. On the other hand, companies unable to use corporate information to solve problems and meet market demands are finding it increasingly difficult to compete. Such companies will, over time, fall by the wayside in terms of market share.

For most corporations, the need to "open up" corporate information to analysis and decision making is running into a barrier created by their existing information systems. These existing "legacy" systems, built largely in the 1970s and early 1980s, were designed when corporate information was viewed as "data" (hence the term: "data processing"). In such systems, access to data is available to few people, usually on a batch-report basis. These systems, by their very design, do not offer the flexibility and access to information that today's rapidly-changing organizations demand.

To meet today's needs, a new definition, or model, for information systems must be introduced. That new model is client/server computing.

INFORMATION AS A COMPETITIVE TOOL

Business conditions today mandate that a company's information assets be used as management tools to help take advantage of market opportunities. Companies who want to remain competitive are seeking ways to implement flexible and easily managed computing environments that put accurate, up-to-date information into the hands of those who need it.

Forrester Research, Inc., a well-respected information technology research firm, has identified the growing trend in major corporations to include information systems as a competitive tool:

"The pressures driving FORTUNE 1,000 firms to look beyond the mainframe are not about to subside... the mainframe's cost and inflexibility will be highlighted as companies are forced to find new ways to compete. Senior management, MIS, and end-users alike will be looking for alternatives. This will fuel the ongoing re-deployment of corporate applications to other platforms."

Wal-Mart and Toys-R-Us are two excellent examples of companies who decided, early on, to base their businesses on distributed, open systems that use information as a competitive tool. These two companies now

dominate their markets and are causing their smaller, less responsive competitors to scale back or leave the business altogether. The competitive edge they have established is a direct result of their use of corporate accounting information to speed decision making about inventory levels, product pricing, etc. As a result, they have not only improved their service to customers, they have also dramatically reduced their operating costs below their competitors'.

Arthur D. Little's *Forecast on Information Technology* sums up the current views on information as a competitive tool:

"...though the integrated information system is not the only key to success for the enterprise of the mid-1990s, the lack of such a system will be one of the surest ways to guarantee failure."

DEALING WITH LEGACY SYSTEMS

Unfortunately, most organizations who would like to use information for competitive advantage are finding out that their closed, centralized computer systems make this nearly impossible. These systems, designed 10-20 years ago, cannot provide the access to information that today's business environment demands. As a result, people throughout the organization become disenchanted and dissatisfied with their company's information systems capabilities. This includes:

- The CEO who wants the flexibility to reorganize quickly and easily as market demands change.
- The CFO who wants to reduce the costs for computer hardware, software, maintenance and personnel and to optimize the

company's long-term system investments.

- The CIO who wishes to increase productivity and have more choice of new, cost-effective software and hardware solutions.
- The programmers who want to spend their time creating new, valuable information systems rather than being tied up in program maintenance.
- And, most importantly, the users at all levels of the organization who want better and faster access to information.

As consultant Frank Gens stated in a recent interview with *Computerworld*:

"Corporate management won't be sympathetic to IS [not doing] something strategic because they were busy taking care of their legacy systems... The IS organization's role is to facilitate, not dictate, business change."

Strategic Maneuvers Are Inhibited

The rigid structure of legacy systems can limit a company's ability to adapt to changing business conditions. Often a company finds that system enhancements and modifications, needed to reflect new products and organizational structures, are expensive, slow and complicated to implement. Because mainframe applications are usually designed from the "top down," any change, no matter how minor, has an impact on the entire system.

Take, for example, the effect that an organization's restructuring has on its corporate accounting systems. If the hierarchy of an organization changes, even slightly, the entire system can be

affected in ways that are often hard to predict. Expenses may be tied to the wrong cost centers, making it impossible to accurately assess business performance and profitability. This can lead to incorrect decision-making that adversely affects the company's performance. At its worst, this inflexibility prevents restructuring, even though it is needed to better respond to the needs of the business. In these situations, the company's information systems restrict corporate growth instead of supporting it.

Decision Making is Impeded

Making informed decisions is always critical to any company. But in the ultra-competitive business environment of the 1990s it has become the key to corporate survival. Companies that cannot track and quickly report information on revenues, expenses, cashflow, asset values, and other key areas are in particularly vulnerable positions. These financial numbers are to a corporation like vital signs are to an individual. If they are good, the person or the company will survive and thrive. If they are bad, immediate action must be taken to restore the person or company to health. A company that tries to run without rapid access to the proper information is in the same position as a person whose doctor is looking at last week's blood pressure. The information may still be accurate enough, but is the risk worth taking?

In the closed world of most legacy systems, information is not shared between systems and users have little real-time access. For example, the manufacturing system may not link cost information to the company's general ledger system. Data must be re-keyed many times from system to system because of this lack of integration -- a waste of time and resources. Decision

making is adversely affected because the people who need the information cannot get to it in a timely manner. And when they finally do, it may contain errors because of the frequent rekeying. All of this leads to slow and inaccurate decisions - a deadly combination in today's world.

Maintenance Quagmire Delays Progress

Most legacy mainframe system users are caught in a terrible maintenance quagmire. In a traditional mainframe information systems group, so much time is spent on maintenance that little time is left for development of new, mission-critical systems. Studies have shown that as much as 80% of a typical information systems staff's efforts go into maintaining older systems. With numbers like these, it is easy to see why programmer productivity in delivering new applications is so low in many Information Systems (IS) departments.

One of the major reasons for this maintenance quagmire is that most existing mainframe software systems are full of what has come to be called "spaghetti" code. They are characterized by complex, layered architectures that have built up over years of development and modification. These characteristics create an environment in which simple fixes and enhancements take four times as long as the development of an entirely new application. Furthermore, with mainframe software systems there are very few development productivity tools available and, in addition, most of these older systems have very poor, if any, system documentation.

Because of the development backlog and the slow response to requests for changes, users become frustrated with a proprietary mainframe environment.

As a result, user departments frequently create their own information solutions, further adding to the complexity of the overall computing system. As time goes by, the number and complexity of disparate systems grows and everyone sinks deeper into the quagmire.

Mainframe Costs Devour IS Budgets

In looking at the costs associated with a mainframe-based computer system, there are really two kinds to consider: direct and indirect. Direct costs include such items as salaries and benefits for support personnel, and acquisition and ongoing maintenance costs for hardware and software.

Studies have shown that, on average, a mainframe requires twice the staff, 500% higher maintenance costs, 250% higher license fees and 150% higher acquisition costs than a mainframe-alternative UNIX-based system like the HP 9000. In fact, the studies have shown that the costs involved in simply *maintaining* a mainframe system are higher than the total cost of acquiring a mainframe-alternative, UNIX-based system.

A good example of this is offered by Hyatt Hotels and Resorts, which shifted from a proprietary mainframe to an open UNIX computing environment. The initial hardware investment for Hyatt's UNIX system was significantly smaller than the costs required just to *support* their previous mainframe-based system. Hyatt says that they have reduced costs more than 25% with their new open systems-based solution.

In addition to high acquisition and operating costs, any upgrades to a proprietary mainframe system are usually quite expensive. Also, the customer is usually locked into a single vendor for obtaining these necessary upgrades. On top of all this, depending

on configuration, mainframe systems are not as easily scalable as typical UNIX-based systems. This makes both functionality and power difficult to add as an organization grows.

Indirect costs of a computer system include such items as lost profits resulting from poor decision making due to a lack of access to information, as well as increased personnel costs for user departments forced to deal with systems that are difficult to use. In many organizations these indirect costs can be much higher than the direct costs of acquiring and operating the systems. For this reason, they must be factored into any analysis that looks at the true overall cost of information systems.

As you can see, the financial burden associated with legacy systems that use mainframe technology is forcing companies to explore new models for computing. According to a study done by Forrester Research in 1992, users have achieved an *"...average \$1.7 million per year...in savings"* by pursuing mainframe-alternative computing solutions. These types of cost savings, which will be explored in a later section of this paper, are helping to drive the ROI numbers that clearly support the move to client/server systems.

OPEN SYSTEMS: PROVEN MAINFRAME ALTERNATIVES

Before going further, it may be helpful to take a moment to clarify what is meant by "open systems."

Open Systems Today, a leading computer industry publication offers this definition:

"Open systems integrate existing computer resources with the newest,

most powerful, and cost-effective technologies in order to deliver the information users need to do their jobs.

Open systems require the purchase of standards-based, vendor-neutral information technology products of which UNIX is the most important example.

Open systems enable interoperability with, and portability to, hardware and operating systems of competing vendors."

Open systems break down the barriers that block the free flow of information within a corporation. They make use of standards-based building blocks to create an information system that adapts to changing business needs. With open systems, companies can easily install and integrate the latest and most powerful hardware and software tools.

UNIX International, the organization of nearly 300 companies who have joined together to assure the future of open computing based on UNIX, recently summed up the prospects for open systems in the decade ahead in this way:

"The most valuable type of operating system for the computing environment of the 1990s is, of course, open -- by which we mean that no one company or organization controls it; that it will facilitate the building of interoperable solutions; that it will enable open network solutions; that it will incorporate the latest, most powerful technology; that it will run on every important computer platform; that it will provide ease of software portability; that it will adhere to industry standards, and that it will be made freely available to vendors, resellers, and users."

All of the factors mentioned earlier are leading companies to make the change from closed, inaccessible mainframe systems to more accessible "open systems" that are based on a client/server architecture.

In a June 4, 1990, *Computerworld* survey of 194 IS Chiefs, "48% of the respondents said they were considering mainframe alternatives." According to these IS executives, greater flexibility, reduced cost and greater user access were recognized as the primary benefits of moving applications off the mainframe. A 1992 Forrester survey showed additional evidence of the move to mainframe alternatives and stated that, based on the numbers, the move to open systems was rapidly accelerating. The Forrester study reported that

"...a staggering 80% of the 75 FORTUNE 1000 firms interviewed are looking to break the mainframe's grip on application processing."

Motorola's General Systems Sector is an example of an organization that has embarked on an ambitious open systems strategy. Corporate Vice President and Director of Information Technology, Bill Connor, has been quoted as saying, "My feeling is that downsizing is a part of making American business competitive." When asked which applications were staying on the mainframe in his sector of the company, Connor explained that the strategy is to "...totally get off mainframes... There's nothing that should stay on the mainframe." Quite a statement from one of the world's leading technology companies!

It is clear then that open, client/server systems provide a computing environment that supports the demanding business needs of the 1990s. Client/

server systems give people throughout an organization better access to critical information. At the same time, client/server systems help reduce the cost of computing -- another mandate for businesses today.

CLIENT/SERVER SYSTEMS AND CORPORATE ACCOUNTING

Client/server systems provide a new model for computing that is changing the information technology landscape in many organizations. But given the wide range of applications that companies use in their daily business, which are companies targeting as the first to migrate to the new client/server model? The accounting applications.

Accounting systems are at the very heart of every organization, so it is logical for companies to want to ensure that they are running efficiently and effectively. They provide critical, timely information that is used by everyone from senior management to departmental users to make decisions about the operations of the business. And because almost every other system in the company feeds into the accounting systems, they are the perfect choice as the first place to implement a client/server systems strategy. By choosing accounting software that utilizes the technologies of client/server systems, companies make it possible to share financial information seamlessly between applications.

Corporate accounting and financial departments at a number of companies are leading the move to client/server. According to *Computerworld's* 1993 forecast issue,

"financial departments will at least start to consider a full or partial move to open systems this year in an attempt to speed up day-to-day processes and make

financial data more accessible... Because accounting, payroll, personnel and applications of that ilk are more generic than, say, manufacturing systems, they're an easier target for ... a distributed computing model."

The growing trend toward client/server accounting is confirmed in studies done by several leading research organizations. For example, according to a 1992 *Datamation/Cowen & Company* survey, 7% of survey respondents were currently using UNIX-based accounting systems and an additional 6% were planning to move to UNIX-based accounting during 1993. Likewise, research done by International Data Corporation (IDC) shows that even at IBM mainframe sites, the move to UNIX-based accounting systems is gaining momentum. A recent IDC survey of 100 IBM mainframe customer companies showed that 7% say they have some portion of their accounting systems running on a UNIX platform, and an additional 21% expect to do so by 1994.

Often, the biggest impediment companies face in bringing the advantages of client/server to accounting is a sense of fear and doubt on the part of mainframe system users. These users have grown comfortable with the idea that only a mainframe-based accounting system can give them the full range of accounting functions, security and controls, not to mention transaction processing power, that their businesses need. And, up until recently, they were entirely correct.

Today, however, there are some new client/server accounting packages that demonstrate a thorough understanding of accounting issues and that are designed to take full advantage of the latest technologies. These new systems can easily match the capabilities offered

by older mainframe systems. They offer the functionality, security, auditability and controls of a mainframe, while providing equal or better performance, greater flexibility, and lower costs. Because of their central importance to future information systems plans, these new systems are quickly moving to the top of many companies' priority lists.

In addition to allowing users to access information from multiple systems, today's client/server based accounting packages help users in other ways. The best designed applications are easy to learn, use, and modify. These advanced accounting packages offer ease-of-use features such as Graphical User Interfaces (GUI's), pop-up help windows, zoom windows, operator prompts at each field, and arrow key or mouse "point and click" capabilities. As a result, users need very little training before they can work comfortably with a well-designed client/server based accounting application.

LEVERAGING THE BEST NEW TECHNOLOGIES

It is impossible to talk about the strengths of a client/server approach to corporate accounting without discussing the underlying technologies that make such systems so effective.

Technologies such as the UNIX operating system, relational database management systems (RDBMS's), fourth-generation languages (4GLs), wide area networks (WANs), and executive information systems (EIS's) are key parts of any highly-effective mainframe-alternative accounting system.

These technologies, when used in combination, provide an accounting information tool that can support the needs of any size organization, from a

small regional distributor to a FORTUNE-sized global corporation.

UNIX

"UNIX" and "open systems" are often used interchangeably to describe the new computing environment that we have discussed. This usage, however, is not entirely accurate. While most UNIX-based systems can truly be called "open systems," UNIX is not the only operating system that has some degree of "openness." UNIX is, however, the operating system that takes openness to its highest level.

According to Amo Penzias, Vice President of Research at AT&T Bell Laboratories, the original developers of the UNIX operating system,

"Strictly speaking, the term 'open systems' is tied to the notion that the supplier of the hardware and operating system need not be the same. The operating systems are specified by publicly-generated standards, not by manufacturers of hardware or by the operating systems themselves. UNIX ... is the only operating system that is specified by a standard."

Thus, the main reason that UNIX is considered the most open operating system is that it is the most standards-based operating system available. Because a standard is by definition consistent, software and hardware that are UNIX-based can change, so long as the new pieces of the system are kept compatible with the standard. Standards allow for change and improvement while protecting information technology investments.

UNIX-based software that is developed to industry standards also has the advantage of being portable and scalable. Users are free to move from

one hardware platform to another, so new and/or bigger machines can be implemented in the future. This means that the software system can easily change to meet a growing business' evolving needs. Being standards-based also means that applications are interoperable, or easily able to share information.

UNIX has established strong market presence as the operating system of choice in the area of open systems. This strategically positions UNIX for continued growth. UNIX is the only standard operating system that has been in use for twenty years and that will run on a full range of hardware platforms, including mainframes, mini-computers, servers, workstations and PCs.

Today, more than two hundred different computer vendors offer products based on the UNIX operating system. And there are over 25,000 commercial UNIX applications, ranging from small business management to large scale industrial applications, being used by companies in a wide variety of industries. Because of this widespread use, progress remains constant and enhancements to UNIX are continually released.

The future prospects for UNIX are quite strong, as confirmed in market research reports by several leading firms. For example, research from both IDC and InfoCorp shows that UNIX license sales will grow to \$32 billion in 1995 from \$18 billion in 1991. Dataquest Inc. predicts 1996 UNIX license sales will climb to \$44.7 billion and that the compounded annual growth rate (CAGR) of UNIX units shipped from 1990 to 1995 will be a phenomenal 27.5%.

Thus, UNIX-based software represents an excellent choice for any company seeking mainframe-alternative sol-

utions. This is true not only because of the open, standards-based technology, but also because the outlook for UNIX's continued success means that an increasing number of hardware and software solutions will be made available to users of UNIX-based systems.

Relational Database Management Systems

Up until the late 1980s, data management software had not kept up with advances in hardware. In recent years, new heights have been reached in terms of performance and data availability with the introduction of relational database management system (RDBMS) products, such as Informix's OnLine, that bring the power of mainframe computing to open systems platforms.

Open systems RDBMS's offer performance-enhancement features such as shared memory, a cost-based optimizer, and direct I/O. They also provide key distributed RDBMS features such as multi-threaded architecture, two-phase commit, declarative referential integrity, and stored procedures that allow information to be shared between multiple systems. Finally, to assure high availability of key information, these new open systems RDBMS's offer on-line archiving, disk mirroring, and fast recovery mechanisms.

Most importantly, relational database management systems offer users a fast, easy way to obtain information in a format that meets their needs. With an RDBMS, reports and queries can easily extract information from multiple database files simultaneously. This allows, for example, users of client/server accounting solutions to execute queries from any field in any

window on the client. As a result, they can easily review or examine all levels of detail about a given transaction without interrupting the flow of their work.

Advanced Programming Tools

No matter how good an application is when it is first implemented, business and organizational changes will force the modification of systems to meet new requirements. As we discussed earlier, in the traditional programming environment using third generation languages (3GLs) such as COBOL and BASIC, this ongoing program maintenance consumes large amounts of programmer time. In fact, up to 80% of a programming staff's time may get tied up in making these needed changes.

Fourth generation languages (4GLs) offer productivity improvement in the development of software code. These state-of-the-art tools require as little as one-tenth the lines of code needed by a 3GL to accomplish the same function. This means that fourth generation languages reduce initial application development time and significantly speed the process of modifying software. The result is that programmer productivity is improved tremendously with a 4GL, giving the organization more new applications and more application enhancements than it would otherwise get. This in turn results in significantly more satisfied users and managers throughout the organization, as well as more satisfied customers who come in contact with the company's information systems capabilities.

In addition to 4GLs, tools exist that can help software application developers by doing some of the routine programming steps for them. Computer-Assisted Software Engineering, or CASE, tools generate code automatically from a

specification entered by the programmer. These CASE tools can add a great deal of value by freeing programmers from the mundane tasks involved in creating code.

You should keep in mind, however, that some code and report generators available in the marketplace promise more than they can actually deliver. These tools are helpful for creating very simple programs, but they do not produce or allow for sophisticated, highly-adaptable programs. The "pseudo-code" they use is limited, and the reports they produce may not suit the special needs of the company using them. Most programmers prefer to use a 4GL with its inherent flexibility, rather than dealing with the limitations and short-comings associated with many of the available code generation tools.

Sophisticated Networks

The movement to open systems hinges, in large part, on the ability to communicate critical information between disparate systems. Today's wide area networks (WANs) and local area networks (LANs) help achieve this interoperability of systems by allowing different sizes and types of computer systems to work together.

In a client/server environment, WANs and LANs do not require any particular vendor's computer system or proprietary standards in order to function. The UNIX operating system on the server offers built-in networking capability and supports widely acknowledged standards, such as TCP/IP, OSI, NFS and DCE, for linking computers and creating distributed systems.

Today's ease of networking also allows for the scalability of hardware platforms to meet specific processing needs. For example, applications requiring high transaction volumes and/or numbers of

users can reside on large servers, while applications which have fewer users and/or transactions can be placed on smaller clients.

It is important to note, however, that interoperability goes beyond connecting, or networking, internal company systems. Interoperability often means support for links with systems outside the organization, through such technologies as electronic data interchange (EDI). Complex computing architectures, operating as a cohesive system both inside and outside the organization, are most capable of supporting a variety of business needs.

Executive Information Systems

We've already discussed how client/server systems help increase and speed access to information within an organization. But unless the users of that information are given the right tools for interpretation and analysis, not much is really gained.

According to the 1993 Forecast issue of *Computerworld*,

"There are signs that top corporate chiefs in marketing, sales, administration and other nontechnical areas are showing more interest in using technology to improve their own productivity..."

Fortunately, the latest software offerings in this arena are easy to use and provide the detailed analysis capabilities that corporate users need to make informed business decisions.

An important concept that has grown in use over the past several years is that of an executive information system (EIS). An EIS accesses data and presents it through spreadsheets, graphs, and charts for analysis. Users

of a well-designed EIS do not need technical expertise or knowledge; they can easily and quickly produce visual representations and interpretations of accounting data to aid in their decision making. An executive information system is especially useful for managers involved in evaluating business performance and strategic alternatives based on company financial data.

The value of executive information systems to today's top decision makers was confirmed in a recent *Computerworld* and Andersen Consulting survey of top executives in 200 large U.S. corporations. The study showed that executive use of desktop technology is rising significantly. According to the study,

"Many (top executives) expressed interest in using knowledge-based and executive information systems."

PROCEED WITH CAUTION

When a company decides to make the move to client/server, it should do more than just "slap a new coat of paint" on an old system. In other words, rather than simply putting a new graphical user interface on the system, or replacing one centralized system with a network of smaller systems, the company should focus on how to best support the current and future information needs of the business. Every aspect of the company's operations should be considered. Only then can the computing system be designed to support, rather than dictate, the company's operations.

Demand Full Accounting Functionality

In the specific case of accounting applications, it is usually best for a

company to purchase new accounting software that takes full advantage of advanced technologies, rather than spending the time and resources to build the applications internally. Accounting applications by their nature are fairly standard, allowing good packaged solutions to meet the needs of a wide range of companies in a variety of industries. The key is to seek out functionally-sound and feature-rich accounting software with proven success in the marketplace. This way, companies can start with a solid foundation and can go on to tailor the software to meet their specific needs.

Buyers of accounting software should be especially cautious of software vendors who try to sell a system that offers "modifiability" as the product's strongest feature. Such vendors will try to sell a package that has only a shell of accounting functionality with the notion that the buyer can then use the advanced programming tools we spoke of earlier to modify the system to their needs. While advanced programming tools, like 4GLs, are very useful in *refining* software packages, companies should not be forced to use these tools to "reinvent the wheel" by writing most of an accounting system themselves. After all, one of the main reasons to buy an accounting package is to avoid the tremendous time and effort involved in developing a system from scratch.

The best way for a company to insure that the package will meet the functional requirements of their business is to:

- 1) involve users in reviewing the functionality of the alternative packages, usually through a detailed product demonstration
- 2) check the references of the possible software providers to

determine whether they have proven results with their accounting software at companies whose needs are similar.

By choosing an accounting systems vendor that has proven knowledge of client/server technologies and corporate accounting processes, buyers of accounting packages will increase the odds of success and, ultimately, the effectiveness of their accounting information systems.

One final note on purchasing accounting applications code. Most companies, as we have discussed, require the option of customizing application software to meet their specific needs. For this reason, it is important that application software licenses include full access to source code, preferably at no extra charge. This will give an organization the flexibility it needs to address changing business requirements.

Establish Consensus on Key Terms

Another potential problem to watch out for when selecting a client/server applications vendor is a lack of consensus as to what is actually meant by various technical terms. For instance, "*client/server*" can mean different things to different people, especially vendor salespeople. A software or hardware vendor may present an "client/server" strategy that is actually nothing more than a nice graphical user interface (GUI) to a mainframe-based application. In such a scenario, the cost and proprietary nature of the mainframe remains, and the added expense and complexity of maintaining PCs with additional interface software is introduced. It is important to gain up-front agreement on key terminology so that there are no surprises later on.

Avoid Old, Tired Technologies

Trying to bring mainframe-based applications into the world of open systems has other problems to consider. Simply porting existing mainframe applications to a UNIX platform brings with it the burden of applications designed for the computing architecture of a decade ago. For that reason, buyers should avoid packages based on old architectures and tools. In the case, however, of a company with a large number of custom applications, porting may be an acceptable alternative to rewriting. The trick is not to bring all the software maintenance problems and inflexibility from the mainframe down to the new platform.

Don't Expect Too Much from PCs

At the other end of the spectrum from mainframe applications is the PC world. In some cases, client/server may be incorrectly defined as a *DOS-based LAN*, where processing and data are scattered across the desktops of the organization.

Unfortunately, local area network, or PC LANs, do not come close to providing the security and controls that companies rely on in a centralized mainframe environment. In the worst case scenario, high-volume, mission-critical applications may not even execute in the PC LAN environment because of the lack of processing power.

PC LANs do allow users to share data and functions in a limited way. Access is open, but usually only one user interacts with one piece of data at a time. For example, in the typical world of PC software, one user might create a letter or spreadsheet and then release it for another user's review and revision. Such simple user interaction and utility

sharing needs can easily be met by a PC LAN environment.

PC LANs, however, will not accommodate the transaction processing and data integrity needs of large corporations. In such organizations, especially in the financial area, the need for users to interact is much greater. Usually data is being entered to any given system by several concurrent users. Furthermore, data must frequently feed automatically, even instantaneously, into a related system. For example, billing entries may need to feed into the accounts receivable system. High-volume, on-line transaction processing can be better handled by a back-end UNIX server environment, where data integrity tools are already in place and where processing power can be much greater. The Windows-based PC client then acts as a user interface and application integration device to facilitate information access.

The bottom line to consider when deciding whether to move applications off the mainframe to a client/server environment is that the design of the system, including functionality, should match the operational needs of the business. Client/server systems offer a tremendous amount of flexibility to an organization, but they must be based on a technological and functional foundation that supports the company's current business needs.

REAL-WORLD SUCCESS STORIES

To better understand what is involved in moving to client/server systems, and what benefits such a move can have, it may be useful to look at the experiences of some companies who have actually been through it.

What follows are three case studies of actual companies in different industries who decided to implement a client/server strategy. While their businesses are quite different, they all had very similar information systems requirements in order to run their companies more effectively. All three found many of the same benefits to client/server systems, including substantial reductions in costs.

Billion Dollar Food Processing Company

Profile:

A \$1 billion division of a FORTUNE 200 company was time-sharing accounts receivable software from Computeristics on their corporation's IBM mainframe. Because their information processing was supplied by the parent company, the IS department for the division was small, employing only about 6 people.

A major problem faced by the division was timely access to their financial information. The division found batch input of transactions with overnight verification too slow -- errors were not dealt with until the next day. Responses to requests for information involved batch reports and took several days. Once the batch reports were prepared they had to be mailed or sent by overnight express to the division from corporate headquarters.

After considerable deliberation, the corporation decided that to manage the business more effectively, they had to bring their accounting operations into the 1990s. The plan that was developed involved moving off the mainframe entirely.

The first attempt they made at downsizing their accounting systems proved unsuccessful. It involved

rewriting the COBOL-based mainframe accounting applications they had been using. Unfortunately, they found that in doing so they were perpetuating an outdated design based on past models of their business.

System strategy and requirements:

After their initial failure, the corporation set a bold new information-systems strategy. To increase their systems flexibility and reduce overall costs, the decision was made to go with a UNIX operating system, HP 9000 hardware and to use Informix's Relational Database Management System (RDBMS) and 4GL.

Among the goals that were set were to:

- 1) Integrate billing feeds at four locations with accounts receivable,
- 2) maintain mainframe-quality audit controls,
- 3) obtain, for the first time, real-time user interaction with on-line validation, and
- 4) set up on-demand reporting.

Solution milestones:

The project plan allocated seven months to installing and implementing a new UNIX-based client/server accounting package. One month into the project the system was installed and testing of its functions began. As part of the testing process, users were heavily involved to make sure that the functions implemented would meet their needs, especially in terms of information access.

This user involvement proved to be an extremely important part of the project because as the users actually tried the system, and became aware of the possibilities for improvement, the "functionality floodgates" opened. Their wish list of customizations and

enhancements expanded significantly as they saw what was actually possible given the capabilities of the new, 4GL-based system.

Over the following 6 months, because of the programmer productivity improvements resulting from the use of the 4GL, these user-driven improvements were easily implemented. By the end of the seventh month the mainframe was unplugged as planned.

Benefits:

Obviously the most significant impact of moving to a UNIX-based client/server accounting system was the complete elimination of high mainframe-operations costs, as well as dedicated communication costs.

The change also meant that the division was able to take complete ownership and control of its own data, giving it better information access for decision making and greater accountability over its operations. In addition, because of the improved information flow, corporate A/R collection capabilities were streamlined and enhanced, resulting in better accuracy and reduced collection times a major cashflow improvement.

Finally, users were delighted to have easy-to-use, on-demand decision support capabilities tied to the more timely and accurate data. In fact, the system proved so popular with users and IS alike that after its first year in operation it was voted the best application in the corporation by both groups.

Division of Thirty-two Billion Dollar Service Company

Profile:

The U.S. service division of a \$32 billion, global corporation had 50 regional operating centers throughout the country and no internal IS staff. To run their accounting operations, the division was using outsourced MSA financials on a service bureau's IBM 4381 mainframe. Unfortunately, as a result of this arrangement the division found that it had no control over the rapidly escalating mainframe maintenance and upgrade costs that were being passed on to them by the service bureau. In addition, the division's users were extremely frustrated with very poor access to information and with the lack of service bureau responsiveness to requests for system modifications. On top of this, response to requests for new applications was also very slow.

System strategy and requirements:

Because of the cost-control and information-access problems associated with their outsourced systems, the company decided to terminate their outsourcing arrangement with the service bureau. The division management decided that all of their previously outsourced systems, including their accounting applications, were to be replaced with UNIX-based client/server systems that offered greater functionality and ease-of-use. Their criteria for selecting an accounting package specified mainframe-quality auditability, security and controls. In addition, greater access to information for site managers and significantly reduced operating ex-penses were critical requirements.

Solution milestones:

The first step in the project was to pilot new operational software for job cost accounting and to hire and train a new IS staff capable of supporting the new systems. Training for these people included both UNIX and 4GL skills and required a total elapsed time of three months. After this training was completed, the rollout of the accounting applications to remote locations began.

Over a period of one year, the division's new financial accounting software was installed and implemented, including migration of existing financial data from the service bureau mainframe system. The company now operates an HP 9000 server at headquarters, and the division's financial transactions from across the U.S. are posted through a wide area network that connects all 50 remote locations to each other, as well as to headquarters. The company is pleased to point out that the project of migrating their financial applications to a UNIX client/server environment was completed on time and within budget.

Benefits:

The new system is viewed as a complete success by company management, both functionally and financially. It met all of the users' expectations for increased functionality and enhanced information access, as well as their stated requirements for security, auditability and controls. The company's financial managers are pleased that the organization now has multi-location reporting capabilities with centralized cash management. A two-person IS staff now supports both the central computer and users at 50 remote locations. Thanks to the relational database and 4GL, these two people are able to maintain normal operations while still having time to

create application enhancements and customized reports.

From a cost standpoint, overhead for such items as software license and maintenance fees has been substantially reduced. In addition, day-to-day accounting operations have been greatly simplified and redundant tasks have been eliminated. For example, the 17 individually-created spreadsheets previously used for period-end reporting are no longer needed. And because of improved information access, the user community within the division feels that it is empowered with information, not just data. In fact, each operating location has increased control over their computing tasks and needs, resulting in much greater satisfaction with the company's IS resources.

Two-Hundred Fifty Million Dollar Technology Company

Profile:

Another example of a company that successfully made the move to client/server systems is a \$250 million high-technology company with 12 remote operating centers. The company found that its ability to make strategic moves in a highly dynamic market was hindered by the inflexibility of its computer systems. In particular, the rigid, layered architectures and 3GL code of its mainframe system were restricting the company's ability to modify its information systems to reflect the operational impact of organizational changes. The company was using MSA's financial accounting software running on an IBM 4381 mainframe. As a result, high operating, upgrade and maintenance costs for both hardware and software had become financially burdensome. In addition, a large IS staff was required to provide support, thereby increasing the costs associated

with operating and maintaining the company's mainframe system. The company was also challenged by a growing number of incompatible user-created systems that were in widespread use throughout the organization.

System requirements:

The company decided that to continue to grow, it would have to combine its multiple, fragmented systems into a single, fully integrated, decentralized environment. To accomplish this, the company chose to move to UNIX-based client/server systems. They also decided that their new systems would be based on a relational database and 4GL tools to provide flexibility, easy access to corporate information, and a rich development environment that would allow for easy modification. The specific goals that the company set for their new systems were:

- 1) to fully integrate all systems in a decentralized environment,
- 2) to provide a rich development environment for internal IS use in creating and modifying company-specific applications,
- 3) to maintain the audit controls and accounting functionality of a mainframe system, and
- 4) to provide the flexibility necessary to accommodate the CEO's desire to be able to change the accounting systems to reflect the dynamic corporate structure.

Solution milestones:

The first step in the project was the complete retraining of the IS staff in UNIX and 4GL technologies. As this was being completed, the company installed new UNIX 4GL-based

accounting software on a development platform so IS personnel could experiment without disturbing existing applications. In addition, the accounting software that the company selected was customized to their needs and installed for them.

After the installation of the new system, the data conversion process from the IBM 4381 was completed. Following this, the old and new accounting systems were run in parallel for a period of time during which the new system was fully tested. At the end of the test period the new system went live.

Benefits:

With the implementation of new UNIX-based client/server systems, the company realized an 80% reduction in their overall information systems operating costs. A significant portion of that reduction came from the dollars saved due to reduced hardware and software maintenance costs. In addition, the new 4GL development environment led to a 500% productivity increase in application development. This has allowed new applications, needed to address dynamic business needs, to be completed in a fraction of the time they previously took.

The new client/server systems have also resulted in substantial increases in the operational effectiveness of the company. The "openness" of the new systems means that all operational and accounting software systems are fully integrated and, as a result, company managers are better able to review and control financial activity and business performance at remote operating centers. As an added benefit, reporting-structure changes, needed as a result of the company's response to market opportunities, are easily handled with the more flexible open systems-based software.

CLIENT/SERVER: DOLLARS & SENSE

It would be wrong to look at the many benefits of client/server without also focusing on the financial justification accompanying such a project. For top corporate decision makers, the bottom line on client/server systems is "Do they make dollars and sense?" As with all company investments, new computer systems must be measured by the return on investment that they bring to the corporation.

To give you some idea of the tremendous financial benefits possible with client/server-based accounting solutions, what follows are actual financial numbers from three major companies who made the move to client/server systems. Their returns on investment are typical of such projects and they show that client/server corporate accounting solutions deliver tremendous savings over traditional mainframe-based systems.

Leading Transportation Company

A \$300 million transportation company in the Midwest was paying \$1.2 million annually in outsourced information-technology expenses before downsizing to client/server systems. The company decided to move from outsourcing to an in-house UNIX-based system to trim costs. The total acquisition cost for their new system came to \$670,000 for hardware and software. With the client/server UNIX-based solution, the company's annual information technology expenses were reduced to \$500,000 - an annual savings of \$700,000. As a result, their investment in the new system was recovered in just 12 months.

Major Retailer

A \$1.5 billion retail company was paying \$9 million annually in information systems costs before implementing a client/server solution. Their one-time acquisition cost for a UNIX-based system, including hardware, software and custom programming, totaled \$4 million - less than half the yearly expenditure on their existing mainframe system. After downsizing, annual information systems costs were reduced to \$4 million - an annual savings of \$5 million. As a result, the new system investment was recovered within 10 months.

Rapidly Growing Technology Company

A \$250 million health-care technology company was spending \$460,000 annually on maintenance alone for their mainframe system's hardware and software. The company decided to downsize to a client/server solution at a total cost of \$650,000, including hardware and software. After downsizing, the company found that the annual maintenance on hardware and software amounted to only \$60,000 - an annual savings of \$400,000. Thus, within 20 months the initial investment in the new system was recovered through savings on maintenance costs alone.

ISSUES TO CONSIDER WHEN MOVING TO CLIENT/SERVER

The decision to move to client/server brings with it a wide range of challenges and opportunities for both the IS department and users alike. Careful planning and thoughtful consideration of the many issues involved can mean the difference between success and failure in implementing these new systems. The following are some key issues that have proven to be most critical in

companies' client/server migration plans. While the list is not all-inclusive, it should help you in formulating a successful implementation plan that addresses the major IS and user department concerns.

Re-engineer business processes

Migrating to a client/server environment provides an excellent opportunity to re-engineer existing business processes that have proven cumbersome and inflexible. Corporate users should ask themselves how many of the procedures they have in place have been imposed by the computer systems and applications they are using or have used in the past. Does the system adapt to and meet the specific needs of the company, or does the system force the company to adapt operations to it? Client/server offers companies the chance to re-design and simplify business processes to streamline operations.

Consider security and controls

Despite impressions that people may have of UNIX from its early years, today's UNIX-based systems provide excellent controls and security features for corporate users. The best UNIX-based accounting applications are designed to mimic mainframe features in this regard.

On the client end, when considering the move to a Windows-based client/server accounting solution, it is critical that companies review the product to find out how the application handles transaction and data integrity. Any weaknesses that an accounting system has in this area can be easily and quickly revealed through careful questioning during software demonstrations.

Training is critical

Client/server systems require an entirely new skill set for IS professionals. They must not only understand broad new concepts such as open systems, distributed computing and client-server, but also develop skills on new products such as UNIX, Windows, TCP/IP, relational databases, 4GLs, and object-oriented programming. Companies moving to client/server-based accounting systems should either re-train their existing IS staffs in these new skills or, if that is not possible, hire new programmers with hands-on experience in these new technologies. Project plans for implementing client/server systems should always include action items to insure that IS personnel have the necessary skills to both install and support the new systems.

Communicate goals to all involved

The move to client/server almost always brings with it some rebellion among the information systems staff and often the system users as well. Many IS staff members may be concerned about job security while users may resist anything new and unfamiliar. The challenge for management is to address individual concerns while focusing on the benefits to the corporation.

One key to success is to insure there is a high level of communication with those staff members who will be affected by the new system. In the case of the IS staff, management should communicate its commitment to training existing staff in the new technologies. If staff reductions are necessary, management should let people know that the company will look to attrition, instead of layoffs, to achieve those numbers.

People should also be made aware that the new systems are being implemented as part of an overall effort to align IS goals with the organization's priorities. Goals for the project should be clearly defined and should be stated in business, not technical, terms. Also, compensation and incentives should no longer be tied to the size of the department's budget or its headcount. Instead, salary levels and rewards should be based on the department's ability to solve the company's business problems with cost-effective solutions.

These ideas, coupled with examples from other companies' client/server successes, should be broadly communicated to help motivate the IS staff. Likewise, users should be made aware of the many benefits that the new systems will bring. By championing the new systems to IS staff and users alike, IS management can play a critical role in moving the organization to new levels of success.

Make it a team effort

Having the right partners can make all the difference to a client/server implementation. Along the road to client/server, difficult issues often arise that can hamper the success of the migration process. One of these, for example, is data conversion. Getting data from the old proprietary systems to the new client/server systems can be one of the most difficult parts of the entire project. Without some help from outside experts, the project may slow down or even fail.

Fortunately, experienced professionals are available to handle these and other specialized tasks involved in moving to a client/server environment. An effective team can be assembled to address all hardware, software and integration concerns. This team should be headed by the IS department.

To be certain that a potential partner has the necessary experience, references should always be carefully checked. Corporations need to make use of the expertise of those who have proven results in making client/server work at a number of companies in a wide variety of industries. If a potential vendor cannot supply real and recent references, the vendor should not be selected.

Accounting systems are, as we discussed earlier, the heart of an organization's information systems. A company should not trust this valuable resource to anyone who does not have proven expertise in successfully implementing such systems for companies in a variety of industries.

CONCLUSION

Information management has become increasingly important to competing effectively in today's business environment. Having the right information, easily available, at the right time can help an organization make strategic and operational decisions that lead to competitive advantage and corporate success.

Client/server systems offer companies the flexibility and ease-of-use they need to manage their information. Proven success stories from companies who have made the move to client/server show that for most large companies the operational benefits and significant cost savings of client/server far outweigh the potential short-term problems associated with moving to a new computing architecture. And, when carefully thought through and planned, the transition to client/server can be handled very smoothly with both IS and user department personnel feeling good about the change to the new systems.

Accounting applications represent a perfect area for companies to focus their initial client/server efforts. Today's client/server-based accounting packages offer better performance, functionality, and user control than the mainframe accounting applications that they often replace.

In addition to saving companies money, these new systems leverage the technology strengths of UNIX, Windows, relational databases and 4GLs to deliver faster access to information, ease-of-use, ease-of-modification and greater interoperability with external systems.

For the IS department willing to accept the challenge, the move to client/server brings with it a redefined leadership role of even greater importance within the organization. A client/server-based computing strategy offers the IS department an opportunity to show a clear contribution to the success of the company, as well as significant cost reductions and improved ROI. Users also are quick to recognize the success of a client/server implementation, and their opinion of the capabilities of the company's IS resources is often greatly enhanced.

The proven success stories from companies who have accepted the challenge of migrating to client/server demonstrate that client/server represents a new level of price/performance and flexibility for corporations. The demonstrated benefits of client/server are leading companies to view the client/server decision as one of "when?" rather than "if?". Companies who answer that question by developing carefully-considered, near-term migration plans will be positioned to compete in the information-driven business environment of the 1990s.

BIOGRAPHICAL INFORMATION

John M. Woolsoncroft is the Vice President of Marketing for Concepts Dynamic, Inc. in Schaumburg, Illinois. In this role he is responsible for the planning and execution of marketing programs for Concepts Dynamic's client/server-based financial accounting software products.

Mr. Woolsoncroft has been involved in the selling and marketing of information systems hardware, software and services to major corporations for over fourteen years. Prior to joining Concepts Dynamic, he was with Andersen Consulting where he directed the worldwide marketing efforts related to Andersen's distribution and logistics software. Mr. Woolsoncroft began his career with IBM's Data Processing Division in 1980. He holds a B.S. degree in Business Administration from the University of Illinois and an MBA degree from the University of Chicago Graduate School of Business.

Mr. Woolsoncroft is a noted author and speaker on information technology and its application to business strategy. He has presented at the national meetings of such organizations as the Data Processing Management Association (DPMA) and the Association for Computer Operations Managers (AFCOM). In addition, his insights on trends in technology and marketing have appeared in such publications as Computerworld, Information Week, Sales and Marketing Management, and the Wall Street Journal.

SELECT BIBLIOGRAPHY

- Bozman, Jean S. November, 1992. "Informix Chases DBMS Rivals." Computerworld.
- Cunningham, Peter. September, 1992. "UNIX Means Business" supplement to Open Systems Today. pp. i-xii.
- Faden, Mike. September, 1992. "When Downsizing Means Saving \$60 Million for the Company." Open Systems Today. pp. 88-91.
- Faden, Mike. September, 1992. "Can't Afford These Damn Big Glass Houses." Open Systems Today. p. 88.
- Forrester Research, Inc. 1992. "The Computing Strategy Report: The Mainframe Voyage."
- Gill, Philip J. January, 1992. "What is Open Systems?" UniForum Monthly. pp. 23-28.
- Informix Software, Inc. 1992. "Informix and Hyatt Hotels Success Story."
- O'Donnell, Patrick. June, 1991. "Don't Overlook Human Issues." UNIX Today! pp. 72-74.
- Parker, John. Fall 1992. "Making the Unix Connection: Commercial Potential." VARBUSINESS. pp. 17-22.
- Pike, Helen. November, 1992. "Unix's Corporate Climbing." InformationWeek. pp. 56-57.
- Poole, Gary Andrew. January 1992. "Open Systems in 1992: How will the global market fare?" Unixworld. pp. 73-78.
- Snell, Ned. November, 1992. "Mainframe Accounting Moves Down." Datamation. pp. 112-115.

Paper Number 2013
Thinking About an Image Management System:
Think Again and Again

Stephen Fontana
Director of Imaging Technology
Software Systems Technology
305 Broadway
New York, NY 10007
212.964.9600

Handouts will be provided at time of presentation



**Paper Number 2014
Data Center Consolidation at Hewlett-Packard**

**John Podkomorski
c/o Ella Washington
Hewlett-Packard Co.
408.447.1053**

Handouts will be provided at time of presentation



INTEREX 94

INDEXING:
WHEN, WHEN NOT, AND HOW

Paper #3000

Mark A. Gardner, CDP
Director of MIS

Bradmark Technologies, Inc.
4265 San Felipe, Suite 800
Houston, TX

1 (800) 621-2808



**INDEXING:
WHEN, WHEN NOT, AND HOW**

The topic of indexing is nothing new, but has certainly been pushing its way to the forefront of current issues. With the advent of TurboIMAGE 4.0/iX, indexing is now a part of life for any Hewlett-Packard Spectrum shop via the TPI (Third Party Interface - Hewlett-Packard's new interface that allows IMAGE to directly recognize and utilize third party indexing tools). It's valuable to understand why indexing is important and how to know when/where to use it. Indexing is not a panacea to all DP shop problems, but it can provide a great deal of benefit if incorporated properly. This paper focuses on indexing in general and is not tied specifically to the IMAGE database, but databases in general.

Indexing is a tool that accomplishes several well-defined tasks, and provides a great deal of benefit to those incorporating its use. Those benefits can be identified in general terms, irrespective of the platform they are being used on.

To get the most out of indexing a database, it is imperative to first understand (or at least be cognizant of) what an index is. Indexes are typically stored in a separate file (from your data) and are comprised of two basic parts; a replica of a data field (or fields) from your dataset, and a pointer to the location of the record it represents. The data represented could be the name of a customer, a zip-code, an address line, etc. The purpose of storing a replica of your data in this separate file is to provide direct (key) access to that field in your dataset.

Now, you probably already have some keys setup that you use to access your data, but what if there is not a key on the field you need to access? This requires you to read all the records to locate the record(s) you want (typically, a time consuming serial read process). Perhaps there is an existing key on the field you want to

index, but sometimes you need a range of values, or a partial key lookup (i.e. using wild cards). These types of retrievals are rarely addressed by most database key structures, but are addressed by indexing. The benefits of indexing focus on providing faster, more flexible access to your data.

Two main sections will be addressed. The first section deals with the benefits of indexing, and gives several examples of different situations in which they can be applied. The second section describes situations where indexing should not be used. Both sections together will give you a better understanding of when, where, and how to get the most out of indexing.

EXAMINING THE BENEFITS OF INDEXING

The benefits of indexing stem from the increased speed and flexibility of accessing your data by any item (or field) in any dataset and by being able to specify wild cards, ranges, etc. Perhaps the best way to examine some of the prime features that make indexing so valuable is through examples.

Partial/Generic Retrieval

Partial/Generic access refers to being able to include such wild cards as the @, the ?, and the # in your argument. When you only want to specify part of a value (or only know part of a value), normally you have to read all the records in the dataset. Partial/Generic access allows an argument with wild cards to directly qualify the appropriate records.

For example:

STR?NG	STR?NG@	??RT?N	@ITE@	12JP#3
=====	=====	=====	=====	=====
STRING	STRANGE	BARTON	UNITED	12JP73
STRONG	STRING	BURTON	UNLIMITED	12JP63
STRUNG	STRINGER	MARTIN		12JP33
	STRINGING	MORTON		
	STRONG			
	STRUNG			

Relational access

This topic is perhaps one of the most important features of indexing. Relational access refers to the use of the Boolean operators AND, OR, and NOT, and provides the ability to include these operators in your arguments. For example, if you needed a report containing all customers with a last name of either Smith or Jones, you would normally execute a serial read, or run the report twice (once for each name). Indexing provides this access directly. Now you can specify an argument of "Smith OR Jones" and extract the combined list of both names at once. The power of this access creates great flexibility in entering your argument(s).

Range retrieval

Perhaps one of the most frustrating lookups is trying to extract a range of values. Monthly reporting is a prime example that almost all businesses face. Since most database key structures do not support range retrieval, the issue of serial reads always exists. Most companies would like to store more historical data on-line, but the more data, the slower the serial reads become. For example, trying to produce a report of last months transactions requires a serial read of all the transactions you have. If you store three years worth of transactions, you must search through 35 extra months of data just to locate last month's.

Indexing allows direct access to any range of values. This means that the previous example would only read one month of transactions instead of 36. This not only allows for faster access, but now you can maintain as much history on-line as you would like without concern for slowing down your reports. And, as the following section will describe, your data is returned in sorted order automatically.

```
>=920101<=920131  >=A@<=C@  
=====
```

```
920101      ALBERT  
920105      ALLEN  
920120      BILL  
920125      BROWN  
920131      CHARLES
```

Sorted retrieval is automatic

Most reports require data to be sorted in a particular order for the reports to be useful. Chronological order, hierarchical order, alphabetical order, numerical order, etc... are at the heart of turning data into information. Indexed access can automatically extract data in sorted order by value. This means that if you create a customer report and select by customer name, the index will extract your customers in sorted order by customer name. If you want to extract a range of transactions by date, the transactions will be returned in date order. The idea here is that your data is retrieved in sorted order by the field you have indexed. This means your reports no longer have to do most of their own sorts (which can save quite a bit of time).

Concatenated index: Fields consistently used together

Concatenation allows an index to be comprised of several data items. This is common for data items (or fields) that are consistently used together. For example, you can make an index out of Region, Warehouse, and Part-Number. This type of index allows you to qualify a record based on the values of all three different fields in a single lookup. This can greatly reduce the time required to select the desired records. Concatenated indexes also provide multi-item level sorting (often used in control break logic (e.g. sub-totaling within a report)).

Creating an index out of the middle of an item

The problem of needing to access a certain number of digits out of an existing field is a very common issue. For example, a Part-Number is typically comprised of several digits or ranges of digits, each piece containing some unique information (e.g. region, department, assembly, etc.). Through most databases, you typically have to execute a serial read to extract records under these criteria. Indexing provides a much better alternative.

Indexing allows the creation of an index out of any range of positions within a data item. This will allow you to directly access any piece of a field that is important to you. This is typically referred to as an "offset", and still supports Partial/Generic, range, relational access, etc.

To take this concept one step further, consider the example of concatenating an item to itself. Many cases exist where the format that your data is stored in is not the way that you would like it extracted. Perhaps the most common scenario involves date fields. Most reports like dates sorted first by year, then month, and then day. This allows chronological reporting, either in an ascending or descending manner.

Many dates are not stored in the needed YYYYMMDD order, but instead in a MMDDYY or DDMMYY order. This would normally require your report to re-order the date field (probably after having to execute a time consuming serial read to extract a range of dates) so that it can properly sort the report's information. This is no longer necessary with indexing.

Since indexing tools allows you to build an index out of any range of characters within a data item, you can also create an index with a different digit order than the original data item. For example, a date of MMDDYY can be stored in an index as YYYYMMDD. This index eliminates the need for your report to execute a serial read, and eliminates the re-ordering and sorting procedures from your report (remember, indexes can automatically return your data in sorted order).

This is done by concatenating the same date field to itself several times, each time specifying which part of the original item should be first, second, etc. Now you can specify an argument like `>=920101<=920331` to extract (via indexed access) all those records with a date between January 1, 1992 and March 31, 1992. Again, this is indexed access, not a serial read like you were performing before! Just watch how much faster your report will run without having to execute either the serial read or the sort!

Approximate Match Retrieval

If you are looking for an entry that does not actually exist, an index (stored in a B-tree structure) can find the next best thing; the closest matching entry. Indexing will allow you to retrieve the entries that are closest to the actual value entered. This is very useful in situations where misspellings may have occurred at data entry time or proper spelling is not known at lookup time.

Small Percent of Records are Needed

Indexes are highly effective when a small percentage of the records in a dataset meet the necessary criteria. The smaller the percentage of records qualifying, the more time you will typically save. For example, if you currently maintain three years of transaction history in your database, to extract the transactions for a single month will require a serial read through all 36 months of transactions. With indexing, you can now read only the records for the single month desired, eliminating the reading of the additional 35 months of transactions!

EXAMINING THE LIMITS OF INDEXING

Indexing does provide a great number of benefits, but should not be applied to all situations of data retrieval. The following are some example situations where indexing (in general) does not apply.

High Percent of Records are Needed

If a given report requires access to all (or a high percentage, e.g. > 50%) of the records in a dataset, an index should not necessarily be used. This is because to extract records from your dataset, an index has to be read, then the corresponding data record, then another index, then the next data record, etc. It is often faster to execute an accelerated serial read of the dataset in such a situation.

Few Unique Values Exist

A second example of when not to use an index is when a field only has a few different possible values. For example, a geographic survey may have a location of either "NORTH" or "SOUTH". Since

the number of records qualified by either argument will be approximately 50%, an accelerated serial read may be faster in this situation.

Key Access to Every Item

Even though indexing does allow any/every field in any/every dataset to be an index, it probably is not wise to do so. As records are added and deleted, machine resources are required to maintain each index to be in sync with the data it represents. The larger the number of indexes, the more time required. Select those items that would need indexed access on a recurring basis, or are not modified often.

Too Big for Indexing

It is always better to keep indexes as short as possible. This speeds retrieval and saves disc space. Typically, indexes should not exceed 20 characters in length.

Unused/Oversized Indexes Waste Disc Space

Indexes that are not used should be removed. These extra indexes use extra disc space, and require CPU time to be maintain. Also, indexes that are larger than needed (contain more characters than are actually used for extractions) take up extra disc space unnecessarily. Being efficient in your index design can greatly affect the amount of disc space required for indexing.

SUMMARY

With the increasing popularity of indexing, it is important to be aware of not only its potential benefits, but also its limitations. Several major issues have been identified on both sides of the coin. Being informed is always important when considering another technology. With some of these basic guidelines in mind, indexing can provide a great deal of benefits to most DP shops, both in terms of speed and flexibility.

RELATED TOPICS FOR DISCUSSION

1. TPI (Third Party Indexing) - Indexing within IMAGE.
2. Indexing and Client Server issues.



Criteria For Relational Data Base Management System Selection

James A Hepler
Hewlett-Packard Company
39550 Orchard Hill Place
Novi, MI, 48376-8024
810-380-2207

Abstract

With the industry movement towards standards based Client/Server Open System environments there is often a necessity to properly select the Relational Data Base Management System that is most appropriate for the needs of the user community and thus the business needs of the company. Often these needs are immediate but require a long term strategic view of the future.

Many customers have asked for a check list or decision tree or methodology to select which RDBMS is best. While it is impossible to create a methodology that will work for every customer situation, it is extremely useful to understand the features and benefits available in RDBMS and what types of information to gather from the Data Base vendors and internal to the company to have an organized approach or methodology to make the optimal decisions. This paper will not recommend a specific vendor or even list the benefits and advantages or disadvantages of RDBMS. It will instead equip the data administrator or other MIS personnel to establish the specific criteria of importance to the successful selection of RDBMS for the strategic business needs of the company and end user.

Some of the topics to be covered include security, performance, SQL compliance, programmatic interface, distributed data environments, transaction monitoring and control needs, middleware, user interface tools, and backup/recovery capabilities.

Introduction

Many computer users are moving towards technology solutions often referred to as Open Systems. While there are many definitions of what Open Systems really means, the basic concept includes:

- Portability
- Interoperability
- Scalability
- Availability

Without getting into a long discussion on what it takes to have an Open Systems environment and all the benefits it entails, let us just state that openness is the ability to choose a variety of *off-the-shelf* products, applications, and technologies that are based on standards. These standards may be industry defined or de facto, but they must be widely adopted, available at a reasonable cost, maintainable, and likely to be sustainable for a suitable length of time.

One of these standards is the Structured Query Language (SQL). Many data base management systems state they have relational access methods or are similar to relational in structure. For the purposes of this paper, we will define Relational Data Base Management Systems (RDBMS) to mean those that are SQL compliant. RDBMS selection is usually in the critical path in moving to Open Systems. The suitability of the RDBMS to the four concepts of Open Systems above forms the basis for choosing the RDBMS supplier or suppliers.

Many Open Systems adopters start their movement towards Open Systems by selecting a hardware platform or application package. This selection then directs them towards a particular RDBMS or perhaps reduces the choice to two vendors. Other adopters select their RDBMS first and then select the hardware platform and application packages based on what is available for that RDBMS. Commonly, as the business evolves, there is ultimately a collection of hardware platforms and RDBMS and indeed other file structures that are part of the environment adopted.

Whatever situation exists, the process contained in this paper will provide conceptual criteria for selecting the RDBMS with the best fit for the requirements as defined by the computer user.

Relational Data Base Management Systems

To take fullest advantage of the benefits of standards based open systems, it is critical to choose a Relational Data Base Management System (RDBMS). These RDBMSs are available from a variety of vendors and while they all have basically the same backbone of features, the various vendors have different degrees of standard compliance, different types of tool sets available, different implementations of client/server and networking features, and differing levels of performance characteristics.

The relational data model was described by Dr. E.F. Codd of IBM in 1969. There have been many modifications of that proposal since. A standard approach and measuring stick was not available until February of 1987 when Structured Query Language (SQL) became the official standard for data base language by the American National Standards Institute (ANSI). When a RDBMS is discussed, that generally means SQL compliant data bases using an internal structure encompassing the Relational Model.

The relational data model defines all data and access of the data as simple tables made up of rows (records) and columns (fields). The advantage of this method of structuring data is that data base and application design is easier because all data is in easy to conceptualize tables. New tables can be defined and related to previously defined tables. Column or field changes can be made easily if new fields are needed in existing data bases. In summary it is easy to access your data quickly and flexibly.

SQL uses English like commands to easily access and manipulate data in sets rather than one record at a time. This means that programming with SQL means less code, simpler logic, and easier maintenance in the future.

The relational model makes data access flexible and improves programmer productivity.

Relational Technology -- A Productivity Solution - Hewlett-Packard

Programmer productivity is greatly enhanced with interactive program development and CASE tools available with RDBMS. Debug facilities can be interactive allowing faster development because code can be tested with immediate on-line results. Test data bases can be easily created and tested against as needed. In addition, programmers can be productive in different operating environments -- often without retraining.

Depending on the RDBMS selected, development tools are available from either the data base vendor or other third parties to allow even more elegant and powerful development with graphical user interface (GUI) tools that allow program development with point and shoot techniques.

SQL code is more readily portable across different hardware platforms and even with different RDBMSs because of the standard for SQL. This gives customers greater vendor independence and flexibility in the future. Virtually all major hardware, operating system and RDBMS vendors have adopted this concept of openness.

Data base administrators also receive many benefits that accrue back to the company and the end users. Data design and access are simpler. Future access paths do not have to be predicted since any data group can be accessed with a logical view when required. In contrast, non-relational and flat files must have access paths predefined which is inherently inflexible. Modifications to a RDBMS are also much simpler. Since access paths are not predefined, rows and columns can be added without affecting existing applications.

Depending on the RDBMS vendor and implementation of Unix, Dynamic Space Allocation is also possible. This allows the DBA to control disk allocation to optimize performance and reduce disk space utilization to a minimum.

Where appropriate, some RDBMSs allow indexing to be created to enhance performance. The SQL optimizers available from various RDBMS vendors use the fastest access method and often statistical algorithms to find the data selected in the most efficient manner.

Access control is also simplified because access can be controlled down to the field level if necessary. Other levels of control include rows, columns, views, tables, system catalogs, and entire data bases.

Relational Technology is preferred when the application requires flexibility, growth, or change.

Relational Technology -- A Productivity Solution - Hewlett-Packard

Selection Process

In selecting a RDBMS, it is best to reduce bias as much as possible by creating an environment that will allow a fair evaluation of all vendors. It may be useful to hire outside consultants to help with the selection process to reduce internal bias

and to add a degree of experience to the process. It may be worth the cost to make the decision quickly and accurately while protecting the company and users.

Many computer users start by listing capabilities of the RDBMS and then designing their applications or selecting their applications based on what the RDBMS facilities can support. It is generally better to analyze the needs of the users and the company first, and then evaluate which RDBMS meets those needs the most closely. Clearly the viability of the vendor is also part of the criteria.

Part of the evaluation process will include an analysis of which needs are not being met and the cost or effort for the MIS staff to provide the missing pieces or create integration as required. Also a risk analysis should be done to quantify the factors that may impact the company or users while the implementation is taking place and after completion.

If possible, review the success of other similar companies with the same RDBMS. During the selection process hands-on evaluation is extremely valuable. If hands-on access is not possible during the evaluation, then the plan should include some alternatives if the RDBMS becomes unsuitable during the trial or startup phase of implementation. One of the advantages of the SQL standard is that the change from one RDBMS to another early in the cycle is of relatively low effort and cost - therefore feasible and with low risk.

An analytic approach is also useful to help quantify the results of the complex layers of information. The method illustrated herein includes a high level criterion approach to filter the vendors to be analyzed. This filter process is to eliminate those vendors that just cannot meet the minimum requirements to do the job. After the vendor list has been reduced to a manageable level, the next step is a more detailed analysis with weighting factors to accentuate the most important factors to the customers and user community based on the needs analysis/requirements definition.

There are therefore four phases in the selection process for RDBMS:

1. Needs Analysis/Requirements Definition
2. Vendor Evaluation
3. Product Evaluation
4. Hands-On/Trial Evaluation

There may be some overlap or review of previous phases, but in general the phases are sequential and should be managed as a project with a project manager assigned if the selection team is large enough. A project manager can make sure the effort is coordinated and that team member commitments are fulfilled. Without

direction, these types of efforts often drag on much longer than necessary and success is less likely. If necessary, a project manager can be brought in from an outside consulting firm.

Needs Analysis/Requirements Definition

Note: The terms needs analysis and requirements definition are interchangeable for purposes of this document.

The needs analysis/requirements definition is the basis for the selection and is therefore extremely important to be done as rapidly and as carefully as possible. It is often the case that a company relies on one individual who is their resident data base guru to make the decision for them. This is frequently a mistake because of the narrow focus a data base guru generally is allowed. It is usually better for all concerned if a committee is formed to do the needs analysis.

If you are selecting a RDBMS for a single PC or workstation, clearly this is not necessary because there is only one user -- but if it is for a multi-user commercial application of any degree of importance it is imperative to put a reasonable number of heads together to do a complete analysis to allow a sound business decision later.

The group should typically include both technical and end user personnel. Outside consultants should be considered if there is a shortcoming in the availability of in house expertise. Both the on-line and batch oriented parts of the application will need to be considered. It is important to have people on the team who understand that part of the business requirement. The broader the expertise included on the committee, the lower the probability that something will be overlooked later. Another benefit to wide representation is that the selection process will enhance the acceptance of the solution by the various parts of the company or user community.

After deciding who will do the requirements definition, the needs analysis can begin. This task is usually multi-level with parallel efforts proceeding. This means that while the User Profile is being developed other parts of the selection process will be proceeding. One of the steps in the needs analysis is to establish a user profile. The other steps follow.

User Profile

The user profile is simply a list of who will use the system, what they will need to do, when they will need to do it, where they will be located, and how frequently

they will be doing it -- the five W's minus the why and plus the how. The type of information that needs to be gathered here includes:

- Data sharing requirements -- one user at a time or concurrent
- Types of access -- read only, heads down data entry, etc.
- Availability requirements
- Skill levels of users and access tool requirements

Processing Types

Is this application typically On-Line Transaction Processing (OLTP) or Decision Support? If it is OLTP how large are the transactions? For example, an order entry application has a smaller number of large transactions than a payroll application does. An executive support application is generally read only with a large component of Ad Hoc inquiries.

If it is OLTP, it is important to include information on total number of concurrent users, the number of transactions and size of them, and how large the total data base will be.

For Decision Support, factors such as how much data is downloaded from other systems and how often are important. How large and how frequent are the hard copy reports that are generated. What are the requirements for Ad Hoc inquiry?

If there are both OLTP and Decision Support, do they access the same data base and if so the same Views or Tables?

Application Requirements

Naturally the application requirements tend to drive the RDBMS selection. If the application is already in house and is meeting user requirements but is being transferred to an open environment, is the code transferable? Can the current data structure be migrated to the selected RDBMS with a reasonable effort? Will the performance be suitable or will re-engineering of certain parts of the application be needed?

In the case where an application is being purchased, the application may be available on several RDBMS and therefore, criteria such as what does the application provider recommend for the RDBMS should be considered. If the provider does not have a standard recommendation for the customer's environment, the selection process can proceed as if it was a user written application.

Another scenario is that the application will be written from scratch with new state of the art development tools such as fourth generation languages (4GL), case tools, etc. Another possibility is that the application will require client software such as windows based point and shoot functionality for ease of use. If so, the RDBMS will need to have an Application Programming Interface (API) that allows that. Most of these tools are SQL based and will work with most RDBMS, but not all.

If the application is being developed in house, clearly the debugging aids and productivity and tuning tools are also important for the developers. This is less important for the support of purchased applications, but may be weighted to reflect that.

Availability Requirements

Availability defines the time of day and week that the data must be available to the users. It also defines the acceptability of downtime -- or more properly the allowable recovery time from unscheduled downtime whether caused by hardware or software or human error. It also should encompass the frequency and duration of scheduled downtime that may include backup and other batch activity. In the case of decision support or data warehousing types of applications, it may include the daily update or rebuilding of the data base including the download from other systems or data bases.

Service Level Agreements (SLAs) should be written to describe the availability requirements and performance criteria for each application. These agreements should be an important basis for RDBMS selection as each data base has different backup and recovery features as well as logging facilities.

Naturally the solutions from the hardware and system management software vendors will have to be considered in this decision as well. Some hardware vendors have different facilities available such as disk arrays or disk mirroring or automated system switchover. These types of facilities may alter the importance of the RDBMS vendor providing such facilities as server replication through the network. What is of importance is to define the need for availability, then determine the optimum solution combining the RDBMS, the hardware facilities, the backup/recovery facilities, and perhaps the network facility.

Another factor affecting availability may be differing capabilities of different RDBMS vendors on different hardware platforms. For example, RDBMS vendor A may have a better backup/recovery facility on hardware vendor X than on hardware vendor Y. However RDBMS vendor B may have their best

backup/recovery speed on hardware vendor Z with their least effective solution on vendor X.

Operational Needs

The needs for operational staff to complete night time batch processing including daily, weekly, monthly, and year end must be considered. Also the system management tool integration features of the RDBMS should be considered. Is it possible to instrument and alarm the data base for events such as poor performance or premature shutdown? Can it be monitored for best types of SQL access?

What about integration with other RDBMS, file systems, or other operational environments? What about roll back or roll forward requirements to add to application functionality or recovery speed?

The portability and interoperability requirements should be defined here.

Distributed Processing

Some of these topics have already been touched on in earlier sections. Clearly if only one data base is going to exist on one system, the criteria to consider is much simpler. If there are multiple systems or perhaps client/server environments many other capabilities should be considered. The network topology and protocols to be used are important. The physical location of the servers will have an impact.

The needs analysis must consider where the data must reside and where it will be accessed from. Additionally, data replication features may need to be reviewed. This of course will have an impact on the availability considerations discussed earlier.

Future Considerations

After considering current requirements, the business plan should be enfolded to be sure the growth of the company or user community is included in the plan. For example, if in one year there will be a significant increase in the amount of data or demand on the data base, that should be planned. At a minimum, a two year business plan should be included in the RDBMS selection.

RDBMS Vendor Evaluation

Selecting a RDBMS means you are selecting a vendor to support you in the short and long term. The data base selected and the applications surrounding it need to have a long life expectancy to cost justify the move to open systems. Therefore

support is a critical issue in vendor evaluation. Does the vendor have a history of providing timely updates, bug fixes, enhancements, etc.? Are there likely to be performance improvements in the future? What about add on products?

There are dozens of Relational Data Base Management System Vendors. In evaluating them, one should consider the following areas:

- Breadth of product offering
- Support of current and emerging Open Systems standards
- Strategic direction for product line
- Company history -- especially financial stability
- Product pricing structure

Company Background

As your long term business partner, the company needs to be able to provide the support you require. This means when and where you require it and in a reasonable amount of time. To do this, the company must be financially viable and have enough employees in the locations where you need them. If your users or locations are scattered, the RDBMS vendor must be able to provide support at all or most of your locations as your needs analysis determines.

Their market position as well as revenue stream should be considered. Are they a market leader or follower? Do they have a sizable and stable installed base that is growing? Are new releases available on a regular basis or are they infrequent? What type and how many commercial application packages are available on their RDBMS?

All of these questions should be considered and weighted appropriately to the needs of the company and user community as defined in the needs analysis.

Support Services

Do not assume that standard support services that you expect are available. The following should be provided at a minimum.

Local System Engineering Support -- on site support when required. By phone when desirable. Both at the hours and response times needed. If you need 7X24 support can you get it?

Emergency or Priority Service -- when necessary, special situations may arise requiring personnel from the vendor with a deeper understanding of the RDBMS internals or features than the phone services might provide.

Training -- at training centers or on site if desirable.

Consulting Services -- for startup and for major critical times or to solve unique problems that may appear from time to time with new releases or changes in the business environment, consulting will be needed from the vendor or from third parties.

Software Update Policies

What is the frequency of releases of software for maintenance (bug fixes) and enhancements? How do they notify their clients and help them plan for the next release? What services will they provide to assist with installation of the new releases?

This information is needed for planning and operational purposes.

Documentation

Another area often taken for granted is the quality of documentation. Is it well organized? Is it readable? Is it available on a variety of media? Is it on line?

Does the documentation include quick startup sections in addition to more complete documentation for the experienced developers, administrators, and users? What about quick reference cards and integrated help facilities?

Pricing Structures and Licensing

The situation for a single system license and the associated price is much different than a distributed licensing arrangement. There may be multiple components of the pricing also. Tier and user pricing as well as volume end user discounts and corporate discounts may all apply. Ask for written documentation/quotes on all of these items. Be sure to evaluate total price and not unit price times the number of units you think you will have. The vendor may have a pricing model that changes as the number of units or tier changes.

There may also be runtime versions at a much lower cost than developers' versions. For multiple systems, this will control the costs appreciably. Additionally if you plan to re-market your applications, this should be a critical factor for you to consider. You may even be able to arrange a re-marketing license for the RDBMS to bundle with your applications and their runtime libraries.

If you can get a history of price changes for the last few years, to evaluate the trend for each vendor there may be information of value in your selection process.

Third Party Services and Products

A major factor in evaluating the viability of a RDBMS vendor is the number of third parties that provide either services or software based on that RDBMS. If this number is very low, then the vendor either has a short history or a questionable future.

As a data base product matures in market acceptance, a follow on market is created for third party applications, tools, and utilities as well as services. Value Added Resellers (VARs) and independent consultants are in business to build applications and services for this market niche created around RDBMS. If there is too small a number of these third parties, it may be an indication that the RDBMS vendor does not support third parties or the data base is not a viable one for application development.

For application and data base design and tuning it may be extremely important to have consultants available to assist and provide needed expertise.

RDBMS Product Evaluation

Items to consider when choosing a Relational Data Base Management System to meet the needs of today's commercial business customers include:

General Categories

SQL Compliance -- some are subsets of standard, some are extended

Database Capacity -- all have limits on rows, columns, tables, indexes, data bases, views, joins, and disk space

Data Types -- in addition to standard, some offer additional choices including object oriented

Data Integrity -- forces data type match, some allow null data, some do not allow duplicate values, etc.

Data Security Features -- different levels of security are supported

Concurrency Control -- controls contention with different types of locking and deadlock avoidance features.

Criteria For RDBMS Selection

Restart & Recovery Features -- different logging and archival features supported including distributed features

Transaction Monitoring Features -- different implementations of Rollback and Commit -- including two-phase commit (see separate section below)

Portability -- across hardware and operating systems platforms

Middleware -- what Middleware is available with this RDBMS? (See separate section below)

Client/Server Features -- separation of user interface from data base

Data Dictionary/System Catalog -- available for inquiries?

On-Line Transaction Processing Capabilities -- response times in on-line and mixed batch environments

Performance -- resource consumption per user, utilities for tuning and monitoring, indexing available, etc.

Ability to take advantage of multi-processors or parallel paths

Operational Management and Administration -- recovery after failure, responding to changing access types, integration with system management facilities, etc.

Tools Availability

Tool availability:

Data Definition -- full screen, menu driven

Ad Hoc Access -- full-screen interface for prototyping, etc.

Interactive Inquiries -- full screen "Query by Example"

Forms and Menu Creation -- automatic generation of forms and menus with edits without programming

Report Generation -- ease-of-use features and flexibility

Error Handling -- on-line information on error conditions available with recommended actions

Development/CASE Tools -- what tools available

End User Ad Hoc Inquiry and Reporting Tools

Embedded SQL Support -- support of SQL calls from third generation languages

Fourth Generation Language Availability -- tightly integrated with SQL.

Monitoring and Alarming Tools Backup/Logging/Recovery Utilities

The selection process should consider all of these items to some degree. Priority should be given to points of emphasis in the company's business plan. This is where the weighting parameters come into play.

Middleware

Middleware is a new class of solutions (usually software) which allow the creation of client/server applications. It is the "glue" that ties the client to the server. Middleware such as Enterprise Data Access/SQL (EDA/SQL) from Information Builders makes access of data on servers much easier for the application developer and end users. Many RDBMS vendors have Middleware tools to help provide linkages to data structures other than their own. Third party Middleware may or may not work with certain RDBMSs. If Middleware solutions are needed, that should be part of the decision on which RDBMS to purchase.

Transaction Processing Monitors

Transaction Processing Monitoring can be provided by products like Transarc's Encina, the Open Software Foundation's standard for such capabilities, or by several other products available in the Unix marketplace. Another TP Monitor is AT&T's Tuxedo. However a movement to Relational Data Base Management Systems often precludes the necessity of such a piece of software.

The feature set of most RDBMS protect the integrity of the data while reducing the maintenance effort of the applications. The cost of a RDBMS is generally much less than the additional maintenance and the cost of lost business opportunities that may result. Many RDBMS even support Transaction Monitoring through a distributed network. Some do not however and for multi-threaded distributed data bases a TP Monitor may still be required.

If coordination across multiple systems or multiple RDBMS is required or between non-relational file structures and RDBMS, then a Transaction Monitor may be required.

Hands On/Trial Evaluation

Often the RDBMS vendor will provide a trial copy of software and perhaps some assistance in configuring and installing the software. Many times this will not be possible until after some type of purchase commitment. Clearly if you can acquire the RDBMS package for evaluation before making the selection decision, the

opportunity to study the vendors claims is invaluable. It may be possible to review all of the evaluation criteria described above to some degree.

The support can be tested for responsiveness. Documentation can be reviewed. Certainly the operational aspects can be tried including backup/recovery strategies and monitoring of data base operation.

Ad Hoc tools can be tested. Application development tools can be used to prototype applications. Performance of batch and on line applications can be measured. In short, a proper albeit a relatively small volume but real test environment could be created and evaluated.

The best thing to do is review those items that are most important based on the needs analysis. It may also be useful to resolve uncertainties that you may have about the viability of the vendor as your business partner.

Weighting Criteria

The selection process described above is not intended to be complete and cover every issue that the practitioner may have to address. It is intended to be representative of the types of issues that are typically found during the selection process. Experience and judgment need to prevail in all cases where decisions are made and not just rely on the analytic process. The higher the weight assigned, the more important the criteria that is assigned the weight.

The scores will need to be assigned based on some expectation of results. For example, if the Batch Performance test has a goal of finishing in 8 hours, a finish by a vendor in less than 8 hours scores a 10. A finish in longer than 8 hours will score 10 minus the time greater than 8 hours. Other methods exist for scoring such as giving the best finish a 10 and the second place finish a comparative score. For example, the best Backup time was 3 hours; the second place was 4 hours. The first place is given a score of 10 with the second place being 33% longer -- they receive a score of 33% less or 6.7. Many other methods exist for scoring.

Earlier in the paper I suggested that weights should be applied in the decision process. The idea is that if SQL Compliance is your most important criteria, that should be given the highest weight. To illustrate look at the following short example. SQL Compliance is defined to be our most important criteria. Therefore let us give that a weight of 5. You could use any scale you prefer, but for this discussion the weight is 0 to 5.

There are three remaining RDBMS vendors under consideration and we have evaluated their products for SQL Compliance, Batch Performance, and Backup Speed. The series of tables below show the calculation of score for each row as weight times score for that vendor to give the total score for the criteria. The scores are then added to give a total score for that vendor.

	Weight	Vend A	Wgt Total
SQL Comp	5	8.2	41.00
Batch Perf	4.2	7.6	31.92
Backup Speed	3.1	7.7	23.87
Total			96.79

Therefore Vendor A has a total score of 96.79.

	Weight	Vend B	Wgt Total
SQL Comp	5	7.9	39.50
Batch Perf	4.2	6.1	25.62
Backup Speed	3.1	4.1	12.71
Total			77.83

Vendor B has a total score of 77.83.

	Weight	Vend C	Wgt Total
SQL Comp	5	6.2	31.00
Batch Perf	4.2	8.1	34.02
Backup Speed	3.1	8.9	27.59
Total			92.61

Vendor C has a score of 92.61. With these three criteria, Vendor A would get the selection slightly over Vendor C with Vendor B coming in a distant third. Vendor B was hurt by their slow backup speed -- thus the low score. This is a very simple case, but it illustrates the concept of weighted scoring. The Selection Committee should set up their own criteria list (the rows). The weighting factors should be assigned based on the needs analysis.

Case Study

This simple case study is based on experiences with an actual client. The basic assumption is that the hardware has been selected, so that all decisions involve the RDBMS. The Needs Analysis has been completed and the results reflect those needs. The following table shows the elimination of some vendors based on Vendor Evaluation:

RDBMS Product	Market Position	Standards Strategy	Distributed Features	Client Server Features
Vendor A				
Vendor B	NO		NO	
Vendor C				
Vendor D		NO		NO
Vendor E	NO	NO		NO
Vendor G				

Any vendor with a NO in any of these columns was eliminated because of general deficiencies in the company or product. The next step was to evaluate the surviving vendors -- Vendors A, C, and G. Application availability was the next critical feature that could eliminate a vendor. The following Table shows the results of this study.

Application	Vendor A	Vendor C	Vendor G
Distribution	NO	YES (1991)	YES (1994)
Financials	YES (1992)	YES (1992)	NO
Human Resource	YES (1993)	NO	YES (1993)

This means that to purchase all three applications, multiple RDBMS vendors would have to be selected. For example, Vendor A could be chosen for Financials and Human Resources, but Vendor C or G would have to be selected for Distribution. It would also be possible to search for another application or to plan on writing our own application if that made sense. In this case, it was decided to select two vendors. The product and vendor evaluations created the following greatly reduced table showing only the key elements.

Criteria	Vendor A	Vendor C	Vendor G
Technical Evaluation	225.60	246.73	143.99
Performance	160.06	147.75	180.34
Vendor Business Status	200.00	225.00	175.00
Risk	225.00	200.00	200.00
Total	810.66	819.48	699.33

The rows above are aggregates of the detailed criteria used during the selection process. The price and other factors that were not already included did not change the results further. On the basis of the table above, Vendor G was eliminated and the decision remaining was whether to purchase Financials from Vendor A or C.

Summary

By using an approach similar to the one described in this paper, it is possible to analytically decide which relational data base is the optimum for the needs of your company and user community. This is not an exact science but one requiring much judgment and cooperation within the selection committee and others who may be involved.

This type of methodology has been used successfully by customers and consultants to wade through and evaluate the large volumes of information available from vendors on their products. Personal biases that may exist can be reduced in order to make the best decision on RDBMS vendor. The methodology has been used to select application vendors and hardware vendors.

The criteria for evaluation described here could be applied subjectively or simply as a checklist for what the RDBMS vendors being considered have to offer. Many times this is just as useful as the analytic approach. The objective is to gather the correct type of information to make the best business decision possible and to document the criteria for that conclusion.

3002
An Overview of HP IMAGE/SQL

Hewlett-Packard Company
Commercial Systems Division
Database Lab
19111 Pruneridge Avenue
Cupertino, California
(408) 447-5649

This paper represents the collected works and efforts of several individuals from the Database Lab and Marketing organizations of the Commercial Systems Division of Hewlett-Packard Company. This paper will be presented by any one of these individuals at every major Users' Group gathering in 1994.

Some of you are new to IMAGE, and others of you have been familiar with the database since the mid 1970's when it was winning awards for function and design. Thousands of applications, from Email to finance to production control, all operate using IMAGE as their database management system. And if you count the users executing programs against data stored in IMAGE, their numbers would, no doubt, be in the millions.

Without argument, IMAGE gives you high reliability and high performance. After almost twenty years of improvements and refinements, you, as customers and users, would expect nothing less. With so much going for this product, with a dependability that few other database products offer, as a storage management tool that is a de facto standard on HP 3000 computer systems, what could possibly be done to improve it? The answer is quite simple. Standardize the access to IMAGE!

Hewlett-Packard is proud to introduce HP IMAGE/SQL, relational access to IMAGE data using industry-standard Structured Query Language (SQL). This method of access includes full read and write capability using ANSI standard functionality. This new access method makes a myriad of application development and decision support tools available to IMAGE that have never been available before.

The most astounding part of this whole story is that SQL access to IMAGE and the access that you enjoy today are completely compatible. Complete coexistence. No conversions, no recompilation, no changes. The only change will be the way you look at your data once you try the application development and decision support tools now available with IMAGE. And these are the same tools that can

be used with all the big name relational databases, including ALLBASE/SQL. IMAGE has moved into the Open Systems arena.

Those of you familiar with IMAGE will undoubtedly be asking the question, "but what will this do to performance?" Of course, every database design and application will have its own performance characteristics, but, as a general rule, IMAGE/SQL will perform at about 70 to 90 percent of native TurboIMAGE access. This term, 'native' refers to intrinsic level access using COBOL, PASCAL, et cetera. So, you will have the best of both worlds, fast native access and relational access.

With IMAGE/SQL you will have everything that you have today, wrapped in a new relational package, with an enormous selection of new tools, without any hassles. Let's take a look at the issues HP has tried to address.

TOP ISSUES

There are a multitude of issues which HP hopes to resolve with the introduction of IMAGE/SQL, and these will be explained shortly. Unfortunately, some new issues will be created as well, and these will also be revealed. These issues have been grouped according to the kind of organization that will be impacted, and they are VARs (Value Added Resellers) and ISVs (Independent Software Vendors), MIS or IT (Information Technology) Departments, and End-users.

Most VARs and ISVs, whose products include applications and tools, have been limited to non-relational application development tools when dealing with IMAGE. There are also very few PC-based GUI (Graphical User Interface) products which offer transparent interaction between PC clients and HP 3000 servers using IMAGE. As mentioned before, thousands of applications have been developed by uncountable companies all using IMAGE as the storage management system. Now, with IMAGE/SQL, all the new SQL-based application development and decision support tools, most using some type of GUI, can be used by VARs and ISVs to improve and enhance their products. This new transformation of IMAGE data into relational data will revitalize the VAR/ISV product environment.

Turning to MIS and IT departments, the primary issues are:

- o reducing the application backlog,
- o providing flexible information access,
- o recruiting SQL-trained personnel.

Each of these deserves some expansion.

Ideally, IMAGE/SQL should reduce the application development backlog for any MIS/IT organization. This reduction would be accomplished using any of the easy-to-use application development and decision support tools now available. Sadly, the opposite effect, that of increasing the demand for applications, will probably become true. As the end-users see the amazing results from new implementations using these tools, all kinds of previously hidden application requests will surface. Once the power of IMAGE/SQL is known, MIS/IT departments will be flooded with new requests for information.

Does this mean that these departments should avoid using IMAGE/SQL just to avoid this rampage? Absolutely not! The gains a company can realize from improved information review and analysis, as the result of improved data availability, can be significant. And MIS will not lose control. Users will be able to do their own queries and reporting, MIS need only supply the access. Some of the other issues in this area will further explain these statements.

The second issue in this area pertains to flexible information access. What this refers to is the powerful, command-driven language which characterizes SQL. Complex and sophisticated queries can be executed to supply end-users with their specific information requirements.

SQL uses command sentence constructs of SELECT and WHERE clauses, and the relational JOIN clause, to define its syntax. This level of knowledge is not necessarily required to use the various tools, however more sophisticated applications might require SQL expertise. Finding the right people to develop these queries and deliver these applications to the end-users will be critical. This brings us to the last issue for the MIS/IT department -- finding qualified personnel.

Relational concepts in information management have taken center stage in the last ten years. These concepts have become the de facto standard for data storage systems at educational institutions all over the world. As such, new and old computer professionals alike have had either some or extensive exposure to relational concepts.

SQL has become the relational language standard and is taught widely. Finding qualified personnel to program using SQL today is easier than finding experienced IMAGE programmers. So, even though your information investment is in IMAGE, the investment you make in accessing that information can be in relational technology.

The end-user has repeatedly come up in the last several paragraphs. Their issues are improved productivity and improved decision support. Both of these issues rest firmly on the same foundation; information that the end-user requires to be

more productive and upon which decisions are made must be located where the user can exploit it and be presented in a meaningful form. This can be achieved with the use of PC-based client/server tools featuring graphical user interfaces (GUI). These are the very same tools the MIS/IT department will be using for development and that VARs and ISVs will use to enhance their offerings.

THE NEW TOOLS

The new tools have been mentioned over and over. Before getting to specific products, let's first describe them generally, and then break them into categories to better understand what they can do and offer.

All of these tools are PC-based and function primarily in a client/server environment operating with Microsoft Windows. With exception, these tools use ALLBASE/PC API, which is based on the Gupta standard SQL API, as the link to the server.

For those of you not familiar with what an API is, some explanation is in order. API stands for Application Programming Interface and refers to the component of the client/server model which performs the interactive linking between client and server over the network for the purpose of data exchange. This can be thought of as application level handshaking, much the way RS232C is an electrical signal handshaking in data communications.

Besides Gupta, several other PC API standards exist. The use of the term 'standard' may seem vague as it is used here, but it basically refers to an agreed upon set of operating conditions. None of these standards represent an industry standard, but some type of API is required for client/server computing, so each vendor must select one or more standards with which to operate. So far, the Gupta standard serves the majority of the tools which have been certified for use with IMAGE/SQL. Microsoft has announced ODBC, its API standard, and this API is becoming a significant player in client/server technology. Support for ODBC will be available in January 1994.

Our first category of tool is Decision Support System (DSS) Tools. As the name suggests, the purpose of these tools is to support business decisions, and this is accomplished through information analysis, reporting and graphical representation. Typically, this type of tool is oriented towards end-users doing financial and managerial analysis where numerical quantification and graphical representation of data is useful. These tools also tend to offer formatting of data for subsequent importing to spreadsheet products for further manipulation and review. End-user knowledge of SQL is not a prerequisite for using these tools.

Some examples of Decision Support Tools are Impromptu by Cognos, Q+E Database Editor by Q+E, and HP's own NewWave Access. Although this last tool is not new, the relational access now delivered through IMAGE/SQL will dramatically reduce the overhead which will improve administration of these systems and make this solution easier to setup and maintain.

Our next category of tools is called Executive Information System (EIS) Tools. The key aspect of these tools is their exception management capability, where information is monitored within user defined limits and deviations highlighted. These tools use colors and graphics extensively to bring attention to situations where limitations have been exceeded. Other features include trend analysis and information drill-down. Drili-down simply refers to digging out the detailed elements of summarized information. A good example of an EIS tool is Forest & Trees by Trinzic which will be discussed in more detail later.

The last category of tools for use with IMAGE/SQL is Application Development Tools. Falling also into the category of Fourth Generation Languages (4GL), these tools offer PC Windows programming capability with Graphical User Interfaces which allow programmers fast development of critical end-user applications. In many cases, some knowledge of SQL is necessary.

Where DSS and EIS tools tend to be read intensive or read only, application development tools are intended to create interactive applications for online transaction processing. Some offerings in this category are PowerBuilder by Powersoft and SQLWindows by Gupta.

All the aforementioned categories make up the new tools available for use with IMAGE/SQL. These tools also operate with HP's ALLBASE/SQL and the other major independent relational databases. A review of some of these tools follows in the last section, and these tools are summarized in Table 1.

	Cognos Impromptu	HP Information Access	Q+E Database Editor	Trinzic Forest & Trees	Gupta SQL Windows	Powersoft Power Builder
Type	DSS	DSS	DSS	EIS	Application Development	Application Development
API	Proprietary	Proprietary	Allbase/PC	Allbase/PC	Allbase/PC	Allbase/PC
Connection Type	Serial/LAN	Serial/LAN	LAN	LAN	LAN	LAN
Operating System	Windows	DOS, Windows, NewWave	Windows	Windows	Windows	Windows
Learning Curve	Hours	Hours	Hours	Days	Weeks	Weeks
SQL Knowledge Needed?	No	No	No	Some	Yes	Yes
Phone	(800) 426-4667	(800) 752-0900	(800) 876-3101	(800) 289-0053	(800) 876-3267	(800) 395-3525
Fax	(613) 738-0002	(408) 447-0264	(212) 967-6406	(603) 427-0385	(415) 321-5471	(617) 273-2540
Complementary Products	PowerHouse Windows		Q+E Database Library	InfoPump		

TABLE 1

CURRENT TOOLS OFFERINGS

In addition to all the PC client/server tools mentioned above, a rich assortment of direct-access 4GL tools exist for accessing HP's relational databases. This prior sentence specifically indicates databases in the plural. All the tools in this section work with both ALLBASE/SQL and IMAGE/SQL. The important point about these tools is that they execute on the host, not the client/server cooperative execution of the previous set of tools.

The products in this section have, for the most part, been around for some time. Details will not be presented here about features and benefits. The main point here is that several vendors offer SQL-based tools for accessing relational databases.

Some of these 4GL tools offer native (intrinsic) access to IMAGE. However, the relational access they offer ALLBASE/SQL now extends to IMAGE/SQL. Below is a list of products and the companies which offer them:

ALLBASE Toolset
Transact
Powerhouse
Focus
JAM
Speedware
Uniface

Hewlett-Packard
Hewlett-Packard
Cognos
Information Builders
JYACC
Speedware
Uniface

As you can see, these tools, along with those indicated for PC client/server, make an impressive arsenal in your development efforts.

Two relational databases from HP can reside on your system, ALLBASE/SQL and IMAGE/SQL, and all these tools can access each. Most of what this paper describes is what IMAGE/SQL offers you. Some explanation of what ALLBASE/SQL offers follows.

WHEN TO USE HP ALLBASE/SQL

HP offers two database management systems on the HP 3000 platform. Each has its place with any given application, and many applications could use either. The majority of database usage on the HP 3000 is currently IMAGE, but there are certain applications where ALLBASE/SQL is the better choice.

ALLBASE/SQL is a full-featured relational database management system (RDBMS) with functional compliance of ANSI standards. HP sees four specific areas where ALLBASE is the preferred DBMS:

- Mainframe class computing,
- Distributed applications,
- High-volume online transaction processing,
- Object-oriented applications.

Below are expansions of these topics.

HP has positioned ALLBASE as the alternative to mainframe-class database management systems (DBMS). This works in conjunction with HP's mainframe downsizing strategy which offers high-end HP 3000 systems as Corporate Business Systems. Providing online backup and restructuring, and supporting very large file sizes, ALLBASE is the DBMS of choice in this area.

Distributed information and the applications which support them are also addressed within the ANSI standards for relational database systems. Here, again,

ALLBASE has been specifically featured. Using two-phase commit protocols, distributed transactions can be ensured of completion and accuracy.

ALLBASE also supports Encina technology from TRANSARC Corporation. This technology provides a standardized method of transparent distributed transaction processing in a multi-platform environment. These distributed application features make ALLBASE the clear choice compared to IMAGE/SQL.

Where high-volume online transaction processing (OLTP) is the objective, and relational concepts are required, ALLBASE is again the best choice. This pertains mostly to new application development. Since ALLBASE has been designed from the ground up as a high-end relational database, applications requiring high-volume SQL OLTP will benefit.

Object-oriented computing is being seen more and more as a viable solution in many applications. Most assuredly, any application which has multimedia requirements can benefit from object technology. Here ALLBASE/SQL has the advantage. Object storage, especially that of video, photograph, audio, et cetera, requires data structures foreign to IMAGE/SQL. ALLBASE/SQL already provides the necessary storage with Binary Large Objects (BLOBs).

From these descriptions you can see that there are specific situations for ALLBASE/SQL.

COEXISTENCE

The HP 3000 now offers a complete range of data management choices. Native IMAGE still provides the highest performance in the industry for mission-critical OLTP business applications. IMAGE/SQL provides data access through the multitude of 4GL and PC client/server toolsets at a small performance premium. Client/server computing is clearly the emerging trend in information processing, and IMAGE/SQL is now an important element, ensuring the protection of your information investment.

ALLBASE/SQL provides the highest SQL performance of any relational database in the industry and provides support for distributed database and distributed transaction processing. With Corporate Business Systems, ALLBASE/SQL can handle the requirements of nearly any enterprise at a fraction of the cost of traditional mainframe solutions.

Together, these database management systems make a powerful team. And they work together. Concurrent access is possible by linking the two together within a single environment. Your applications, whether host-based or client/server, can

An Overview of HP IMAGE/SQL

simultaneously access ALLBASE/SQL and IMAGE/SQL information. That is coexistence at its best.

But there is more.



THE RELATIONAL PICTURE

HP offers you two relational databases, IMAGE/SQL and ALLBASE/SQL. But the story gets better. Also available on the HP 3000 are Oracle and Ingres, and each of these RDBMS products have their links to IMAGE as well, Oracle through their OracleConnect product, Ingres through their ALLBASE Gateway to IMAGE/SQL. This creates a SQL shell over every database management system available on the HP 3000. And with this shell comes all the 4GL and client/server tools which have been mentioned throughout this paper. This is a very powerful offering, indeed.

Taking a step farther, HP has proposed several object-oriented functions for addition to the SQL standard, the objective of which is the defining of Object SQL (OSQL). Should these proposals be accepted, object technology will be a step closer to IMAGE/SQL.

This last section gives a picture of where we are today and a glimpse of what the future might hold. Let's step back and review what has taken place in the last several years to get us where we are today.

HP DATABASE DEVELOPMENTS

By looking at just the past couple of years, it's easy to see that HP is actively working to make their database products superior. In 1992, we released the following new features:

IMAGE

- o Third-party indexing interface
- o Critical item update
- o 4GB file sizes
- o Corporate Business System tuning

ALLBASE/SQL

- o Database shadowing
- o Record level locking
- o Remote unattended backup
- o Stored procedures

An Overview of HP IMAGE/SQL
3002-9

- o Business rules and triggers
- o Two-phase commit (via XA interface)
- o Additional third-party tools

By far the biggest announcement, occurring late in 1993, was the introduction of IMAGE/SQL. Release of this product provided the following features to IMAGE:

IMAGE/SQL

- o Relational access via SQL
- o Stored procedures
- o Business rules and triggers
- o SQL PC API support

The year 1994 will not be without its share of enhancements, either. Announced for release in 1994 are:

IMAGE/SQL

- o Dynamic detail set expansion
- o Increased software resiliency
- o Performance tuning
- o Predicate-level locking (SQL access)
- o Third-party index aware (SQL access)
- o Native mode Query

ALLBASE/SQL

- o Fast recovery from media failures
- o Roll forward of physical file DBA operations
- o 4Gb log files
- o Dynamic setting of isolation level and transaction priority
- o Parallel pre-fetch for faster table scans
- o Access path modification for greater control
- o Ongoing multiprocessor scaling
- o SQL monitor tool for performance analysis/troubleshooting
- o Faster load algorithm
- o Parallel loads
- o Truncate table (empty table quickly)
- o SQL audit tools for viewing DBA changes
- o Case insensitivity option

IMAGE/SQL & ALLBASE/SQL

- o ANSI'92 entry level compliance (except AS' Clause)
- o Additional third-party tools
- o ODBC support
- o Improved pre-processor concurrency
- o Multi-row stored procedure result sets
- o Extract command for module movement
- o Application thread support

This impressive list of delivered and promised features is a clear demonstration of HP's dedication to providing the best data management systems on HP computer systems. But for all the features, if the product is not easy to use, the product will not be used. Let's next explore just how simple it is to use the new IMAGE/SQL.

ELEGANT SIMPLICITY

Accessing IMAGE relationally is as simple as 1-2-3.

- Step 1: create the environment;
- Step 2: attach IMAGE to the environment;
- Step 3: access IMAGE relationally.

Each of these steps will be briefly described in this section.

Step 1: Create the SQL Database Environment

This step involves the creation of the SQL Database Environment (DBE). This structure contains control information about the data represented by it. This environment is created using the ISQL utility as follows:

```
:ISQL  
ISQL => START DBE 'SQLDBE' NEW;
```

If a DBE already exists, this step can be omitted.

Step 2: Attach the IMAGE Database to the DBE

The attachment process examines the IMAGE database and places equivalent SQL-structure information in the DBE. This step can include a variety of mapping functions for fine tuning relational access. This step uses the IMAGESQL utility.

```
:IMAGESQL  
> SET TURBODDB CUSTDB  
> SET SQLDBE SQLDBE  
> ATTACH
```

The assumption in this case is that CUSTDB is an existing IMAGE database.

Step 3: Access IMAGE Data Relationally

Any number of methods could be employed in this step to demonstrate this access. In this case, ISQL is used.

```
:ISQL  
ISQL => CONNECT TO 'SQLDBE';  
ISQL => SELECT * FROM CUSTDB.CUSTOMERS;
```

The result of this query would be a tabled list of all entries in the CUSTOMERS data set. You can also display information about the database itself.

```
ISQL => SELECT NAME, OWNER FROM SYSTEM.TABLE;
```

This would return all table names associated with the SQLDBE environment.

Updating IMAGE databases is also very simple. Here is another ISQL example which updates a column (item) called PRODUCT_NAME.

```
:ISQL  
ISQL => CONNECT TO 'SQLDBE';  
ISQL => UPDATE CUSTDB.ORDERS  
          SET PRODUCT_NAME = 'IMAGE/SQL'  
          WHERE PRODUCT_NAME = 'IMAGE';
```

This update finds all entries where PRODUCT_NAME is 'IMAGE', then changes that value to 'IMAGE/SQL'.

As these steps demonstrate, accessing IMAGE data relationally is very simple and straight-forward. Once the database is attached to the DBE, nothing else is required in the regular use of the system. There are additional administrative tasks related to security and data type mapping, but none of these are overly complex or cumbersome.

Next, let's discuss the simplicity of some the client/server tools.

Forest & Trees by Trinzic

If you recall, Forest & Trees (F&T) is a PC client/server tool falling under the category of EIS. This means this tool reads, reports and formats data from the IMAGE/SQL database (or any of the other host- or PC-based databases).

Quoting from the Reference Guide: "Forest & Trees... collects and combines data from a variety of sources and monitors the resulting information in order to track information at all levels from business vital signs to underlying detail.

"To help you display, integrate, and use the collected information, F&T has a large set of data manipulation and object control functions. These functions can be used in formulas, queries, triggers, graphs, and reports. "Forest & Trees can also be fully customized with pictures, buttons, menus, and other graphic attributes to create distinctive applications."

Although the details of this tool are too extensive for this paper, some details will be discussed to demonstrate this tool's ease-of-use.

F&T has a tool bar at the top of the window which is filled with object icons for ease in application control and design. Views of data are defined by the user and extracted on demand or when scheduled. These views can be at any level, from high-level summary to low-level details. The drill-down feature of this tool allows display of all levels of information from top to bottom. Graphic views of the various levels of information can also be created and displayed.

One of the outstanding features of this tool is that user-defined limits or ranges can be set to monitor selected items, formula results, and sums. Color coded warnings, green for within limits, yellow for warnings, and red for outside limits, can highlight monitored data. Exception management is much simpler when the software isolates and highlights the exceptions.

PowerBuilder by Powersoft

Another PC client/server tool is PowerBuilder by Powersoft. This tool is from the Application Development category presented earlier and is both a reporting and transaction processing tool.

Quoting from a Powersoft sales brochure, "PowerBuilder... is a comprehensive Microsoft Windows-based development environment for constructing graphical client/server database applications through object-oriented development.

"PowerBuilder supports the rapid design and development of Windows-based client/server relational database applications for all marketplaces.

"PowerBuilder... contains full support for the MS Windows GUI technology, such as radio buttons, command buttons, list boxes, bit maps, etc., [and] supports custom user objects that can be defined as standard Windows controls."

As indicated, PowerBuilder is a very full featured application development tool. As mentioned above for Forest & Trees, this paper is not intended to be a detailed discussion of this products features. Instead, to summarize, this product offers significant productivity gains through development and use of its object technology orientation.

CONCLUSION

HP IMAGE/SQL is the most significant enhancement of the IMAGE DBMS of all time. The opportunity which this announcement provides to commercial and company developers is enormous considering the selection of tools now available for use with IMAGE. IMAGE/SQL also solidifies its coexistence with HP's other relational database, ALLBASE/SQL, with standard access to both.

The simplicity with which relational access occurs with IMAGE/SQL makes this perhaps the most painless 'conversion' in the history of the computer industry. The term 'coexistence' applies equally well to the dual accesses to IMAGE, native and SQL, as it does to the two relational databases, IMAGE/SQL and ALLBASE/SQL.

For all that this paper presents, the most important point is this: With IMAGE/SQL, HP has once again protected your investment in the HP 3000!

Paper #3003
Intro to IMAGE/SQL
Denys P. Beauchemin
HICOMP America, Inc.
P.O. Box 22758
Houston, TX 77227-2758
USA
(713) 626-3842

roduction

1992 Hewlett-Packard introduced an SQL interface for IMAGE, under the name SQL for IMAGE. A few months later, HP renamed it to IMAGE/SQL and today, it is one of the hottest topics of the HP3000. This paper will address all aspects of IMAGE/SQL. We will cover the latest developments in IMAGE, attach a TurboIMAGE database to an ALLBASE/SQL database environment (DBE) and use SQL to retrieve data in the IMAGE database..

ALLBASE and TurboIMAGE

We can look at IMAGE/SQL as the end product of marrying two existing technologies, to wit TurboIMAGE and ALLBASE. We are all quite familiar with the IMAGE side, so let's spend a very short amount of time on the ALLBASE aspect.

ALLBASE	TurboIMAGE
Concept	equivalent
DBE	Collection of databases
Objects of one OWNER	DATABASE
TABLE	DATASET
ROW or TUPLE	IMAGE RECORD
COLUMN	FIELD or ITEM
INDEX	CHAIN or KEY
GROUP/USERS	PASSWORDS

Figure 1. Database Objects.

DBE (Data Base Environment) is made up of various MPE files on the system and is created with an SQL command issued programmatically or through ISQL.

The command, *START DBE* allows the creator (also known as the Data Base Administrator DBA by default, though more users can be granted DBA capabilities later) to specify the name of the DBECON, DBEFILES and DBELOG files. The DBECON file's name is also the name of the DBE and this file contains the configuration of the entire DBE. One should view the DBEFILES simply as repositories of data. The DBA can add more space to the repository by simply adding more DBEFILES to the existing DBE. This is done via other SQL commands. Finally, the DBELOG files are used for, logging... Of course, all these files are privileged files on MPE/iX.

The DBE in effect is a collection of databases which share the same logging and recovery mechanism.

DATABASE is a collection of tables which belong to the same OWNER. More on this a little later.

TABLE contains the data, like a stand alone detail dataset.

ROW or TUPLE is a collection of columns within a table.

A *COLUMN* is a single storage entity within a *ROW*.

An *INDEX* is the method used by SQL to streamline access to the data contained in various tables. When *IMAGE* makes use of *KEYs* for masters and *CHAINS* for details, *ALLBASE/SQL* supports b-tree access to all tables and a form of *HASHED INDEX* to specifically designated tables.

The *GROUP/USER* concept is used by SQL to control which user can perform which action to what data. A *GROUP* is made up of *USERS* and other *GROUPs*. *USERS* and *GROUPs* can be *GRANTED* various capabilities by the *DBA*. These same capabilities can be *REMOVED*. A

The mission is therefore to map the *IMAGE* database into the *ALLBASE DBE* and away we go.

INTERLUDE

Before going any further, it is important to realize a few things about *ALLBASE/SQL* and to keep in mind a few rules for this exercise.

ALLBASE uses itself to store the directory of tables, indexes, columns, groups, and other objects. Entries regarding these objects are stored in a database which is made up of tables belonging to a user called *SYSTEM*. Thus if one wants to see which tables are in a *DBE*, one could retrieve this information from a table called *SYSTEM.TABLE*.

Only one *DBE* can be opened by a session at any one time. Therefore we should plan on *ATTACHING* all related *IMAGE* databases to the same *DBE*. In our example, we will create the *SALESDBE* and we will *ATTACH* two *IMAGE* databases: *ORDERS* and *INVTRY*. With release *F.0* of *ALLBASE*, a process can connect to up to 32 *DBEs*, but a transaction must still be limited to a single *DBE*. The command *SET CONNECTION TO 'dbe'*; is used to switch the connection to the various *DBEs* during the process.

The concept of passwords does not exist in *ALLBASE/SQL*. The user name is used at *CONNECT* time to acquire the capabilities in the *DBE*.

IMAGE/SQL follows all *MPE* restrictions so let us be aware of the various database, *DBEs*, groups and accounts. In our example, the *DBE* and the *IMAGE* databases are in the same group and account and we are the creator for all of them.

IMAGESQL will prompt for a maintenance word for both the *DBE* and the *IMAGE* database if the user is not the creator. So make sure that the maintenance words are set properly for both.

CREATING A DBE

The first thing to do is to create a *DBE*, in this case we will call it *SALESDBE*. We use the program *ISQL* (*Interactive SQL*) to issue the following *SQL* command:

```
:ISQL
isql=>START DBE
  'SALESDBE' MULTI NEW           (1)
  DBEFILE0 DBEFILE DBEFILE0     (2)
  WITH PAGES = 1000, NAME = 'SAESD0',
  LOG DBEFILE DBELOG0           (3)
  WITH PAGES = 1000, NAME = 'SAESLG0';
isql=>EXIT;
```

Figure 2. Creating a DBE.

(1) Here we specify the name of the *DBE*, the fact that it is new and the fact that it can be accessed by multiple users.

Here we specify the size and the names (internal and external) of the original DBEFILE DBEFILE0. The size is in pages which represent 4k bytes each. Here we request that the DBEFILE0 be called DBEFILE0 internally, that it be allotted 1000 pages of 4k and that it be created as MPE file SALESDB0.

Here we specify the type, name (internal and external) and size of the log file. In our example with are using DUAL logging and the logging will be circular and to the MPE file SALESLOG0. Note the ; to end the command.

Use SQLUTIL to switch settings and assign a maintenance word.

```
:SQLUTIL
>>SETDBEMAINT
DBEnvironment name: SALESDBE
Current maintenance word: <cr>
New maintenance word : mword
Retype New maintenance word: mword
>>EXIT
```

Figure 3. Using SQLUTIL.

ATTACHING AN IMAGE DATABASE

Once we have created the DBE, we want to attach the two IMAGE databases to it. For this we will use a program called ATCUTIL or IMAGESQL. By the way, ATCUTIL will create a DBE for you, if you wish.

```
:IMAGESQL
>> SET SQLDBE SALESDBE
>> SET TURBODB ORDERS
>> ATTACH
various warnings and voila!
>> EXIT
```

Figure 4. Attaching a database.

What just happened? Well, many things, let us examine closer.

Each dataset in the IMAGE database has been added to SYSTEM.TABLE with the name of the IMAGE database as the owner. The default owner can be altered at ATTACH time by specifying OWNER=. However, remembering that in ALLBASE parlance a database is a collection of tables belonging to one owner, IMAGE/SQL fits very well within the concept.

Each field in each dataset has been added to SYSTEM.COLUMN with the NOT NULL attribute as IMAGE currently does not allow valid null items.

A file called ATCINFO is created which contains mapping information for each dataset in the IMAGE base. This ATCINFO file is created as a privileged file which accompanies the DBE, in essence, it is now part of the DBE and is declared in the DBECON file. Mapping involves transforming data items or dataset names, data types and security features. Specifically if item or set names contain special characters, IMAGESQL will transform these characters to an underscore. For example, IMAGE's CUSTOMER-MASTER will become CUSTOMER_MASTER, and MARKUP% will become MARKUP_. There are more things which will be added to this file in a few minutes.

The DBE can be attached to multiple IMAGE databases, so the ATCINFO file will contain information about each and every attached IMAGE base.

A file called dbnameTC is created which is a privileged file that accompanies the TurboIMAGE database. Since an IMAGE database can be attached to multiple DBEs, the accompanying DBTC file will contain entries for each DBE to which this IMAGE base is attached.

MAPPING DATA TYPES

If a data field in any of the datasets of the IMAGE base is a compound item, a message about compound items being SPLIT will be displayed at ATTACH time. This is because ALLBASE/SQL does not support compound items. What actually happened at ATTACH time, is that IMAGE/SQL generated (re-mapped) a series of columns which correspond to the subitem count of the IMAGE item. For example, the IMAGE item ADDRESS is defined as 4X30, IMAGE/SQL will map this item into 4 columns like so:

IMAGE item	IMAGE/SQL MAPPED column
ADDRESS 4X30	ADDRESS_1 char(30)
	ADDRESS_2 char(30)
	ADDRESS_3 char(30)
	ADDRESS_4 char(30)

Figure 5. Splitting compound items.

But data type mapping goes further than splitting compound items into multiple columns. IMAGE/SQL allows the DBA to map a non-compound item into multiple columns. IMAGE/SQL allows data type mapping so that the data contained in the item can be handled as a type differing from the IMAGE type. Let us first look at user mapping.

There are many databases which have been designed with a single item, usually rather large, that actually (logically) encompasses many fields. This is prevalent in databases which came into being before TurboIMAGE. In pre-Turbo days, an IMAGE base could only have 255 distinct data items declared for use throughout the database. This presented problems for larger, more complex databases. The common solution was to declare a large item, like x200 or j50 and have the program actually interpret the data correctly. For users of 4GLs, this was a neat trick, as the various dictionaries handled these items with no problems. For example, an item defined as x200 could actually contain various integers and strings which add up to 200 bytes. In this case, the item is just used as a storage space and IMAGE as a storage manager with non-representative data items.

However, this will not work for IMAGE/SQL in the SQL aspect. SQL has something known as data integrity constraint. This means that you cannot have the same X200 as above, to contain a mix of integers and characters and other types. SQL, when manipulating this column, will be very unhappy finding data which is not a valid character string.

So IMAGESQL enables the DBA to create new columns which map onto this X200 field and actually represent the data properly. This mapping is done via the SPLIT command in IMAGESQL. It is important to know precisely what is contained in the sub-fields and the length of each sub-field.

Let's use an example: In our x200 field above, which we will call CUST-INFO, contains the customer name, 4 address lines, country, telephone number and related customer number. The split command would be:

```

SPLIT      CUST_MASTER.CUST_INFO INTO
           CUST_NAME:X30 :CHAR(30),&
           C_ADDR1:X30 :CHAR(30),&
           C_ADDR2:X30 :CHAR(30),&
           C_ADDR3:X30 :CHAR(30),&
           C_ADDR4:X30 :CHAR(30),&
           C_COUNTRY:X30 :CHAR(30),&
           C_TELEPHONE:X16 :CHAR(16),&
           C_RELATED:I2 :INTEGER

```

Figure 6. Using the SPLIT command

ATCINFO file would then be updated to contain this mapping and the newly specified columns would also be inserted in SYSTEM.COLUMN.S. These new columns would now be available for SQL access.

The last important concept of data type mapping has to do with the data type itself. IMAGE, while having several data types, does not have data constraint. Which means that the data is not checked for overflow going in or out of IMAGE. In fact, the only common tool from HP which makes use of a data integrity constraint concept is QUERY. For example, QUERY will not allow the user to enter a non-numeric character when it is asking for customer-no. QUERY verifies that the input is valid for the responding IMAGE data type.

SQL on the other hand, while having fewer and sometimes different data types, has strong data integrity constraints. This means the data must fit the data type or something untoward is going to happen at some point. During the original ATTACH, IMAGE/SQL will map the IMAGE data types to equivalent SQL types. But as is usual in cases of conversion, it is not 100%.

SQL has SMALLINT, INT, DECIMAL and FLOAT data types to which IMAGESQL can map the following types: LOGICAL, REAL, ZONED and PACKED data. It is outside of the scope of this presentation to go into data mapping at this time, suffice it to say that IMAGESQL allows for alternate types in cases of problems. By using the UPDATE command, the various columns can have their data types changed to alternates which can better contain the data.

The mapping is elementary for X and U fields into CHAR of corresponding length and I1 and I2 types map into SMALLINT and INT respectively.

Again, all this mapping information is carried in the ATCINFO file and is of course, contained in SYSTEM.COLUMN for each column.

It is recommended to perform the data mapping task before adding users (next step), since all views relating to the altered data are dropped if the data type is modified or the data is split.

ADDING USERS

After the ATTACH command, only one user is defined in IMAGE/SQL, the database creator or DBC. In order to allow other users, the DBC, having DBA capability in the DBE, has to declare additional users.

As we said earlier, the concept of passwords is not implemented in SQL. Instead, it has USERS and GROUPS with capabilities. For our purposes, the USER is the MPE USER and ACCOUNT of the user connecting to the database. For example if I had logged on as MGR.DENYS my CONNECT ID to IMAGE/SQL would be MGR@DENYS.

In IMAGE, the access to the data is decided at DBOPEN time, using a combination of the password and the DBOPEN mode. The password used, if valid, corresponds to a user class number between 1 and 63. Class 64 is reserved for the creator and class 0 is used for any invalid or nonexistent password. In the

schema, the creator will have specified the READ and WRITE accesses to data items and datasets, specifying which classes can do what action to what data.

For IMAGE/SQL, we need to map USERS to MODE/PASSWORD combinations. This is done as declare new users to IMAGE/SQL using the ADD USER command:

```
>> ADD USER MGR@DENYS WITH PASS=PW, MODE=5
```

Figure 7. Adding a user.

Now we get various messages.

IMAGE/SQL will equivalence the password SUPERPW to a user class, for example 63 and will create GROUP to represent this class. The GROUP name is made up of the IMAGE database name (the OWNER, remember?) and the user class. In our example this would be ORDERS_V63. The USER is added to the GROUP and then IMAGE/SQL creates a series of VIEWS which reflects the access that the MODE/PASSWORD combination allows. There is one VIEW created per each dataset which is accessible. The VIEW is named with the owner, the dataset name and the class number. Within the view, the accessible columns are represented.

In our example, if class 63 permitted access to the dataset CUSTOMER-MASTER, the VIEW ORDERS.CUSTOMER_MASTER_V63 would be created and it would contain the names of the columns to which class 63 had access.

A VIEW is nothing more than a stored SELECT command.

ACCESSING IMAGE/SQL

Now that the preparations are done, let us access the IMAGE data via the SQL environment.

There are three ways to do this. One, write a program, two, using any of the neat 3rd party products, and finally, via ISQL. For the purposes of our discussion, we will limit ourselves to ISQL.

Actually we have already used ISQL when we created the DBE originally. This shows us that ISQL is more than just a QUERY equivalent, in fact, ISQL, which stands for Interactive SQL, allows one to use ALL SQL commands.

For IMAGE/SQL tables, however, these commands are limited to the commands of the Data Manipulation Language (DML) portion of SQL. IMAGE/SQL does not support the Data Definition Language (DDL) commands. The core of the DML is the SELECT command:

```
select-statement ::=
SELECT [ ALL | DISTINCT] select-list
INTO host-variable-reference, [,host-variable reference]...
FROM table-reference [,table-reference]...
[WHERE search-condition]
[GROUP BY column-name [,column-name]...]
[HAVING selection-condition]
[ORDER BY column-name | column-number [ASC|DESC]]
```

and don't forget the ';' at the end of the command.

SELECT Figure 8. The command.

SELECT-LIST: By using DISTINCT, the user prevents the retrieval of duplicate rows. If the user wishes to retrieve all the columns, the asterisk can be used. Also, aggregate functions can be specified here, as well as expressions and constants. Aggregate functions are: MIN, MAX, AVG, COUNT and SUM.

ession is simply a mathematical expression enclosed in parentheses, eg: (1.10 * SALE_PRICE) would
n all sales_price marked up by 10%. This is done by SQL not ISQL.

permits the user to directly issue a SELECT command. Since ISQL is a program itself, the user
not specify the INTO clause. Programmatically, the host-variables are the storage areas in the user
gram to which SQL will return the retrieved data.

JM: This instructs SQL as to which tables are to be retrieved from. Multiple joins are easily executed,
ply by specifying multiple tables. Note that if in the select list, there are duplicate column names due
multiple tables, these columns must be fully qualified by adding the table name in front of the column
te.

ERE: This is how one controls the selection of rows during a retrieval. One or more search condition
be specified thus giving greater control over the retrieval. Of course, using KEY items and SEARCH
is in the where clause accelerates the retrieval of the rows. If the where clause is not specified, or the
ction criteria are not KEY or SEARCH items, the datasets will be scanned serially.

GROUP BY: Using this clause, one identifies columns which are used for grouping for aggregate
ctions applied to a group of rows.

HAVING: This is how one further controls the retrieval of groups defined by the GROUP BY clause. The
GROUP BY has to be specified in order to use this feature.

ORDER BY: This clause is used to sort the retrieved rows by the specified column or columns in either
ending or descending order. The columns can be specified by column name or column number in the
ect list.

he WHERE is used, it takes precedence over the GROUP BY, HAVING and ORDER BY. The rows
selected before the groups are formed. If the GROUP BY clause is used, SQL creates groups out of the
rieved rows. If the HAVING clause is used, the groups which do not qualify will be eliminated.

he selection criteria are many and quite powerful, but one must pay attention lest a large select be sent
properly specified, against a large dataset. SQL supports partial keys, between, comparisons, exists,
, nulls, quantifieds.

NAME EXAMPLES

Retrieve all customer numbers and name of customers in Texas, sort them by customer name in
ending order:

```
SELECT customer_no, customer_name
FROM   dbcust.cust_master
WHERE  state = 'TX'
ORDER BY customer_name;
```

Figure 9. Example 1 - A simple select.

Retrieve all customer numbers, customer names and year-to-date orders, group them by state
d total up the sales by state:

```
SELECT customer_no, customer_name, ytd_sales, SUM(ytd_sales)
FROM   dbcust.cust_master
GROUP BY state
ORDER BY customer_name;
```

Figure 10. Example 2 - Using GROUP BY in a select.

3- Now we are on a roll, let's retrieve the outstanding orders from the orders file and select only the customers which have orders outstanding for over \$10,000. Further let's group the orders by customer and get the customer name as well.

```
SELECT      cust_master.customer_no,customer_name,total_value, SUM(total_value)
FROM        dbcust.cust_master,dbcust.order_header
WHERE       cust_master.customer_no=order_header.customer_no, order_status ='O'
GROUP BY   cust_master.customer_no
HAVING     SUM(total_value) > 10000
ORDER BY   cust_master.customer_name;
```

Figure 11. Example 3 - A complex SELECT.

CONCLUSION

We have seen that it is quite straightforward to make a TurboIMAGE database into an IMAGE/SQL database. Only in trying it out, and experiencing for one's self will one be able to get a glimpse at what is now possible with this new environment.

The 3rd parties, now able to access IMAGE through its OPEN SYSTEMS interface, are going to supply. There are now, client server-based tools which make decision-support tasks a joy. There are going to be many more tools, and opportunities. The winners are the users.

IMAGE/SQL ushers in OPEN SYSTEMS for IMAGE users. Whilst not a Relational Data Base Management System (RDBMS) in the true sense, IMAGE/SQL enables relational-type access to IMAGE data, and for virtually all users this will be more than sufficient. For now...

Paper # 3004

**OK it's installed... Now What?
Image/SQL Future**

**Liz Norman
Hewlett-Packard
100 Mayfield Ave
Mountain View, Ca
94043**

You've seen the demos at the trade shows or from HP and HP even sent you the software for this new product they call "Image/SQL" so you decide well I might as well install it. Now that you have it installed you're wondering why you did install this product.

One reason you may have installed Image/SQL is because at some conference a vendor showed your users their wonderful tool, how easy it is to use and what power they will have when using their tools. Users will no longer be waiting on the programmer for an ad-hoc report because he can now do it himself... Ya right!

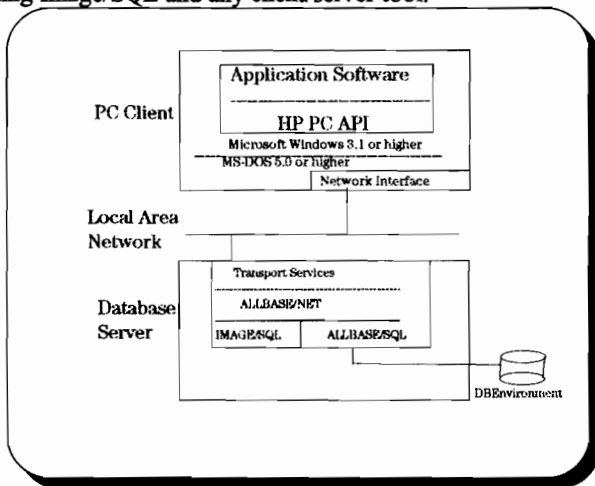
The problem I've been finding is that the Information Systems (IS) staff has been receiving information on Image/SQL and the client server tools but most of the problems many people are facing are what I call the "Middle Steps" problems which until now are not really being addressed by anyone. This paper is designed to help you bridge the gap between the HP3000/Image/SQL side of the equation and the lovely client server tools on the user side of the equation.

This paper will address the following middle steps:

- 1) What's included in a client/server environment
- 2) What's required on the server side
- 3) What's required on the client side
- 4) How do I start the software
- 5) Preparing the client
- 6) Client tools
- 7) Troubleshooting: What to do when it doesn't work!!

1) What's included in a client/server environment

Let's start at the beginning. Below is a picture of the client/server environment using Image/SQL and any client/server tool:



In this diagram, at least two parts should be familiar- the Application Software and the Image/SQL. The application software are those great client/server tools and the Image/SQL is what you installed. The rest of the picture includes some elements you may or may not have yet.

Let's first talk about what you will get with Image/SQL:

- 1) Image/SQL product
- 2) Restricted version of Allbase/SQL unless you want to purchase the full package.
- 3) Allbase/Net product- this is the server networking interface to Allbase/SQL
- 4) HP PC API- PC Application Programming Interface

2) What's required on the server side

Hardware you will need on the server side includes:

HP3000 Series 900

ThinLan 3000/iX Network Link

Data communications software: One of the following, depending on the protocol you are using.

TCP/IP protocol- HP ThinLan 3000/iX

SPX/IPX protocol- HP ThinLan 3000/iX and Netware for the HP3000.

3) What's required on the client side

On the client side you will need to supply the following to have a client/server environment:

Hardware

- 1) Intel-based 80386 or 80486 PC with graphics capability
- 2) Mouse
- 3) LAN Card
- 4) Minimum of 4 MBytes of extended memory; 8 MBytes is recommended
- 5) Minimum of 2.5 MBytes disk storage

Software

- 1) MS-DOS 5.0 or higher
- 2) Microsoft Windows 3.1 or higher, standard, or enhanced mode.
- 3) Application Software (Client server tool)
- 4) Networking software.

4) How do I start the software

The first step is to ensure you have all of the required hardware and software for both the client and the server. The next step is to ensure you have the correct versions of these software packages. Unfortunately this is not a living document so I will list what products are current as of this writing, 6/23/94.

We need to test the versions of the server software:

- 1) Allbase/SQL
:SQLVER.PUB.SYS.

This program will give you the current versions for Allbase/SQL. The version should be A.F0.00 or higher. Version A.G0.20 will be released with the 5.0 version of the MPE operating system.

2) Image/SQL

:IMAGESQL.PUB.SYS

This program will give you the version number which should be HP36385B B.F0.27 or higher.

After we are sure the products on the server side are current we must ensure that the Allbase/Net listener is waiting for a request. To start the listener follow these steps:

- 1) Check that the spooler is started and STREAMS is enabled. These commands should be in the SYSSTART file.
- 2) To start the listener, use the ANSTART command. You must have NM (Node Manager) capability to do these tasks. This command streams a job which runs the listener. The listener job is streamed with the :HIPRI option to ensure a timely logon. After the listener program is started it will run in the B queue at priority 149.

Syntax:

ANSTART {Network Interface} [DEBUG]

ANSTOP {Network Interface}

ANSTAT {Network Interface}

Ensure that the server network software is up and running. If you are using NS3000 then issue the following command:

NSCONTROL START- to start the networking service

NSCONTROL STAT- to check the networking service

NSCONTROL STOP - to stop the networking service

If you are using Netware then you will use the following commands:

NWSTART- to start the networking service

NWSTAT- to check the networking service

NWSTOP- to stop the networking service

5) Preparing the client

1) Install the application software

2) Check the networking configuration

If you are using ARPA services, then check the network configuration as follows:

1) Type netsetup at the DOS prompt

2) Select Reconfigure network software

3) Check that NetIPC or sockets are configured, depending on the one you are using.

4) Change the Socket Buffer size to 12800 or greater.

5) Set demand load=NO

6) If you are using HP Sockets ensure that the number of Socket Sessions is greater than or equal to the number of concurrent connections that you expect to use.

7) If you are using NetIPC (NS), ensure that the number of NetIPC Sessions is greater than or equal to the number of concurrent connections that you expect to use.

8) Make sure that the PC network library is in your path.

If you are using TCP/IP then after you are sure your network is setup correctly issue the following command:

```
c:\PING 15.24.190.200
```

where PING is a command and the numeric is the network IP address of the server.

If PING responds with a message stating that the connection was allowed then you know your LAN is hardware and lower level software ok.

The HP PC API comes with the Image/SQL files and is stored on the HP3000 in a self-extracting file. The file name is HPPCAPI.PUB.SYS. You will need to download the information to the PC then extract the files on your PC client.

Netware/iX

1) F:\LOGIN> login supervisor

2) Establish a directory for the PC API files:

```
F:\SYSTEM> cd \
```

```
F:\>mkdir HPPCAPI
```

3) Logon to your server as MGR.NETWARE and execute the following commands, replacing server with the name of your HP3000 server, to copy the PC API self-extracting file into your Net Ware file directory.

```
nwcopy.support
```

```
login server:supervisor
```

```
copy hppcapi.pub.sys;\hppcapi\hppcapi.exe
```

```
EXIT
```

ARPA Services (FTP)

From your PC, do the following:

1) Create a directory called PCAPI

2) Change directory to PCAPI

```
cd PCAPI
```

3) Copy the self-extracting file to your PC with the following commands, replacing server with the name of you HP3000 server, and use MANAGER.SYS,PUB as the User for the FTP prompt:

```
ftp server
```

```
binary
```

```
get hppcapi hppcapi.exe
quit
```

NS Services

From your PC, do the following:

- 1) Create a directory called PCAPI
- 2) Change directory to PCAPI
cd PCAPI
- 3) Copy the self-extracting file to your PC with the following commands, replacing server with the name of you HP3000 server:

```
dscopy -r -F -B -L256 server#manager.sys,pub#hppcapi hppcapi.exe
```

Extract the PC API file

```
c:\PCAPI> hppcapi
```

Two file should result: hpodbc.exe and hpgupta.exe.

If you are using ODBC then do the following:

- 1) Extract the ODBC files from the self-extracting file
C:\PCAPI>mkdir ODBC
C:\PCAPI>cd ODBC
C:\PCAPI\ODBC>..\hpodbc
- 2) Run Microsoft windows
- 3) Ensure all applications except the Program Manager are closed.
- 4) From the File menu, click on Run.
- 5) Execute setup to install the PC API files. Use the path where your PC API files are located.
c:\pcapi\odbc\setup
- 6) Press continue to get the Install Drivers dialog box.
- 7) From the Install Drivers dialog box, select the HP ALLBASE/SQL driver by clicking on it and then click on OK to install the files. All files will be copied to \WINDOWS\SYSTEM directory and the ODBC Administrator will be installed in the Control Panel with the icon name of ODBE. The ODBC Administrator is used to add, modify, or delete Data Source Names (DSN).
- 8) From the Data Sources dialog box you can configure Data Source Names. Click on Setup
- 9) From the HP ALLBASE/SQL ODBC Setup dialog box, type in the required data then click on OK.
- 10) From the Data sources dialog box, click on Close.

You can add, modify or delete Data Source Names at any time by using the ODBC Administrator. To use the ODBC Administrator, open the Control Panel and click on the ODBC icon.

- 11) Copy the ODBCVIEW.HP file from your PC client to the server. For example:
C:\cd windows\system


```
C:\WINDOWS\SYSTEM> DSCOPY odbcview.hp  
host#username/passwd.someacct/passwd#odbcvie
```

12) Execute the odbcvie command file on the server for each DBEnvironment.

```
isql=> CONNECT TO 'DBName';
```

```
isql=> START odbcvie;
```

```
isql=> exit;
```

13) The last step is to ensure the ODBC.INI file is setup for your specific environment and the ODBCINST.INI is setup correctly:

ODBCINST.INI file should have the following contents:

```
[ODBC Drivers]  
HP ALLBASE/SQL=Installed
```

```
[HP ALLBASE/SQL]  
Driver=C:\WINDOWS\SYSTEM\allbase.dll  
Setup=C:\WINDOWS\SYSTEM\allbassp.dll
```

The ODBC.INI file should have the following contents:

```
[ODBC Data Sources]  
Parts=HP ALLBASE/SQL
```

```
[PARTS]  
Driver=C:\WINDOWS\SYSTEM\allbase.dll  
Description=Parts Database  
LastUser=#mpeix/HP3000:PartsDBE.Somegroup.Someacct#session1,Someuser.S  
omeacct  
DefaultIsolation=
```

Gupta Interface:

Installing Gupta PC API Interface

The Gupta interface uses approximately 2 MBytes of disk storage. To install the Gupta interface, do the following:

```
1) C:\PCAPI\mkdir GUPTA  
C:\PCAPI>cd GUPTA  
C:\PCAPI\GUPTA> ../hpgupta
```

2) Run Microsoft Windows

- 3) Ensure that all applications other than Program Manager are closed
- 4) From the File menu, click on Run.
- 5) Execute setup to install the PC API files. Use the path where your PC API files are located.
`c:\pcapi\gupta\setup`

Configuring Gupta Interface

To DO list:

- 1) Edit the SQL.INI file
- 2) Check for duplicate files in other directories
- 3) Use Scriptor to verify the connection and create views
- 4) Test the application software

- 1) Edit the SQL.INI file

Each client should have one copy of the SQL.INI file. This file was installed with the other PC API files. This file contains router information and connection string information for the application software.

Following is an example of the [ALLBASE] section of the SQL.INI file:

```
[ALLBASE]
mpedbname=server1,virgo:partsdb.demo.imagesql,sk
mpeuser=sysadm,mgr.imagesql,demo
```

where

mpedbname is the keyword that identifies this entry as the database code, the node name and the DBEnvironment name.

Syntax for mpedbname:

```
mpedbname={SQLDatabase}, {node_name}:{DBEnvironment_filename}
[,network_type]
```

mpeuser is the FULL MPE logon string to be validated by the target server.

Syntax for mpeuser:

```
mpeuser={SQLUser},{mpelogon}
```

Syntax for mpelogon:

```
[session_id], {user_name}[/user_pass] {account_name}[/account_pass]
[,group_name[/group_pass]]
```

The group_pass can also be /? if you do not wish to imbed the passwords in the SQL.INI file.

2) Check for duplicate SQL.INI and SQLAPIW.DLL file.

Ensure these files are only found in the C:\ALLBASE directory

3) Use scriptor to verify the connection and create views

6) Client tools

What is the scriptor?

The scriptor is a tool created by Hewlett-Packard to verify the connection to your HP3000 without installing your application software. The tool will verify the connection and is used to install views in your environment which will be used by the application programs to access system files.

Where is scriptor?

The scriptor group should have been created when you installed the Gupta interface. In the Program Manager double-click on the icon labeled "HP ALLBASE/SQL PC API".

The scriptor dialog box will appear on your screen.

Executing the Verify Script:

The VERIFY.SCP file is installed with PC API. The script checks that the installation has been successful. It connects to the ALLBASE/SQL DBEnvironment, executes a SELECT statement, then disconnects.

To execute the script, enter verify.scp in the Path/Script Filename field and click on EXECUTE, or press ENTER.

As the Verify Script runs, it is echoed in the View Window on the bottom half of the screen along with any error messages it may encounter. The HP DB Router icon appears when a first connection is attempted.

If you have error messages, note the error, click on EXIT to leave the Scriptor dialog box, and return to MS-DOS to fix the errors before continuing on to the next step.

To execute the Views script:

The VIEWS.SCP is installed with PC API. The Views script installs the SYSSQL views in the DBEnvironment. These views map the ALLBASE/SQL system catalog tables to the server-independent system catalog views normally used by Gupta interface applications. Views Script also grants PUBLIC access to the system catalog views that it installs.

To execute this script, enter VIEWS.SCP in the Path/Script Filename field and click on EXECUTE, or press ENTER. The results will be echoed in the View Window at the bottom of the screen. After the Views script has finished, you can scroll up to see any error messages. Click on and hold down the scroll button to

move back through the output to see if the connection was successful. Comments are preceded with a *.

Any application using the Gupta interface must have access to the SYSSQL views. You can also install the views using the ISQL which comes with the Image/SQL product.

4) Test the Application Software

Install your application software following the instructions from the manufacturer. Make sure you change the path in your AUTOEXEC.BAT file to include the directory of your application software after the C:\ALLBASE directory.

6) Troubleshooting: What to do when it doesn't work!!

Troubleshooting is an art. The easiest way to gain knowledge in troubleshooting is to practice. You will get experience in the troubleshooting process when installing your new environment. The best advice I can give is to eliminate as many layers as possible before trying to troubleshoot. Remember the client/server environment is very different than the traditional HP3000 environment. I have found it very useful to test the environment one layer at a time. A network can go down at the blink of an eye and with the users experimenting with tools the system will encounter many new challenges. Following are some tools which are available to help in your troubleshooting venture.

Tools on the database server:

1) HP Glance/XL- this product will allow you to monitor a job, session, or a process on MPE/iX. 2) SLQMON- this product is part of the Allbase/SQL environment which will allow you to monitor the activity on a Allbase/SQL DBEnvironment.

Helpful Hints:

1) SET USER TIMEOUT- in Allbase/SQL you can set a time out feature to help avoid deadlock situations. This feature allows the user to set a timer for the length of time the user will wait on a lock. The default is NONE so any timeout length will help increase the concurrency on the activity.

2) Checking the Listener Log Files- the active listener will log all events to a log file. To see this information you can issue the following command:

```
:PRINT HPDANSLG.PUB.SYS
```

This file contains dates and times when the listener was started and restarted. It then shows the client node names of the user that has been successfully connected

by the listener and the client version. The listener log file is a circular file that holds 1000 entries. The last entry in the file is the most recent.

Tools on the PC Client:

You can capture the ALLBASE/NET packets can be captured using the PC API. The commands to capture this information are HPDRECORD and HPSQLNETLOG. You can use these tools with both the ODBC and the Gupta interfaces.

Gupta tool ----- Filter to capture calls/Scriptor to replay Gupta calls
ODBC tool----- Dr. DeeBee

HPDRECORD: This tool records ALLBASE/NET messages send and received across the LAN and captured on the PC client. The results are stored in a compressed binary file called HPDC.LOG.

To use the tool, do the following:

- 1) C:> set hpdcrecord=1 (turn on logging)
(note there are no spaces)
- 2) Run your application
- 3) C:> set hpdcrecord=0 (turn off logging)
- 4) To convert the HPDC.LOG file issue the following commands:
DISPLOG HPDC.LOG HPDC.TXT -n
- 5) Now text this file into the some editor and you will see the results.

HPSQLNETLOG: this tool records messages send and received across the LAN and captured on the database server. On the PC client, do the following:

- 1) From outside of Windows, turn on logging with the following commands:
setenv HPSQLNETLOG=255
- 2) You can set up a personalized logfile:
setenv HPSQLNETLOG_NUM=nnnn
where nnnn is any four digit number.

The database server log file name will be SLGnnnn in you home group. The default log name is HPSQLSLGB.

ODBC Tools: Dr DeeBee Spy

This tool will log all the ODBC calls along with the input and output values. To turn on logging following these steps:

- 1) From the Program Manager Menu in Windows, click on RUN.
- 2) Type DRDBSP

- 3) Choose OK.
- 4) Select the Data Source Name to trace.
- 5) Choose OK.

Dr. DeeBee icon will appear at the bottom of the screen. It will flash which it is logging:

To turn off logging:

- 1) Click on the Dr. DeeBee Spy icon
- 2) Choose CLOSE.

The log file is located in C:\WINDOWS\DRDEEBEE.LOG

GUPTA Interface Tools

Tracing Calls with Filter:

The filter is used to capture the calls sent by your application and then the scriptor is used to reproduce the problem.

To create the trace file, close the HP DB Router and rename two files:

```
rename sqlapiw.dll sqlapiw.sav  
rename filter.dll sqlapiw.dll
```

The filter creates a trace file named `TRLT.SCP` in the directory where your application is running. If a `TRLT.SCP` file already exists, it is appended.

To stop creating the trace file, exit the application and rename the files back to their original names:

```
rename sqlapiw.dll filter.dll  
rename sqlapiw.sav sqlapiw.dll
```

Using the tracer calls with the Scriptor:

In the Path/Script Filename windows type `TRLT.SCP` to execute the commands. You will see the commands echoed to the View Window screen.

The "Middle Steps" have been disclosed. This can be the most difficult part of implementing your new "Client/Server" environment, but hopefully this paper has helped clear up some of the issues involved with the PC API. This paper was written with the help of personal experience and the HP manual, HP PC API User's Guide for ALLBASE/SQL and IMAGE/SQL. If you are currently evaluating this environment, about to implement or have already implemented a client/server environment then be prepared for a challenge with your networking and PC API implementation.

**Paper Number 3005
IMAGE/SQL and Indexing**

**Larry Boyd
Director of Research and Development
Bradmark Technologies, Inc.
4265 San Felipe, Suite 800
Houston, TX 77027
713.621.2808**

Handouts will be provided at time of presentation

Does Your HP "Talk" Back?

Interactive Voice Response Data Input and Retrieval System for Hewlett-Packard Platforms.

Bruce Frank, President, Frank Solutions
9250 East Costilla, Suite 100, Englewood, CO 80112
(303) 792-5500

ABSTRACT: An introduction into the technology of PC-based automated telephone-operated information and fax retrieval systems, enabling data input and/or retrieval via voice-operated menu selection. Using Hewlett-Packard host platforms, a separate PC, shared disk and standard analog or digital telephone lines, the author's system uses voice and telephone touch tones to accept and distribute information from a host HP 1000, 3000, or 9000 system. Dramatic new technologies enable voice input to be stored on the host system as text, and data output as voice generated text, numbers or hard copy facsimile ("Fax-on-demand").

This paper will explore Interactive Voices Response (IVR) technology as it applies to Hewlett-Packard host platforms, for both data retrieval and input. Essentially, IVR technology enables access by outside users of HP host data from nearly any standard telephone, touch tone or rotary. Callers can choose specific documents or real-time data for faxing to a remote number, or receive information audibly through IVR text-to-speech conversion capabilities. In addition, the system is able understand voice commands given by a caller, and to input information to be converted to data by the host. Capabilities of this new system are made possible by the

addition of four new elements: Voice Boards, Fax Boards, Voice Recognition, and Text-To-Speech.

VOICE BOARDS

At the heart of the new technology are plug-in boards that provide a personal computer with the ability to understand the spoken word, as delivered by telephone. Also known as a speech card, these plug-in modules include a telephone interface and perform basic telephone functions like dialing, answering and hang up.

Specific to the voice processing function, voice boards record incoming audio to a digital file on disk as well as playing it back. Variations include choices in the number of incoming lines, different digitizing processes and compression methods, and sampling options such as rate, bits per sample, and silence compression. Each of these affects the amount of disk storage required and the quality of playback.

Voice boards also form the foundation for voice recognition and text-to-speech capabilities, which will be discussed later.

In use, voice boards enable the voice mail function, providing a means for staff to record information despite the lack of someone on the receiving end. Sales representatives can download spoken information on sales or contracts, messages may be left for co-workers, and anyone using a telephone can send or receive their message to or from disk without a live person at the other end to accept it. Incoming information can be of almost any length, and can be made accessible through security codes or passwords over and over again with no deterioration in quality.

FAX BOARDS

Part of the flexibility of Interactive Voice Response is its ability to integrate with the fax function. Fax boards convert file information on disk to a format that can be transmitted via telephone, for re conversion to a text or graphic format by the receiving fax machine. In response to a request from a caller, 2 types of faxes can be sent:

Existing documents can be stored and sent directly from the IVR system to a remote fax. Examples include sales literature, technical support documents, price lists, or anything of a detailed or lengthy nature.

A more powerful fax application is retrieval of real-time data such as account statements, accounts payable, accounts receivable, or any other host-based report.

TEXT-TO-SPEECH

Text-to-Speech (TTS) is the process of synthesizing spoken phrases directly from conventional text and numeric strings or files. TTS software works with voice boards to create audible speech that is easy to understand, although it is not yet at the point where it could be confused with a living person.

The most significant advantage of the TTS function is it allows an IVR system to convey ASCII information audibly, on demand. The information need not have been recorded by voice in order to be transmitted that way. Also, TTS allows more efficient storage of information; digitized alphanumeric information takes up approximately one-half of 1% of the spoken audio version on disk.

This opens up significant new possibilities in terms of easier dissemination of more current information. Banks and investment houses, for instance, can make up-to-the-minute information on account status, loan rates or stock prices available over the phone. Aviation weather services can provide totally current weather conditions, around the clock. News services can disseminate menu-based information as soon as it has been entered. Companies using e-mail and voice mail can integrate the two functions based on the equipment available at the polling end. Traveling sales representatives can retrieve e-mail messages by telephone through TTS. With the use of optical character generation, faxes can be relayed audibly to a staff member on the phone in the field. Organizations depending on transmitting information from large databases no longer need a voice recording of each entry that could possibly be retrieved.

Variations in TTS systems include the number of recognizers that can run on any one platform, options in synthesized voice volume, speed, gender, excitement, inflection, and language. TTS packages are available in more than a half-dozen major languages, including several dialects.

VOICE RECOGNITION

The most dynamic aspect of the IVR technology is the one at the heart of it: Voice Recognition. Also known as Speech-to-Text or Automatic Speech Recognition, voice recognition offers an alternative to touch tone input for a remote caller issuing instructions to a voice processing system. It operates by matching one or more spoken words to those in a stored vocabulary, including numbers, months, colors, letters of the alphabet, and even astrologic symbols. Performance is limited only by phone line

static, background noise and echo, which can make telephone based VR less efficient than higher-fidelity microphone based versions. Also, input no longer requires a touch tone phone at the sending end, a handy advantage for callers without access to one. (While only about 30% of the telephones in this country are rotary phones, that figure climbs to 95% in other countries.)

There are three classifications of voice recognition: Speaker-Independent allows a wide range of voices and accents, and responds to a relatively limited number of recognized words. Speaker-Dependent uses a wider vocabulary, but the recognizer must be trained to the way a limited number of speakers deliver them. And Speaker Verification is designed to precisely match individual voice patterns of a single speaker for access security.

Voice recognition is a more natural alternative to touch tone input. Job status information or account balances can be communicated, as can time reporting for construction crews on remote job sites, routine information requests for employee benefits, product or parts availability, and college course registration information. voice recognition automates credit card payment or bank funds transfer, and simplifies and accelerates order/data entry. Power companies can process incoming and outgoing information on power outages, and the local weather service offices can simplify severe weather warnings.

Voice recognition represents an easier way for radio stations to process the enormous volume of calls that come in during contests, and cable companies can use it for activating pay-per-view programming. Concert scheduling and ticket sales could be processed via voice recognition, lottery winners and losers could be advised of their status.

Retail chains could keep a tighter awareness of their cash positions if night managers called in deposit amounts at the close of business each day. Surveys could be expedited, as could health care patient information. Appointment scheduling is simplified by around-the-clock access. Hotel wake-up services become more accurate while eliminating the human element. Automated message boards take voice mail to a new dimension.

One of the greatest advantages of voice recognition revolve around its round-the-clock availability. Also, callers often prefer spoken commands to hunting for alphabet equivalents on the touch tone dial, considering it more user friendly. Its hands-free operation is far more easy for handicapped callers.

HP CONNECTIVITY

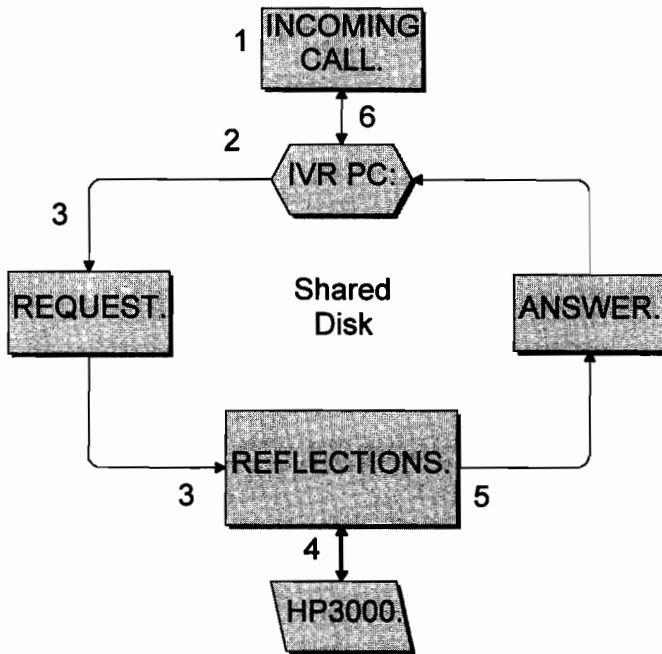
To take full advantage of IVR technology, connectivity to host based data is required(HP 3000, HP 1000, HP 9000). Terminal emulation provides an excellent facility for transferring data between the IVR system and the HP host; usually without the necessity of modifying or developing host application programs. An example utilizing terminal emulation and Message File Exchange is outlined below.

A bank uses an HP 3000 series 947 with an IMAGE database to track bank account information. Application software is in place so customer service reps. can take calls from members and answer questions concerning account balances, specific checks, etc... An IVR system was developed using WRQ's Reflections terminal emulation program and Message File Exchange (MFE) to provide data transfer between the HP 3000 and the IVR PC system.

MFE takes advantage of "shared disks" available to the IVR system and a PC running WRQ Reflections. The shared disk can be a Novell, UNIX, or any other file server as long as both PC's can "see" a shared drive.

Directories on the shared disk are assigned for storing request and answer files. The caller interacts with the IVR system and enters the required information for the query. The IVR system deposits a request file on the shared disk. The Reflections PC "picks up" the request and runs the application program on the HP 3000 to retrieve the appropriate data. The data is then written back to the shared disk as an answer file to be "picked up" by the IVR system and then is "spoken back to the caller".
(see diagram)

IVR / Hewlett-Packard Data Transfer



CUTTING COSTS AND IMPROVING SERVICE

Automated response is becoming an increasingly popular means for companies to improve customer service and speed basic input functions while significantly reducing overhead. A single system can replace up to 4 staff in functions that usually require their own access to a PC. The new developments in voice recognition and text-to-speech make remote access to information more flexible than ever. System payback is accelerated due to the relatively inexpensive investment.

This technology is significant because it represents first time it has been available on standard PC systems, at a cost structure that represents an extremely rapid return on investment, often within a few months.

IVR technology moves the power of active input or access from an internal staff member into the hands of the customer. It represents a strategic advantage for businesses interested in maintaining their competitiveness by getting still closer to their customer, while helping companies reduce overhead.

Proven IVR technology is now possible using widely available PC-based software connected to HP host systems. Not only does Interactive Voice Response exploit the functionality of automation that may already be in place, but it does so with the help of one of the most widespread, reliable, and easy-to-use tools available: the telephone.

Bibliography

**PC-Based Voice Processing by Bob Edgar Copyright 1992, 1994 FlatIron
Publishing Inc. 12 West 21 Street, New York, NY 10010**

Paper Number 3008
The New Relational Database: IMAGE/SQL

Brad Tashenberg
President
Bradmark Technologies, Inc.
4265 San Felipe, Suite 800
Houston, TX 77027
713.621.2808

Handouts will be provided at time of presentation

Paper #3009
Mary Ann Gustafson, Product Manager
Hewlett-Packard
10111 Proseridge Avenue, MS 44MB
Cupertino, CA 95014
(408) 447-1061

How to ... Manage Your HP Information Management Investment



- ◆ Today we are going to speak about how to manage your HP database investment. On the HP 3000, more than 95% of our customers are using an HP database, whether it's KSAM, TurboImage, IMAGE/SQL or ALLBASE/SQL...sometimes more than one at the same time. On the HP9000, ALLBASE/SQL is on both the 700 workstations and 800 multi-user systems. In 1993, we actually sold more ALLBASE licenses on HP-UX boxes than MPE and expect that trend to continue.

Your HP Info Mgmt Investment, Paper #3009-1

Business Realities - 1994



 HUBBARD
PACKAGING

- ◆ Speaking with customers, we find that many of the challenges they face are the same.
- ◆ Money is tight; you're continually asked to do more with less
- ◆ The business world is becoming increasingly competitive and many customers have found their IT investments can be the differentiator that keeps them ahead.
- ◆ Your end-users, in finance, manufacturing or marketing are becoming more demanding. They want faster and easier access to critical business data.
- ◆ With the wave of acquisitions during the 80s, many IT managers are now responsible for operations that span the globe and must be available 24 hours a day.

IT Solutions to Address Realities

Business Realities



Competition



Increasing User Expectations



Cost Controls



Globalization

IT Solutions

- Executive information systems
- Data warehousing
- End-user access to data
- Client-Server
- Application portability
- Migration to more cost-effective solutions
- Integration of data from worldwide systems
- 24x365 worldwide operation and support

 **HP** REALITY
Partnership

3

- ◆ To meet these business challenges, customers are selecting new IT solutions to solve their problems.
- ◆ Like most of you, they already have an investment in HP systems and databases or other vendors', and must leverage these investments, while adding the new solutions.

Your HP Info Mgmt Investment, Paper #3009-3

Reality #1: Competition



IT Solutions

Executive Information Systems

- Exception Management
- Drill down to key data

Data Warehousing

- Centralized decision support with remote data access
- Centralized back-ups for best utilization of MIS resources

HP Data Management Solutions

- IMAGE/SQL or ALLBASE/SQL with PC API and executive information tools
 - Trinzic's Forest & Trees
 - Cognos' PowerPlay

- HP Open Warehouse with ALLBASE/Replicate for shadowing and interoperability with heterogeneous systems
 - IBI EDA/SQL
 - Trinzic's Info Pump
- HP ALLBASE/Net for remote read/write access of IMAGE/SQL or ALLBASE/SQL from HP 3000 or HP 9000

 **HP** INVESTMENT
PAPER #3009-4

4

- ♦ Executive information systems allow access to data residing on systems throughout the company. They are programmed to alert executives when something important has happened, such as too much inventory or low sales in a certain region and allows them to easily get more in-depth information on the highlighted condition.
- ♦ Data warehousing allows you to remotely access and download data from HP systems or other vendors' to a central system for decision support, freeing your other systems for OLTP.
- ♦ As you can see, these solutions can be created using HP databases and 3rd-party solutions, allowing you to leverage your current database investment, while using these newer technologies.

Reality #2: Increasing User Expectations



IT Solutions

End-User Access

- Ad-hoc queries
- Reporting
- Data to desktop

Client/Server

- GUIs for ease of use
- Faster development
- Power to users
- Optimal use of PCs

HP Data Management Solutions

- IMAGE/SQL or ALLBASE/SQL with popular report writing tools
 - Cognos Impromptu
 - MicroSoft Access
 - HP ALLBASE/BRW
 - Pilot Lightship
 - HP Information Access
- IMAGE/SQL upgrade for TurboIMAGE
- ALLBASE/SQL
- PC/API (Gupta and MicroSoft ODBC)
- ALLBASE/Net
- Over 30 popular reportwriting, executive information and 4GL tools



- ◆ In many companies, end-users want direct access to key business data, instead of asking MIS to generate a report.
- ◆ To meet users increasing expectations, MIS is enhancing end-user access using popular PC tools and HP databases. This allows finance, mfg and other depts. to query your IMAGE or ALLBASE databases and easily generate reports to make the right business decisions.
- ◆ Everyone hears about client/server or PC integration and you can do it using your HP databases and the PC APIs that are now bundled with them. These products enhance current IMAGE applications with easier to use front-ends, speed development of portable applications using popular tools from the leading vendors and effectively utilize the personal computer investment your company has made.

Reality #3: Cost Controls

IT Solutions

Application Portability

- Protection of application investment
- Applications on multiple vendor systems
- Single development environment

Migration to More Cost-Efficient Solutions

- Less costly platforms to purchase/maintain
- Access to new technologies

HP Data Management Solutions

- IMAGE/SQL or ALLBASE/SQL with popular 4GL tools from leading vendors.
- ALLBASE/SQL standards compliance (ANSI, ODBC, FIPS, 3GL preprocessors, XA for two-phase commit for TP monitors)
- Multiplatform applications to run your business

- Mainframe database replacement with ALLBASE/SQL for lower cost
- Leverage TurboIMAGE investment to IMAGE/SQL.
- ALLBASE/SQL and IMAGE/SQL are 50-80% cheaper over five years than other databases on the HP 3000 and HP 9000

 **HP** PERSONALITY

6

- ◆ With limited IT budgets, depts are faced with a make vs. buy decision with each investment and protection of their investments is critical.
- ◆ Portable applications can be developed with HP databases using popular 4GLs from vendors such as Gupta, Microsoft, Cognos, Progress and others.
- ◆ Developers who adhere to the industry standards such as SQL, ANSI, ODBC and XA that are found in ALLBASE can port their applications to other databases and hardware platforms.
- ◆ Applications running on HP and other databases are available
- ◆ As hardware costs decrease and new technologies are introduced, customers can enjoy lower system expenses, while enhancing their current environment.
- ◆ Mainframe customers find significant saving using high-end HP systems with ALLBASE to replace their mainframes.
- ◆ HP databases typically cost 50-80% less than expensive alternatives

Reality #4: Globalization



IT Solutions

Integration of Data from Worldwide Systems

- Global decision support
- Remote backups

24x365 Operation

- 100% system availability for users
- Fault tolerance

HP Data Management Solutions

- ALLBASE/NET for remote access from HP 3000 or HP9000 of IMAGE/SQL or ALLBASE/SQL
- ALLBASE/SQL with ALLBASE Replicate
- Third-party heterogeneous solutions (IBI EDA/SQL, Trinzic Info Pump)
- Eight and 16-bit localizable ALLBASE/SQL

- HP ALLBASE/SQL with ALLBASE Replicate or Shareplex for remote, unattended back-ups
- On-line back-ups with ALLBASE/SQL
- HP IMAGE/SQL with Shareplex
- 24x365 HP Response Center Support for one-stop support of *hardware, software and databases*



- ◆ Executives need access to key data, whether its down the street or around the world. Using remote access from a central server running ALLBASE or IMAGE to other HP databases or shadowing of data to a central location, MIS can do global decision support.
- ◆ Using these same capabilities, data can be backed up to a central location to maximize limited MIS resources.
- ◆ ALLBASE's 8 and 16-bit capabilities allow you to use it for Asia.
- ◆ Another challenge is keeping systems available to users 24-hours a day, 7 days a week for critical applications. Using ALLBASE Replicate, for centralized back-ups or ALLBASE's true on-line back-up capabilities, uptime can be increased to 100%.
- ◆ Shadowing capabilities available with both HP databases allow you to shadow critical data to another location that can take over should your primary system go down for any reason. HP's worldwide support is another reason why customer's purchase HP database.

HP Database Strategy

Objective: *Provide the databases and tools to solve customers' evolving information needs and to address changing business realities*

- HP databases low price, superior performance, quality and one-step shopping/support
- 3rd-party databases for multiple platforms and applications
- 3rd-party decision support, EIS and 4GL tools, and standard APIs for evolving current investments and application portability for new applications.
- Continued HP investment in HP databases to:
 - provide world-class third-party applications
 - protect current customer investment in HP databases
 - provide technology to evolve current or develop new information management solutions

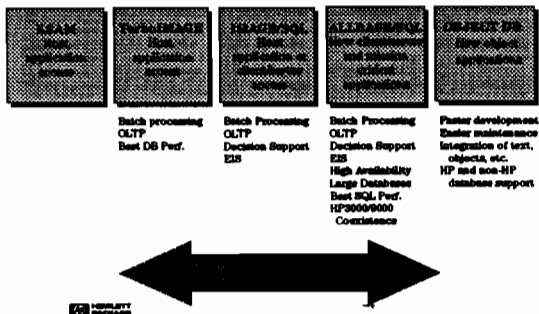
 **HP** HERSHEY
PARIS

8

- ◆ HP's database strategy can be summed up in one word: choice. HP is committed to providing the databases and tools are customers need to run their businesses.
- ◆ HP databases are optimized for PA-RISC and HP operating systems for superior performance at a low cost. Customers can purchase HP quality with the confidence that HP support is there for your assistance.
- ◆ 3rd-party databases are easier to administer in a multi-platform environment and may offer a greater selection of applications.
- ◆ 3rd-party tools and industry standard APIs allow HP database customers to update or access data on IMAGE databases, while developing new portable applications that can be deployed on HP or other databases.
- ◆ HP has been in the databases business since the early '70s. We are committed to continue enhancing and supporting our databases, while providing the applications and features our customers want.

Your HP Info Mgmt Investment, Paper #3009-8

Evolution with HP Database Solutions



- ◆ Regardless where a customer is on the evolution cycle, HP databases are there to meet their key business needs.
- ◆ Customers often are using more than one databases depending upon when new applications were added.
- ◆ About 95% of HP3000 customers are currently running TurboImage applications. Of those, 70% have accepted the IMAGE/SQL upgrade and 10% are using IMAGE/SQL with another 55% planning to use in '95.
- ◆ Over 60% of the HP3000 installed base also has ALLBASE/SQL and 15% are using it with another 25% planning to use in '95.
- ◆ Usage of 3rd-party databases on the 3000 is below 7% with a 5% increase planned for '95.
- ◆ On the HP9000, customers are typically at the SQL stage of the cycle with either a 3rd-party database or ALLBASE/SQL. In 1993, HP sold more ALLBASE licenses on HP9000s than HP3000s.

Your HP Info Mgmt Investment, Paper #3009-9

Database Positioning

HP Databases

- Investment protection **AND** application portability
- Best price/performance for HP servers and workstations
- Single-vendor development, sales and support for no fingerpointing
- ALLBASE developed for high-end HP systems with large databases and many users
- Best integration with IMAGE data and applications with full read/write capabilities

 **HP** INTEGRITY

3rd-party Databases

- Application portability
- Focus on providing databases on multiple hardware platforms
- Single vendor database, tools, CASE, etc.
- PC to mainframe databases
- Wide selection of applications
- Integration with other 3rd-party and some system databases
- Corporate focus on databases

10

- ◆ HP databases offer a number of benefits vs. 3rd party databases.
- ◆ Ability to leverage your current IMAGE application and data investment and develop new portable applications to run your business
- ◆ HP databases are developed only for and to help sell HP systems and, you will find the best price and strongest performance from them.
- ◆ HP stands behind your database, operating system and hardware in development, sales, consulting and support, while supporting industry standards and development tools that allow for application portability.
- ◆ ALLBASE is the only relational database on the HP9000 with full read/write capabilities to IMAGE data.
- ◆ A single 3rd-party database is easier to maintain and administer in a multiplatform environment.
- ◆ 3rd-party databases provide good integration with other 3rd-party databases and some system databases, generally not HP databases.

Database Differentiators

- As databases become commodities, customers are looking at other differentiators including:

- * Quality and level of customer support
- * Breadth and quality of tools
- * Corporate profile and stability
- * Degree of standards compliance
- * Number of consultants and developers with knowledge
- * Research and development organization and budget

 HEMLETT PROGRAM

11

- ◆ While the industry press talks about the latest features, MIS departments are recognizing that there are other criteria that are more important in the selection of a database.
- ◆ Customers with mission critical applications select HP databases for HP worldwide support
- ◆ The leading decision support, EIS and development tools are available on HP databases.
- ◆ HP is financially stable and not in danger of being acquired by another company.
- ◆ ALLBASE was developed using industry standards and Object DB is helping to establish standards for the object world.
- ◆ HP has consultants and 3rd-parties to assist in planning, development, deployment, maintenance and administration of your databases.
- ◆ HP has been in the database business since the early '70s and will continue to enhance our databases to meet our customers' needs.

Your HP Info Mgmt Investment, Paper #3009-11

IMAGE Upgrade Brings Information to Users

Manufacturing: Viking Pump

Objectives:

- Enhance competitiveness by providing end-users with a flexible and open environment to access enterprise-wide data
- Protect company's assets and integrity of data by utilizing centralized servers

Solutions:

- Upgrade TurboIMAGE to IMAGE/SQL with ODBC and use Cognos Impromptu to bring ManMan data to end-users
- Transition from terminals to PCs integrated in a networked environment
- Facilitate sharing of office automation tools by adding LAN Manager for file and print sharing, and centralized back-ups.
- Connect E-Mail client to OpenDesk on HP3000

Benefits:

- More timely and better access to company-wide information
- Optimization of computing resources with LAN and central servers
- Greater flexibility to meet changing business needs



12

- ◆ Viking Pump is a typical HP3000 customer who decided to evolve its environment to take advantage of new technologies. These included:
- ◆ Transition from terminals to PCs in a networked environment (LAN Mgr.) that allowed sharing office automation tools.
- ◆ End user access via a network to manufacturing applications via WRQ's networked terminal emulation solution
- ◆ Upgrade from TurboImage to IMAGE/SQL with ODBC API and Cognos' Impromptu decision support tool to access ASK ManMan data.
- ◆ Connect users to E-Mail client of choice (cc:mail or MS Mail) with HP OpenDesk to increase communication within company.
- ◆ Results include: Ability to leverage current HP3000 environment to use new technologies vs. trying to start over from scratch, more timely access to key info via decision support and E-Mail, optimized use of computing resources and greater flexibility to address changing business needs.

Putting Enterprise Information to Work with IMAGE/SQL

Decision Support

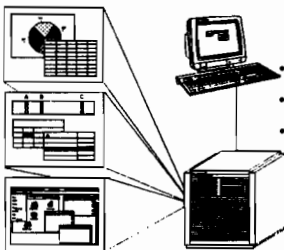
- Cognos Inpromptu
- Gupta Quest

Executive Information

- Trinsic Forest & Trees
- Pilot Lightship

New Application Modules

- Gupta SQLWindows
- Powersoft Powerbuilder
- Microsoft Visual Basic



Your IMAGE Investments are fully protected

- Run existing IMAGE applications
- No charge to database administration
- Leverage existing application expertise

IBM BUSINESS PARTNER

13

- ◆ One of the key ways Viking was able to get the benefits we spoke about was by upgrading from TurboImage to IMAGE/SQL. About 70% of our TurboIMAGE installed base has upgraded to IMAGE/SQL.
- ◆ It means they can still run their existing IMAGE applications unchanged while having many of the advantages of an relational database.
- ◆ These advantages include the use of popular PC tools that have been written to work with SQL databases for decision support, executive information systems or development of new application modules.
- ◆ HP has bundled the PC APIs needed by customers (Microsoft ODBC and Gupta) with IMAGE/SQL at no additional cost.
- ◆ IMAGE/SQL also has ALLBASE/Net bundled in with it. For customers who are operating in a mixed ALLBASE/IMAGE and/or 3000/9000 environment ALLBASE/Net allows full read/write capabilities between IMAGE and ALLBASE databases, regardless whether they reside on HP 700s, 800s or 900s.

Your HP Info Mgmt Investment, Paper #3009-13

Coexistence for Business Flexibility

Manufacturing: 3M

Objectives:

- Share data between HP3000 and HP9000 systems to eliminate duplicate data entry and manual, time consuming procedures
- Leverage legacy HP MM application and add new applications
- Evolve business to make-to-order from make-to-stock

Solution:

- Monitor/UX with ALLBASE on HP9000 accesses HP MM data on HP 3000 via ALLBASE/Net for true interoperability
- Use PC API and PC-based reporting tool with IMAGE/SQL and ALLBASE/SQL
- Utilize ALLBASE/Net read/write capabilities

Benefits:

- 30% reduction in finished goods inventory
- Order turnarounds decreased from two weeks to two days
- Master schedule could be built in 1/2 hour instead of six



- ◆ Speaking of IMAGE/ALLBASE and 3000/9000 coexistence, here's another traditional HP3000 customer who is leveraging its current investment in HP3000s and TurboImage, while adding HP9000s and ALLBASE/SQL.
- ◆ This facility of 3M manufactures data cartridges, computer tape and the well-known sticky yellow notes or Post-It notes.
- ◆ Prior to the interoperability project, the plant received orders for finished goods that were already in inventory, allowing little adjustment for special orders. They also manufactured products to match the master production schedule, not customer orders.
- ◆ As a result of adding new applications on the 9000 with ALLBASE and utilizing the ALLBASE/Net capabilities between IMAGE and ALLBASE databases, allowed them to eliminate manual re-entry of data between the two systems, resulting in the benefits listed above.

Engineering for Business Flexibility

Retail: Long's Drug Stores

Objectives:

- Rapid development of new client/server applications for order processing and pharmacy decision support
- Client and development software independent of database and hardware.
- Ability to manage large volumes of data for more than 300 stores

Solution:

- Three-tier architecture of IBM POS systems, in-store HP3000s and HP3000 992/200 at corporate headquarters using ALLBASE/SQL and OSF's DCE.
- Applications developed using PC tools, ALLBASE PC API (ODBC) and ALLBASE/SQL to manage product and vendor data.

Benefits:

- Open HP3000/ALLBASE solution allowed selection of best of breed tools
- New architecture for high volume transaction processing allows client access to DCE/ALLBASE servers without regard to location on network.



- ◆ Another HP3000 customer using client/server technology to enhance competitiveness.
- ◆ Long's is a U.S. based retail chain with over 300 stores. They are using HP3000s and ALLBASE to develop two new applications, although they are a long-time IMAGE user too.
- ◆ The order processing application will be collecting sales information from IBM point-of-sales systems to in-store 3000s and then downloaded to an HP3000 2-way multiprocessor. This information will allow Long's to more effectively manage inventory and work with its hundreds of vendors to order new inventory.
- ◆ This OLTP application allows Long's to use the latest client and development software independent of the database and hardware platform for complete portability. The use of DCE (distributed computing environment), allows client access to the DCE/ALLBASE servers anywhere on the network increasing Long's business flexibility.

Downsizing for Cost Reduction

Manufacturing: Barber Colman

Objectives:

- Reduce mainframe maintenance expenses
- Implement client/server technology
- Migrate legacy data to relational database

Solution:

- Replace Unisys DMS-II with ALLBASE/SQL on HP 9000 Model 890/4-way
- Utilize systems integrator and Cobol conversion tool for fast migration of over 2000 applications
- Two-step process that will provide relational, client/server applications

Benefits:

- DP savings of \$1 million per year (33% reduction)
- Restructuring allows implementation of relational and client/server for better business flexibility



16

- ◆ A final customer case history, this time on the HP9000. This customer was using a Unisys mainframe with their DMS-II database. To reduce mainframe maintenance expenses, Barber Colman sought out a lower cost alternative.
- ◆ Barber Colman migrated over 2000 business critical, Cobol applications to an HP9000 Model 890/4-way using a system integrator and its Cobol conversion tool.
- ◆ It is the first step in a two-step process that will entail the development or purchase of new relational, client/server applications.
- ◆ Barber Colman saved over \$1 million per year and was able to accomplish this conversion in less than one year.
- ◆ They looked at other databases besides ALLBASE, but after they matched them up feature by feature, felt ALLBASE was a better return on their investment, with the capability to handle large transaction volumes—at a fraction of the cost.

Your HP Info Mgmt Investment, Paper #3009-16

HP ALLBASE/SQL Today

- Complete feature set comparable to popular RDBMSs
- Costs 50-80% less than alternatives with excellent performance.
- Mainframe database capabilities
 - Very large databases (>50GB)
 - Many users (500+)
 - Database shadowing for fault tolerance
 - 24 x 7 system access and support
- Remote data access and PC interfaces *free!*
- Only RDBMS with read/write access to IMAGE data
- Industry standards and 4GL tools for portability

 HPE
Hewlett-Packard

17

- Why ALLBASE then? When customers compare ALLBASE to the leading RDBMSs they find it offers a comparable feature set at a cost of 50-80% less over five years.
- ALLBASE was designed for high-end applications. Its capabilities offer very large databases, many users, shadowing or replication of data and our worldwide customer support.
- ALLBASE is the only relational database running on the HP9000 offering full read-write access to IMAGE data. It allows you to add new ALLBASE applications, while still accessing data residing on IMAGE databases on the HP3000, for true HP3000/9000 coexistence.
- Industry standards and support of leading 4GLs like Progress, with 1000s of applications, means you have complete investment protection and access to the latest applications and technology.

Your HP Info Mgmt Investment, Paper #3009-17

What's New with HP ALLBASE/SQL G.0?

- Open Enhancements
 - Microsoft ODBC support via PC API bundled FREE!
 - Enhanced Sybase compatibility
 - Progress 4GL/applications with ALLBASE/SQL in beta
 - ANSI/SQL'92 entry-level compliance
- Faster recovery of large databases for higher availability
- 24 x 7 worldwide support for unavailable databases
- High performance of large applications
 - Still best price/performance of any relational database running on HP systems
 - ALLBASE SQL performance tool at no additional cost!
 - Increased catalog concurrency for better parallelism
 - High speed erasure via truncate table command

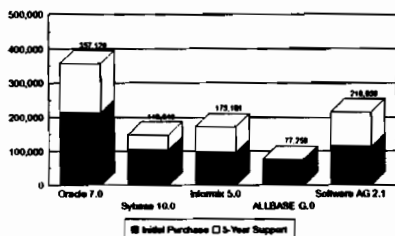
 **HP**
Hewlett
Packard

18

- ◆ Some of you may have heard about the introduction of ALLBASE/SQL Version G.0 in May. It offers new open enhancements such as Microsoft ODBC, and ANSI/SQL '92 compliance. In addition, specific Sybase and Progress database features have been added to ALLBASE G.0 to allow the port of the Progress 4GL to ALLBASE and the easy porting of applications between Sybase and ALLBASE.
- ◆ A new partial roll-forward recovery feature allows ALLBASE users to quickly recover a selected portion of the database, often while having the rest of the database on-line. It is a key feature with large databases.
- ◆ HP's worldwide support has now been extended to help customers to quickly return their database on-line with no fingerprinting.
- ◆ ALLBASE has always been known as the best price/performance SQL database running on HP systems and continues this leadership. We've now bundled in a performance tool that usually costs \$50k from other vendors. Additional features also further enhance performance..

Cost of Ownership Comparison

- ALLBASE Costs 50-80% Less Over 5 Years Than 3rd-party Databases!
- Networking, PC API, DBA and Performance Tools Free!



HP
HEWLETT
PACKARD

10

- ◆ This slide shows the 5-Year cost of ownership for ALLBASE vs. other databases available on the HP9000. Pricing information taken from published 1993 TPC benchmarks and is for user levels between 120 and 256 users.
- ◆ It demonstrates that ALLBASE is an excellent value for customers considering a SQL database on the 3000 or 9000.
- ◆ One reason is the low cost of support; typically less than \$50 per month, in comparison to several hundreds of dollars from other vendors.
- ◆ Another reason is many extras such as networking, ODBC, performance monitoring tools and more are bundled in at no additional cost.

Your HP Info Mgmt Investment, Paper #3009-19

Standard ALLBASE/SQL Features

Industry Standards for Open, Portable Applications

- ANSI/SQL 92, Entry
- Free Microsoft ODBC and Gupta APIs
- PIPS 127.1
- X/A Compliance/Two-Phase for C/CA, Transarc and Tuxedo

Support for Mission Critical Applications

- Worldwide 24 x 7 support for unavailable databases
- "Hot" online backup for 24-hour availability
- Forward, rollback and on-line recovery
- Replication for disaster recovery/remote back-up
- Row/page/table locking
- Multi-processor support

Powerful Application Development Capabilities

- Stored procedures
- Database triggers
- Referential integrity
- Client/server support for BLOBs and multi-row stored procedures
- Over 15 leading 4GLs support ALLBASE/SQL

High Performance for Queries and Updates

- ALLBASE/SQL performance tool bundled FREE!
- Cost-based optimizer
- Hash and B-tree indices
- Outer joins
- Faster table scans with parallel pre-fetch
- Optimiser access path and modification

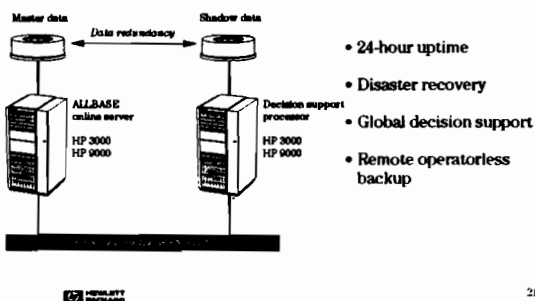


HP
IMMUNITY
PROTECTED

20

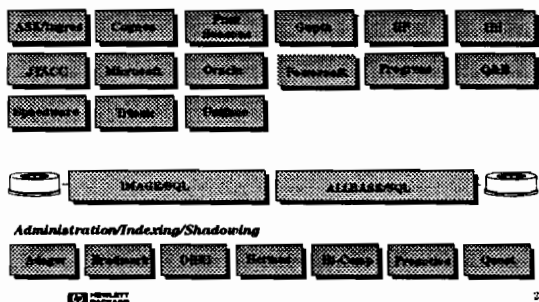
- ◆ When customers look for a new database to add to their present environment, they may have a list of features they want or need.
- ◆ Matching up feature to feature, HP's ALLBASE has the same features the other databases have, while providing read/write access back to your IMAGE data.

ALLBASE/Replicate Today Database Shadowing



- ◆ Another new database technology customers may be reading about in the industry press is replication or shadowing. Many vendors have announced this capability, but few are delivering on HP platforms.
- ◆ ALLBASE/Replicate was introduced in early 1993 and had already been in production for several years as a factory special.
- ◆ For mission critical applications, your database is shadowed on another system, should your primary system be unavailable for any reason, thereby preventing interruption of your business—whether for earthquakes or disk failures.
- ◆ Replication can be used to shadow data from multiple systems to a central location for decision support, as discussed earlier.
- ◆ MIS resources can be maximized by having remote system data shadowed back to central or branch offices for back-ups.

Tools for Use with ALLBASE/SQL or IMAGE/SQL



- ◆ We've been speaking about all the benefits you can gain using tools: decision support, executive information systems, client/server applications, etc.
- ◆ Here's a sampling of the many tools available. In most cases, more than one product is available from vendors such as Microsoft Visual Basic, Access and Excel.
- ◆ In addition, HP has relationships with popular vendors that provide indexing capabilities on IMAGE and databases administration tools.
- ◆ Quest provides additional software that enhances the replication capabilities, along with implementation assistance.

Applications to Run Your Business

- | | |
|---------------------------------|---------------------------|
| • Advanced Laboratory Systems** | Mitchell Humphrey |
| • American Int'l Healthcare** | MultiView |
| • Ares | PeopleSoft |
| • ASK** | Phoenix International* |
| • BSA, Inc. | Progress* |
| • Collier-Jackson | Public Safety Systems |
| • Computer Financial Services** | QAD* |
| • Computer Power Group | RMS |
| • Computron | Satcom |
| • Dun & Bradstreet** | Silton** |
| • DRC | SIS* |
| • Filenet* | Smith, Gardner & Assoc.** |
| • Financial Data Planning | Spectrum Assoc.** |
| • HP Mechanical Design | Speedware |
| • Hulco | Summit |
| • Index** | VTL** |
| • LPO | |

* New ALLBASE
** For IMAGE only

HP
SERVLET
PACKAGE

3

- ◆ While some customers prefer to develop their own applications, many choose to purchase third-party applications to run their businesses.
- ◆ In some cases they use with no changes or may customize for integration with other applications or to fit a particular business need.
- ◆ As you can see, this sampling applications available on IMAGE and/or ALLBASE provides a wide variety of choices for you.
- ◆ They span medical, engineering, manufacturing, financial, banking, distribution, libraries, human resources ,etc.
- ◆ With the port of Progress 4GL to ALLBASE, thousands of new applications will also become available to HP customers on the HP3000 expanding their choices even further.

Your HP Info Mgmt Investment, Paper #3009-23

Evolving Your Information Management Investment

- How to leverage your current information management investment as your business needs evolve
- How HP information management solutions can address your business realities now ... and into the future
- How customers like yourself are using HP information management solutions today to:
 - Bring data to end users
 - Improve decision making to meet competitive pressures
 - Reduce costs while increasing productivity
 - Share data among multiple systems to integrate current and new investments



 **HP**
INFORMATION
MANAGEMENT

24

- ◆ Today, we have discussed how to manage your information management investment.
- ◆ This paper has highlighted the business challenges faced by companies like you and how they are using data management technology to solve them.
- ◆ We have talked about how you can leverage your current IMAGE investment, with new front-ends, decision support or executive information applications.
- ◆ In addition, you can build new portable applications using popular tools on both IMAGE or ALLBASE.
- ◆ We've also looked at customers like you and how they are using HP database products to meet changing business needs.

Working with HP ALLBASE/SQL PC API

Roy Losch
2000 S. Park Place
Atlanta, GA 30339
(404) 850-2232

PC API began as a product a little over two years ago, targeting the client/server marketplace. PC API is based on Gupta Technologies, Inc. C/API. Initially PC API was offered on the HP 3000 HP-PA series, and about six months later the HP 9000 HP-UX platform was completed. The PC API interface can also access TurboIMAGE/XL databases through the IMAGE/SQL interface. Not only can users with ALLBASE/SQL dbenvironments take advantage of the large number of 4gl's and GUIs available today, but also the vast TurboIMAGE/XL installed base.

The thrust of this paper is to provide the end user with information about the components that make up HP ALLBASE/SQL PC API. Having a good working understanding of the pieces that make up HP ALLBASE/SQL PC API can lead to a smoother transition to client/server technologies in the MIS environment. HP ALLBASE/SQL PC API will be referenced to as PC API for the rest of this paper. One of the main difficulties in getting started in PC API is understanding what is needed to achieve success. The intent of this paper is to make clear the pieces of PC API and some of the trouble-shooting techniques available. The two main topics that will be covered are the server requirements and client requirements for PC API. In addition, PC API ODBC and IMAGE/SQL will be discussed.

1.0 SERVER REQUIREMENTS

The server is required as a database engine, and is sometimes called the backend processor. The server in this environment will be the HP 3000 or HP 9000 respectively. The HP 3000 server will be discussed first, then the HP 9000.

1.1 HP 3000 Server Requirements

On operating system MPE/iX 4.0, running ALLBASE/SQL F.0, PC API architecture requires that a listener process be up and running. The listener process is ALLBASE/NET, and comes as part of ALLBASE/SQL with the 4.0 release. This is a plus since ALLBASE/NET was a separately purchased product on releases prior to MPE/iX 4.0. ALLBASE/NET does not require any configuration outside of ensuring that the listener process is running.

The following command can be issued at a colon prompt to verify the status of the listener process:

```
:NSCONTROL STATUS
```

Check for the service named HPIPNS. If the service is not present, the network should be shut down, and the job JCONFJOB.NET.SYS should be streamed. Once the job stream has completed, the network can be brought back up. A system restart is not necessary to have the change take effect. Verify that the HPIPNS service now appears under the services section.

Continuing on with the discussion of server software, the version of ALLBASE/SQL must be checked. To check versions, simply issue the system command :SQLVER at a colon prompt. This will list all of the individual modules that make up ALLBASE/SQL as well as the overall software version. The overall version should be F0.59 or higher for the 4.0 version of the MPE/iX operating system to be successful with PC API. The next step is to make sure access to the dbenvironment is available with ISQL and verify the user has connect authority. With the 4.0 MPE/iX release, the connection architecture used for PC API requires a job to be launched under most circumstances. The job limit should be set accordingly to allow for this process.

One other piece of information of importance is the IP, internet protocol, address of the system. The IP address is used in configuring the HOSTS file on the PC. Issue the following system command:

```
:NETCONTROL NET=LAN1;STATUS
```

If you are using Novell NetWare for your connection, NetWare/iX is a requirement for the HP 3000. The status of NetWare/iX can be verified by logging on as MGR.NETWARE and issuing the following command at the colon prompt:

```
:NW STATUS
```

ThinLAN 3000/iX Network Link is a necessary product on the HP 3000 for use with any network environment solution.

The connection architecture for PC API under the ALLBASE/SQL F.0 version uses a job to interface with then HP 3000. This connection architecture was necessary to implement PC API at first. With ALLBASE/SQL G.0 introduction, the connection architecture will be a daemon type process which launches son processes upon request.

The new architecture should result in a substantial performance gain. Launching son processes would also alleviate having to allow for jobs launching on the HP 3000. The original connection architecture has been a major issue for customers when trying to work with the PC API environment.

1.2 HP 9000 Server Requirements

The HP 9000 the equivalent of the ALLBASE/SQL listener process is the hpdaARPA process. The following command will verify hpdaARPA is running:

```
$ ps -ef|grep hpdaARPA
```

If the listener is running, a display similar to the following will result:

```
rjl 9087 8997 0 16:56:57 ttyv0 0:00 grep hpdaARPA
```

If the listener is not running, enter the following as root:

```
$ /usr/bin/hpdaARPA
```

It is also necessary to make sure that the default port number being used is 987. Look at the file /etc/services and verify that the following line is present:

```
DAserver 987/tcp #SQL distributed access
```

As with the HP 3000, verifying the version of ALLBASE/SQL is important. The version of ALLBASE/SQL should be at least F0.59 or higher. To verify the version of ALLBASE/SQL, issue the command sqlver. Make sure that all modules are the same release level. It is possible to tar (restore) the ALLBASE/NET module while the ALLBASE/NET daemon is present. With the daemon present, the ALLBASE/NET software will not install, leaving the module versions at different levels. To acquire the IP address of the HP 9000, use one of the following command strings:

```
nslookup "HPUX" or execute  
/etc/ifconfig lan0
```

The IP address will be used when configuring the HOSTS file on the PC. It is important to verify that the ALLBASE/SQL dbenvironment is present, and that connect authority has been granted to the user.

2.0 CLIENT REQUIREMENTS

2.1 Basic Configuration

At a minimum the client architecture should be a 486 processor with at least 4 megabytes of memory, and preferably 8 megabytes. In most cases, applications combined with networking software will have better performance with 8 megabytes. 386 processors are supported, but will not perform extremely well in the client/server environment. The faster the CPU the better when it comes to client/server. Either an Ethernet or Ethertwist lan card must also be installed to work in a lan environment. A mouse or "pointing device" is also necessary. The version of DOS should be 5.0 or higher. The Windows version should be at least 3.1 or higher. Given DOS memory constraints, a memory manager would be recommended, but is not necessarily required. Networking software is required, and is discussed in section 2.4.

2.2 PC API Installation & Versions

PC API installation is done through the Windows SETUP.EXE facility. Restore the PC API software to a temporary directory and execute SETUP.EXE. PC API installs underneath the \ALLBASE directory on the PC. To find out what version of PC API is installed, the WHAT.EXE utility is provided with PC API. Different configurations determine what version of PC API is necessary. For most environments the current revision of PC API is not a problem, and is recommended. The latest version of PC API at this time for the ALLBASE/SQL F.0 release is F0.33. The latest version of PC API at this time for ALLBASE/SQL G.0 is G0.03. Executing the WHAT string on the DBALLBAS.EXE file provides information back on the version of PC API:

```
C:\ALLBASE>what dballbas.exe
dballbas.exe:@<#><c>COPYRIGHTHewlett-Packard Co.1994 All rights reserved
@<#> Hewlett-Packard ALLBASE/SQL PC API. Version: A.F0.30
```

2.3 PC API Configuration

Once PC API has been installed, the next step is to examine the configuration files. The main configuration file is SQL.INI, and an example SQL.INI is shown in Appendix B. The two primary sections are the winclient.dll section and the ALLBASE section. A section is denoted by brackets []. The winclient.dll section specifies the router which is going to be used. The router used to access ALLBASE/SQL is sqlawin. Other examples of the router options are sqldbw for Gupta's SQLBASE or sqlowin for Oracle. The ALLBASE section is very important because it documents all of the configuration strings. The necessary configuration strings minimally define a database and a user. Multiple configuration strings can exist in the SQL.INI file, but only one SQL.INI file should exist on the PC at any given time.

Working with HP ALLBASE/SQL PC API

When beginning to test PC API, the following syntax should be used:

```
mpedbname=server1,nodename:db.group.account,network type  
mpeuser= sysadm,ssid,user/pass.account/pass,group/pass
```

The mpedbname string indicates the server that is going to be used, the database name and the network type. The nodename should not be fully qualified with the domain.organization. If the server fully qualified name is HP.HP.HP, then the nodename reference should only be HP. The nodename is configured into a HOSTS file residing on the PC. The HOSTS file contains the IP address and the nodename of the server being used. The network type at this point can be one of the following:

- ip - W3IPC.DLL (NetIPC, HP 3000 servers only)
- sk - WSOCKETS.DLL
- nw - NetWare (HP 3000 servers only)
- ws - WINSOCK.DLL
- lw - WLIBSOCK.DLL (Novell LAN WorkPlace)

The mpeuser string contains the completing information about the user. The correct logon information must be used once the connection is established to the server. The passwords can be imbedded in the SQL.INI file, or prompted for. To cause PC API to prompt for passwords, put a question mark instead of the actual password. Issues exist around embedding passwords in the SQL.INI file because of security. Anyone who has access to the PC would have access to any imbedded password information. Not including the passwords could be a problem based on whether the 4gl software package issues multiple connect strings, as this would force being prompted multiple times for the same password information.

The configuration strings for accessing an HP-UX system are as follows:

```
hpuxdbname=server1,nodename:/DBEnvironment,network  
hpuxuser=sysadm,user:passwd
```

These lines should be included in the SQL.INI file, under the ALLBASE section. The syntax is very similar between MPE/iX and HP-UX. The same nodename configuration concerns are still present when accessing HP-UX servers. The nodename should not be fully qualified in the HOSTS file. The hpuxdbname string contains the server identifier, nodename, dbenvironment path information and network type. The hpuxuser string contains the user code, login name and password.

2.4 Network Requirements

PC networking has several basic concepts that for the most part will be consistent across the different networking products. In all cases a HOSTS file will be used on the PC to resolve an IP (internet protocol) address to a particular nodename. This file should be named HOSTS, and not have any extensions such as .TXT following the file name. The HOSTS file contains the IP address, followed by a nodename. When working with PC API, the nodename should not be fully qualified in the HOSTS file. For example:

15.1.2.3 HP (is correct) 15.1.2.3 HP.SITE1.HP (is incorrect)

The PING facility is used to test the configuration of the HOSTS file. This facility is provided with most network vendors. PINGing the nodename proves that all is in order with the PC network software, and communication to the server. If this simple test fails, PC networking must be addressed prior to continuing with the rest of PC API configuration.

PC API currently only supports lan (network) access. Hewlett Packard has a network package called NS/ARPA which can be used for either TCP/IP or ARPA services. The latest version of NS/ARPA is 2.1. The software is usually installed either standalone under the \HPNET directory, or can be used in conjunction with Microsoft's LanManager, and would be underneath the \LANMAN.DOS directory.

Other examples of products being used today are Walker Richer Quinn and the WRQ 3000 Connection product which contains the Sockets network interface. The WINSOCK connect type is also available. PC API should work with any WINSOCK 1.1 compliant software. The WINSOCK marketplace is rapidly expanding the available number of software products that are WINSOCK 1.1 compliant. PC API will support the NetWare connection as well, running the SPX/IPX protocol stack, with version A.F0.16 of PC API. LAN WorkPlace for DOS is a protocol that can work with Novell NetWare to provide the TCP/IP interface. A few of the considerations for some of the different PC network packages are listed in Appendix A.

2.5 PC API Scripts & SCRIPTOR

The next step is to begin testing the configuration that is established in the SQL.INI configuration file. The method to do this is through a utility that HP provides with the PC API software called SCRIPTOR.EXE. The program group that is created when PC API is installed includes a scriptor icon that can be executed, otherwise the utility can be executed through file manager in Windows by double clicking the scriptor.exe file. The SCRIPT name is entered on the first line.

VERIFY.SCP and VIEWS.SCP are the most important scripts when working with PC API. VERIFY.SCP is a very simple file that tries to use the configuration information from SQL.INI to attempt establishing a connection to the dbenvironment that is configured. The VERIFY.SCP script attempts to use the server name SERVER1 and the user code SYSADM. If the configured values for these parameters need to be changed, alter the VERIFY.SCP file accordingly.

VIEWS.SCP is a script that installs views into the dbenvironment. The views need to be present for 4gls to be successful accessing through PC API. An ascii file named VIEWS.SQL is provided with the PC API software for use on the server. Executing VIEWS.SQL has the exact same effect as using SCRIPTOR.EXE to execute the VIEWS.SCP script. Connect to the dbenvironment on the server through ISQL and then use the start command to execute the VIEWS script. If the VIEWS script is installed on a dbenvironment multiple times, the views will also be installed multiple times. In order to remove views from the dbenvironment, the scripts UNVIEWS.SCP and UNVIEWS.SQL have been provided.

If all goes well after running the VERIFY.SCP script, the returned message states that the connection attempt was successful. If the connection attempt is successful, then it is possible to begin working with the 4gl. If the connection attempt is unsuccessful, then it becomes a method of going back and investigating the individual pieces again to trouble-shoot the problem. The best thing to do is to isolate the pieces that are involved to determine where the problem lies. PC API filter trace functionality can be used to trace the actual PC API intrinsic calls that are made. Another source of information immediately available when an error message is generated, is to open the router icon that appears at the lower lefthand part of the Windows screen. An icon is popped up when going through the layer of PC API that loads the allbase router. If the error in the router icon has been addressed, close the router by single clicking on the icon, and choose close from the menu. This is necessary since the router does not automatically clear itself after an error, and further testing with VERIFY.SCP in SCRIPTOR could bear false results.

2.6 PC API Tracing & Error Facilities

The instructions for initiating the filter trace are in the README file that comes as part of the PC API software. Refer to Appendix D for additional information on the trace facilities that are available to use.

Identifying the error message that is returned from a bad connect attempt is an effective tool. The error message is returned to the router icon, that can be brought up by double clicking.

Error messages are a good source of information that can then be used to refer to either the ALLBASE/SQL Message Manual, or the HP ALLBASE/SQL PC API User's Guide. In some instances the returned error code from the 4gl is 2559. 2559 is a standard code which is returned, and means that the 4gl application has not coded to the SQLXERR intrinsic for PC API. Check with the vendor for advice when the 2559 error is generated by the 4gl application. When the error returned does not appear to be an ALLBASE/SQL message, some information can be gathered from the message catalog file that PC API includes. ERROR.SQL is the message catalog file, and can be viewed with WRITE.EXE, a standard edit utility that comes with Windows.

3.0 ODBC

The Open Database Connectivity standard, referred to as ODBC, was written by Microsoft Corporation. Providing an open systems approach gives a more fluid approach when vendors are coding interfaces for 4gl tools. If ODBC works correctly, this means that vendors can code 4gls without worrying about the communication to multiple vendor platforms. The complete documentation of ODBC is part of a manual that is referred to as the Software Developer's Kit, SDK, and can be purchased from Microsoft.

3.1 PC API ODBC Installation & Versions

PC API ODBC is very similar in setup to PC API Gupta as far as the initial installation. The SETUP.EXE utility is used to install the PC API ODBC product. The software is installed under the \WINDOWS and WINDOW\SYSTEM directories. The current version of PC API ODBC is F0.15 for ALLBASE/SQL version F.0. The current version of PC API ODBC is G0.03 for ALLBASE/SQL version G.0. Use the WHAT.EXE utility program to verify the version of PC API ODBC installed:

```
C:\WINDOWS\SYSTEM>what allbase.dll
allbase.dll: @<#> HP ALLBASE/SQL PC API Library. Version: A.F0.33
@<#> <c>COPYRIGHT Hewlett-Packard Co. 1994. All rights reserved.
@<#> HP ALLBASE/SQL ODBC Driver. Version: X.F0.15
@<#> <c>COPYRIGHT Hewlett-Packard Co. 1994. All rights reserved.
```

3.2 PC API ODBC Configuration & Views

The main configuration file used with PC API ODBC is the ODBC.INI file in the WINDOWS directory. ODBC.INI contains the information on configuration sections for PC API ODBC, as similarly in the SQL.INI file for PC API Gupta. Refer to Appendix C for an example ODBC.INI file.

ODBC also needs to have views installed into the dbenvironment to be successful with accessing through 4gls. ODBCVIEW is an ascii file that must be downloaded to the HP 3000 or HP 9000 and used as a script file through ISQL. Connect to the dbenvironment through ISQL and use the start command to install the views. It is okay to have both ODBC views and PC API views installed in a dbenvironment at the same time. ODBC is different from PC API because each time the ODBCVIEW script is executed, it attempts to deinstall the ODBC views whether they exist or not. Thus an "UNVIEWS" script for ODBC is unnecessary. The ODBC tool has a utility called the ODBC administrator which configures the different connection sections in the ODBC.INI file. It is up to the user to configure the system and logon information. This is done by adding the LastUser line into the ODBC.INI file after the appropriate section:

```
LastUser=#mpeix/hp:dbe.group.acct,network#odbc,user/pass.acct/pass  
or  
LastUser=#hpux/hp:/user/rjl/dbe,network#user:pass
```

Obviously it will be necessary to configure the correct information in the LastUser configuration string specific to the environment.

As with PC API Gupta, the same rules apply to the configuration of a HOSTS file. Specify only the nodename in the HOSTS file, and do not append the domain.organization. ODBC is fast becoming a widely used protocol and standard for the client/server industry. PC API ODBC is available for use on 4.0 systems with the F.0 version of ALLBASE/SQL in beta format. The version of ALLBASE/SQL necessary is F0.59 or higher. When comparing the PC API Gupta and PC API ODBC interfaces, most of the concepts are very similar.

3.3 PC API ODBC Networking Information

The network specification that appears in the LastUser configuration can be one of the following:

- >ip - W3IPC.DLL (NetIPC, HP 3000 servers only)
- >sk - WSOCKETS.DLL
- >nw - NetWare (HP 3000 servers only)
- >ws - WINSOCK.DLL
- >lw - WLIBSOCK.DLL (Novell LAN WorkPlace)

Network interfaces that are currently supported with PC API ODBC are TCP/IP, SOCKETS, NetWare, WINSOCK, and LAN WorkPlace for DOS. The PC API ODBC interface currently works with an increasing number of 4gls/report writers that include Microsoft Access, Powerbuilder, Information Access, Impromptu,

Q+E, Forest & Trees and many others that are continuing to join this market. Refer to Appendix A for additional information on networking.

4.0 IMAGE/SQL

4.1 Version Information

If IMAGE/SQL is being used, the version of TurboIMAGE/XL should be C.04.19 or higher. The TurboIMAGE/XL version is found by running the program QUERY.PUB.SYS and executing the command VERSION. Check for the following lines in the output:

```
TURBOIMAGE overall VUF:  
HP30391C.04.nn TurboIMAGE {nn represents version number}
```

The version of IMAGE/SQL should be at least B.F0.24. To verify the version of IMAGE/SQL, check the banner information when running IMAGE/SQL:

```
HP36385B B.F0.nn IMAGE/SQL Utility {nn represents version number}
```

4.2 Setting Up & Testing IMAGE/SQL

The following procedure can be used to test IMAGE/SQL with PC API. The IMAGE/SQL utility/interface is easy to use and provides the link between TurboIMAGE/XL and ALLBASE/SQL. To test the interface without using an existing database, HP provides a utility program IMSQL.SAMPLEDB.SYS on the HP 3000. This program is a XEQ file, and allows the user to choose from a menu to set an IMAGE/SQL dbenvironment. See Appendix E for an example and brief instructions on how to use IMSQL.SAMPLEDB.SYS.

The IMAGE/SQL utility can create the dbenvironment upon request, thus making the process a lot simpler to get up and running. The following displays an example of attaching a TurboIMAGE/XL database to a new dbenvironment:

```
MPE/iX:imagesql  
HP36385B B.F0.24 IMAGE/SQL Utility  
(C) COPYRIGHT HEWLETT-PACKARD COMPANY 1993  
>>set turbodb music  
>>set sqldb musicdb  
DBE does not exist, do you want to create one? [Y/N] : Y  
>>exit
```

Once this is complete, use the previous information on the server and client requirements for PC API to try and connect to your new MUSICDBE.

4.3 General Information

When working in a production environment with IMAGE/SQL, always make sure that prior to making changes to either side of the environment that the detach process has occurred. The DETACH command is defined in the HP IMAGE/SQL Administration Guide. Note that the version of IMAGE/SQL is B.F0.24, and is displayed in the banner information above. See Appendix F for additional reference material that may help when working with the IMAGE/SQL and PC API environment.

5.0 SUMMARY

PC API has been key to evolving HP's client/server relational business and meeting the key industry needs of today's challenging MIS environments. Understanding the key components of PC API can help when evaluating business needs. As time passes, it is more and more critical to be aware of client/server technology, and understanding the benefits of such an environment. Client/server will continue to evolve and be a major factor in the way business is done in the future.

Appendix A - PC Network Considerations

NS/ARPA:

- The NS/ARPA version should be 2.1 and can be verified by checking the diskettes.
- If the HOSTS file does not exist, create it under the appropriate directory C:\LANMAN.DOS\ETC\HOSTS or C:\HPNET\ETC\HOSTS
- PING is a utility which is used to test the TCP/IP or Sockets connection from the client (PC) to the server (HP 3000 or HP 9000).
- Failure of the PING test indicates a communication problem. If failure occurs, check the software configuration on the PC again. Check any cabling, lan connections, or server network conditions that might contribute to the failure.

WRQ 3000 Connection:

- The HOSTS file is configured under the directory \WRQNET.
- WRQ has a PING facility to test the lan connection.
- Make sure that you are using WRQ 3000 Connection with PC API. The standard WRQ Reflection product only provides VT capability.
- Verify the version of WRQ 3000 Connection is 2.10 or higher.

NetWare:

- The version of NetWare should be at least 3.11.
- NVER is a utility that can be used to tell what version is installed.
- If you are trying to use NetWare, and receive a DATACOMM 250 error follow instructions in the README file provided with PC API to resolve this error.
- A key point is that NetWare allows for use of dual stacks. If this is the case, a TCP/IP protocol stack can be run in conjunction with the SPX/IPX stack provided by NetWare.
- If using the SPX/IPX stack, this requires that both NetWare/IX and HP ThinLAN 3000/iX be present on the HP 3000 server.
- NetWare is used only with the HP 3000.

Novell LAN Workplace for DOS:

- Can be used starting with version A.F0.17 of PC API.
- Version should be 4.1 or higher.
- Version 4.1 requires patch LWP168.EXE. This can be obtained from Novell or downloaded from the CompuServe NETWIRE forum library.

WINSOCK:

- PC API Gupta or PC API ODBC should be able to communicate with any PC networking product that is WINSOCK 1.1 compliant.
Working with HP ALLBASE/SQL PC API

Appendix A - PC Network Considerations (Part 2)

The following table provides a summary of the PC Library files along with the corresponding HP and third party networking products. Note that the PC network libraries are distributed as a component of the networking products and are not a component of PC API.

PC Library	HP 3000 Server	HP 9000 Server
WLIBSOCK.DLL	Novell LAN WorkPlace 4.1	Novell LAN WorkPlace 4.1
NetWare	NetWare 3.11	
WINSOCK.DLL	FTP PC TCP/IP NetManage Chameleon Microsoft Walker Richer Quinn	FTP PC TCP/IP NetManage Chameleon Microsoft Walker Richer Quinn
WSOCKETS.DLL	HP ARPA/NS 2.1(J2246A) Microsoft Walker Richer Quinn	HP ARPA/NS 2.1(J2246A) Microsoft Walker Richer Quinn
W3IPC.DLL	HP ARPA/NS 2.1(J2246A) Walker Richer Quinn	

The above Appendix Part 2 information was gathered in part from the HP PC API User's Guide for ALLBASE/SQL and IMAGE/SQL, HP Part No. 36216-90104.

Appendix B - Sample SQL.INI Configuration File

```
; (C) Copyright Hewlett-Packard Co. 1993
; Sample sql.ini file for the HP ALLBASE/SQL PC API
; patterned from
; gupta technologies, inc. 1990
; sqlbase initialization file
[dbrouter]
heap=40000
[dbwindow]
cache=50
dbdir=C:\SQLBASE
[sqlrtrw]
retry=1
[winclient.dll]
comdll=sqlawin
[winclient.nbiow]
RetryTimeout=10
[dbdfault]
DEFAULTDATABASE=DEMO
DEFAULTUSER=SYSADM
DEFAULTPASSWORD=SYSADM
[winservr]
dbname=demo
[ALLBASE]
mpedbname=server1,mpexl:partsdbe.sql.losch
mpeuser=sysadm,pcapi,roy/pass.losch/pass,sql
@(#)Client-Server ALLBASE SQL.INI. Version A.F0.29
```

Appendix C - Sample ODBC.INI Configuration File

```
[ODBC Data Sources]
partsdbe_mpeix=HP ALLBASE/SQL

[partsdbe_mpeix]
Driver=C:\WINDOWS\SYSTEM\allbase.dll
Description=Connection to MPEXL for Partsdbe sample
LastUser=#mpeix/mpexl:partsdbe.sql.losch,sk#odbc,roy/pass.losch/pass,sql
DefaultIsolation=RR
```

Appendix D - PC API Tracing Facilities

The instructions for initiating the filter trace are in the README file that comes as part of the PC API software, but to reiterate here are the steps that need to be taken:

1. Exit Windows, and cd to the \ALLBASE directory.
2. Rename SQLAPIW.DLL to SQLAPIWD.LL, this is to rename the driver out of the way.
3. Rename FILTER.DLL to SQLAPIW.DLL to enable tracing.
4. Start Windows again and retry the VERIFY.SCP script or 4gl application.
5. The allbase router will appear, and you should be prompted that the trace is active.
6. After generating the error, look in either the \ALLBASE directory or the application directory on the client for the file TRLT.SCP. This file contains the results of the PC API access.

The TRLT.SCP file which is created is a script file, just like VERIFY.SCP or VIEWS.SCP. Using the TRLT.SCP file as a script file, is a key tool in some instances to try and reproduce problems with SCRIPTOR outside of the application 4gl. The TRLT.SCP file can be used as a script file on the first line of the SCRIPTOR utility.

It is sometimes necessary to get a datacomm trace on the PC. Datacomm tracing is enabled by setting on HPDCRECORD outside Windows at a dos prompt:

```
C:>SET HPDCRECORD=1 {enables tracing}
```

Any space in the string "HPDCRECORD=1" will cause the command to fail. Displlog can be used to convert the log file, which is in binary output format, to ascii format:

```
displlog hpdc.log output.log
```

To disable the datacomm trace issue the following command

```
C:>SET HPDCRECORD=0
```

On the HP 3000 NS logging can be checked by creating the file NSLG on the logon group and account where the dbenvironment will be accessed. This will enable logging from the server side.

Appendix E - Using IMSQL.SAMPLEDB.SYS

MPE/iX:IMSQL.SAMPLEDB.SYS

Options for Setting Up IMAGE/SQL Sample DBEnvironments

Choose one:

1. Create Music IMAGE database
 2. Display Music schema
 3. Attach Music to MusicDBE
 4. Display schema for MusicDBE
 5. Purge Music
 6. Purge MusicDBE
 7. Help
 0. Exit
-
-

Enter your choice=>

Here are the steps to follow to create a test TurboIMAGE/XL database, and then a dbenvironment that can be attached:

1. Choose option 1 which creates a TurboIMAGE/XL database MUSIC.
2. Choose option 3 which creates an ALLBASE/SQL dbenvironment named MUSICDBE, and attach the Music TurboIMAGE/XL database.

Feel free to test all of the different functions of the IMSQL program. Attaching is the process of mapping the TurboIMAGE/XL database structure into the ALLBASE/SQL dbenvironment.

Appendix F - Supporting Documentation

When the support upgrade for IMAGE/SQL is purchased, at this time a kit is provided that contains the IMAGE/SQL software along with a number of other items. Some of the manuals and their part numbers included in that kit are as follows:

Getting Started with HP IMAGE/SQL	36385-90008
IMAGE/SQL Administration Guide	36385-90001
HP ALLBASE/SQL PC API User's Guide	B2463-90001

The following manuals would be good to have on hand when working with the HP ALLBASE/SQL PC API environment:

ALLBASE/SQL Reference Manual	36216-90001
ALLBASE/SQL Database Administration Guide	36216-90005
ALLBASE/SQL Message Manual	36216-90009
ALLBASE/ISQL Reference Manual	36216-90004
HP PC API User's Guide for ALLBASE/SQL and IMAGE/SQL (900 Series HP 3000)	36216-90104
HP PC API User's Guide for ALLBASE/SQL (HP 9000)	36217-90187

THE DATABASE ADMINISTRATOR: A ruthless juggler of Reality, Algorithms and Data Structures

F. Alfredo Rego

Adager

Sun Valley, Idaho 83353-3000

(208) 726-9100

Abstract

The Database Administrator (DBA) must keep in perfect balance (emotionally and otherwise) while flawlessly maintaining three dynamically-changing computer-model "eggs" whirling through the air at all times:

1. The "Reality" that the computer model is supposed to reflect;
2. The Algorithms (programs) that manipulate the computer model to produce "results";
3. The Data Structures that support the computer model.

I would like to conduct this session as an open forum with lots of questions and cross-firing aimed at the DBA's juggling task and its awesome implications. Therefore, I will present in this paper only SOME of my opinions on certain topics:

- a. The painful choices involved in the selection and maintenance of the appropriate kinds, shapes, weights, and sizes of computer-model "eggs".
- b. The smashing catastrophes hatched when the computer-model "eggs" happen to be unbalanced.
- c. The performance-time adjustments that the DBA might wish to make to get the act together should one of the computer-model "eggs" get out of hand, break, hatch, or whatever.
- d. The help (or lack thereof) that the DBA might run into. In particular:
 - The tools available (if any) to manipulate reality.
 - The tools available (if any) to manipulate algorithms.
 - The tools available to manipulate data structures. And, specifically for IMAGE/3000, TurboIMAGE and IMAGE/SQL Databases and Adager.

The DBA is the key person in any computer model, regardless of the model's degree of complexity (which can range from a simple game written in 10 lines of BASIC to a very complicated satellite-tracking system).

Before going any further, let me say that I know at least two kinds of DBA's:

1. DE-JURE DBA's, who have been formally and officially appointed, blessed and fully authorized (certified?)
2. DE-FACTO DBA's, who just happen to do the work without any pompous formal titles, positions or certifications (somewhat along the lines of Charles Lindbergh, who flew from New York to Paris in 1927 even though he did not have a Visa!)

Hopefully, the de-jure DBA is the de-facto DBA in your organization. However, should the case be different, then please note that all my references to "the DBA" should be interpreted as references to "the de-facto DBA".

Traditionally, The DBA's position has been linked to the rather misunderstood task of "maintaining the database." I say "misunderstood" because the emphasis has been mainly on the data-structure aspect of the database. Two other fundamental aspects have usually been slighted: the reality the model is trying to reflect, and the algorithms that manipulate the model to produce the results actually desired from the computer model itself.

In this regard, the DBA is faced with unenviable choices:

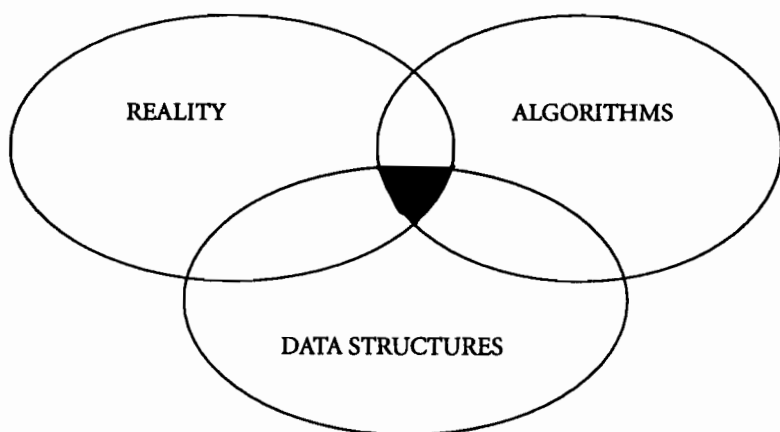
- Which aspects of reality are going to be incorporated into the computer model, and which are going to be left out?
- Which algorithms are going to be allowed to manipulate the model, and which are going to be left out?
- Which data structures are going to be allowed to support the model, and which are going to be left out?

The painful part in these choices comes from the fact that what is left out of the computer model is as important, for the model's effectiveness, as what is actually put into the model itself. And the DBA faces some very frustrating limitations:

1. It is impossible to reflect more than a tiny fraction of reality (ever comprehended or to be comprehended) in the model. For instance: you may include an employee's birth date in your personnel system; you do not include the employee's TIME of birth to the millisecond... although the employee's astrologer might! (As a matter of fact, most of reality is ruthlessly left out of any computer model.)

2. It is impossible to use more than a tiny fraction of all algorithms (ever devised or to be devised) to manipulate the model. For instance: You may use one or two payroll programs for your personnel system; you do not use all the payroll programs written for the HP3000... let alone the payroll programs written for all computers! (As a matter of fact, most algorithms are ruthlessly left out of any computer model.)
3. It is impossible to use more than a tiny fraction of the data structures (ever devised or to be devised) to build the model. For instance: you may use sequential MPE files, direct-access MPE files, KSAM files, IMAGE/SQL databases, VISAM, IMS, ADABAS, ORACLE, SYBASE, INFORMIX, INGRES, DB2, dBASE, FOXPRO, etc. Quite mind-boggling, isn't it?

Graphically, then, a painfully-built computer model may be represented by a Venn diagram that depicts a (rather small) intersection of three (rather gigantic) sets:



The RATHER SMALL INTERSECTION (almost too tiny to even be seen in the drawing above), with the ruthless selection and maintenance of an appropriate mix of its three fundamental components, is the DBA's most important concern.

And even though the costs involved (monetary, emotional and otherwise) may be quite high, the DBA cannot afford to hatch any of the common catastrophes that

inevitably happen when the DBA ceases to be ruthlessly faithful to the model and develops a soft spot for one of the elements at the expense of the others. For instance:

Excessive reality and insufficient model.

Some government agencies, for example, would just love to have all kinds of data on you, like the phase of Venus when you were born and other (apparently irrelevant) bits and pieces that don't mean anything by themselves to the untrained eye, but show a sophisticated, strong correlation coefficient with all your Income-Tax returns of the last 10 years pointing to the 47.5783124% probability that you might be reporting less income and more expenses than you really should.

A very nice and worthy project, indeed. And perhaps by the time Venus has joined the Earth in a single orbit we will have the computation (and storage) capabilities required by this rather ambitious "chunk" of reality.

Excessive algorithms and insufficient model.

Some programmers I know, for example, would just love to sit down at a terminal and create all kinds of really neat programs that are always on the borderline of "just one more compilation" before they are elegant enough to solve all your problems.

A very nice and worthy project, indeed. And perhaps by the time you have used your good old pencil to do the rather trivial calculations required, you will have the right problem for the programmer's "solution."

Excessive data structures and insufficient reality and algorithms to justify them.

Some people I know, for instance, have built complex combinations of IMAGE databases, KSAM files and several kinds of MPE files to build computer models which, I believe, could have been better left as well-organized note pads in their pockets.

Assuming the DBA has, somehow or another, made the initial design choices, we come up with another set of unenviable decisions when it comes to maintaining the computer model.

Which aspects of reality are going to be altered to reflect limitations or changes in the algorithms and data structures?

Examples of limitations in the algorithms and data structures: double-integer variables cannot hold commercial values bigger than twenty-one million Dollars and Cents (or so), but their algorithmic arithmetic is faster than packed-decimal variables, which can hold practically any commercial and super-commercial value

(and take more disc space... a rather vital concern if you want to hold all 260 million Americans or so in your database!) IMAGE/SQL databases can have, at most, 1023 data items; datasets can have, at most, 255 fields, 2 billion (or so) entries, and 16 paths; data entries can have, at most, 4094 8-bit bytes.

Examples of changes in the algorithms and data structures: the unheard-of discovery of a bug (yes, believe it or not; A BUG!) in some vital program. Even though the algorithm per se does not change, or perhaps changes only slightly, when we correct the bug, we know that we have a 50/50 chance of introducing some rather unexpected "secondary" consequences (and, unfortunately, the same unpleasant rule holds for new enhancements!)

Examples of changes in the interfaces between algorithms and data structures: changes in a dataset's security arrangements that will cause a totally different "virtual data entry" to be passed to the buffers of the innocent algorithms! (Please note that neither the data structures, per se, or the algorithms have changed. But awfully catastrophic results would be almost guaranteed!)

REALITY-ALTERING TOOLS: Besides certain well-known drugs (and some that are not so well-known, as well) there are few, if any, tools to help the DBA select and transform reality.

Two references come to my mind: (1) Dr. Murphy, the famous "mathematician", did some work on fitting the universe to the equation (His methods are in the common domain.) (2) God, the Almighty, did rather extensive work on radical transformations of reality. This pioneering work, although well-documented, is unfortunately proprietary and you might not be able to get a license (if you are interested, try the book of Genesis).

Which algorithms are to be altered to reflect limitations in reality or data structures?

This topic deserves a special note: computer-related activities have been heavily biased towards algorithms. ("Computer programming" is a common expression that reflects this bias. Whoever heard of "computer data structuring"?) Nevertheless, we have not developed good ways to transform algorithms!

Doesn't this remind you of the shoemaker's children going barefoot?

ALGORITHM-ALTERING TOOLS: There are few tools to help the DBA select and transform algorithms. There is a lot of hype, though!

Which data structures are to be altered to reflect limitations or changes in reality or algorithms?

Please note that the main concern here is the preservation of the information which

currently resides in the data structure. If the data structure is empty or if you do not care about preserving its information, then there is no point in worrying about transforming it at all! You just destroy the current structure, create a new one, and manually fill it with the appropriate new information. Very simple, indeed (unless, of course, you happen to have 100,000 customers and 250,000 spare parts to re-key!)

DATA-STRUCTURE-ALTERING TOOLS: A tremendous amount of work has been done on this topic.

For sanity's sake, I am going to restrict myself to one specific kind of data structures during this session: the data structures used by IMAGE/3000, TurboIMAGE and IMAGE/SQL, Hewlett-Packard's award-winning Database Management Systems.

I will analyze, briefly, three categories of data-structure-altering tools for IMAGE. The first category includes MANUFACTURER-supplied tools. The second category includes USER-supplied tools. And the third category includes INDEPENDENT-SOFTWARE-FIRMS-supplied tools. Obviously, I cannot mention ALL the available tools in any of these categories. Please do not feel that, if your favorite tool is missing from my analysis (or if your most hated tool IS INCLUDED!), I am implying that it is not good (or bad, whatever the case may be...).

DBUNLOAD/DBLOAD

HP gives you system tools, like DBUNLOAD and DBLOAD, which allow you to do certain transformations. Specifically, as written in the IMAGE/3000 reference manual:

"It is possible to make certain changes to the design of an existing database without having to write special programs to transfer data from the old database to the new one. The general sequence of operations which you use to do this is:

1. Run DBUNLOAD on the old database, copying all the data entries to tape or serial disc.
2. Purge the old database using DBUTIL,PURGE.
3. Redefine the database using the same database name and create a new root file with the Schema Processor.
4. Use the DBUTIL CREATE command to create and initialize the datasets of the new database.
5. Run DBLOAD on the new database using the tape or serial disc created in step 1 to put the old data into the new base.

DESIGN CHANGES

The database design changes for which the above procedure functions correctly are limited. Schema changes that yield correctly transformed databases fall into two general categories: those which always result in a good transformation and those which are legitimate only in some circumstances.

Any of the following schema changes, alone or combined, which are acceptable to the Schema Processor will always result in a successfully transformed database:

- Adding, changing, or deleting passwords and user class numbers.
- Changing a data item or dataset name and all references to it.
- Changing data item or dataset read and write class lists.
- Adding new data item definitions.
- Removing or changing definitions of unreferenced data items.
- Increasing dataset capacities.
- Adding, deleting, or changing sort item designators.
- Changing primary paths.
- Adding new data items to the original end of a data entry definition.
- Removing data items from the original end of a data entry definition.
- Changing an automatic master to a manual master or vice versa.

Other schema changes may or may not be legitimate. A potential change must be judged in light of the particular database and the functioning of DBUNLOAD and DBLOAD, described later in this section. Basically, all entries from an old dataset are put into the new dataset with the same number, except that no entries are directly put into automatic masters. The entries are truncated or padded with zeroes as necessary to fit the new dataset's entry length. DBUNLOAD and DBLOAD always handle full entries, without regard to item positions or lengths. If the new dataset's entry is defined with the items in a different order than the old dataset, DBLOAD will not fail, but the dataset content may nevertheless be invalid. For example, data of type real may now occupy the position of a character type item.

In some circumstances, DBLOAD will fail. For example, if a dataset's capacity has been reduced in the new database to a number less than the number of that dataset's entries on the tape or serial disc."

Interex Contributed-Library.

Certain sophisticated users of IMAGE databases (like Linford Hackman of Vydec, in New Jersey) had transformation requirements that could not be satisfied by DBUNLOAD/DBLOAD, as stated in the IMAGE manual quote of the previous section. Linford's contribution was therefore born and generously placed in the HP3000 contributed library for your benefit: DBREBILD. This is a general-purpose program which transfers information from the old database to the new database. DBREBILD is smart enough to "know" both data structures, so it can map the old data onto the new database. Therefore, it saves you the efforts involved in writing your own special-purpose transfer algorithms.

DBREBILD does not allow you to do data-type changes that preserve the information of the data items, though. For instance, if you change a data item from P4 to J1, and if you want to keep your old information, then you must still write your own specialized application program in COBOL (or whatever) to do the "moves". But DBREBILD allows you to do all kinds of data-entry reshuffling and other very useful transformations.

BEGIN PARENTHESIS-----

Disadvantages caused by the use of magnetic tapes (or serial discs, for that matter!)

DBUNLOAD/DBLOAD and DBREBILD are very useful and reliable. But they share a minor technical detail as a catch: the time it takes to unload the whole database to magnetic tape, not to mention the time it then takes to reload it back.

1. Monetary disadvantages. If you take everything into account, I would not be surprised if your DAILY operating costs come to several thousand dollars, or francs, or whatever (do not forget to include equipment depreciation, employee fringe benefits, insurance, taxes, duties, software maintenance contracts, application software costs, staff, office furnishings, office supplies, building space, air conditioning, electricity, doctor's bills for your ulcers, alimony for your impending divorce due to all the time you spend at the computer center..., etc.)
2. Absolute reliance on basically unreliable entities. The physical risk of the unload/load cycle can be very significant, also. You are counting on absolutely perfect magnetic tapes which will perform flawlessly. And you are counting on a perfectly-maintained power supply from your friendly electrical company. Should anything (anything at all) go wrong, you have to begin all over again. You have no choice. Period.

And are you going to forbid that your operator take breaks during this long

(and, let's be honest: BORING) process, encouraging perhaps the occurrence of more human errors than you can afford?

3. Unavailability of the database during the conversion process. Remember: the whole database is absolutely unavailable for any use whatsoever during all this time. Can you really afford to wait until that long weekend comes along? Can you really afford to close shop for a couple of days?

It turns out that, for any truly on-line application, this is the HIGHEST cost of all: whatever it costs your organization to have the database locked out for housekeeping instead of available to its legitimate users.

END PARENTHESIS -----

Tony Pierce (from BOEING, in Seattle) had a brilliant idea: Why not forget the whole concept of the "old database" that has to be transferred to the "new database"? Why not restrict the transformation space to its proper dimensions? If he was going to change the capacity of just one dataset, why bother (and risk) moving all other 98 datasets around just for the ride?

DBCHANGE (now renamed to DBCHG in the contributed library is Tony's contribution. This very nice program changes the capacities of datasets in a clever way. DBCHANGE does indeed transfer information from the "old dataset" to the "new dataset" while preserving chain information and hashing properties. It does this while leaving the rest of the database untouched. And, MOST IMPORTANT OF ALL, it does not use magnetic tapes (or serial discs) at all.

Just to give you an idea of the implications of Tony's contribution (which is also available in the HP3000 contributed library for your convenience), let me give you a typical example: if all you do is change the capacity of one dataset, you may reduce your transformation time from three days (assuming a medium-size database with detail datasets that have a few paths in them) to three hours (or even eight hours, if you please! Who cares anymore about small change?) Quite dramatic. Unquestionable.

Adager.

The tool I happen to be most familiar with is the tool developed and marketed by my own software firm. Therefore, I will restrict myself to Adager (The Adapter/Manager for IMAGE/3000, TurboIMAGE and IMAGE/SQL Databases). I hope other tools will be brought to everybody's attention during this session, as well as in the vendor exhibition and in other sessions. You, the HP3000 computer user, are the one to benefit the most from a wide selection of tools. And we, the tool manufacturers, are the ones to benefit from the combined experiences of all parties

involved in these endeavors, since we rely on real-life cases that indicate, with de-facto authority, WHICH tools are actually required.

DATABASE.UTILITY: In November 1978, at the HP3000 International Users Meeting in Denver, I presented a paper that mentioned two kinds of topics: (1) the questions I asked myself about the tools I had currently available to me (Unfortunately, I did not know about DBREBUILD or DBCHANGE at the time.) and (2) the answers I had developed. In a way, I may say that I re-invented the wheel in Guatemala, since Linford Hackman and Tony Pierce had already published their work. But then it turned out that I had "ridden" this "wheel" a little farther away into some other previously-forbidden territories: AND WITHOUT USING MAGNETIC TAPES (OR SERIAL DISCS...) at all!

My "wheel" was really more of a unicycle. It worked well, but it had to be ridden by a skilled acrobat of sorts. Not really fit for grandmothers, yet. (Because of the "uncomfortable seating arrangements..., because of the quick reflexes required..., because of the rather involved and laborious manual processes still required by "DBEDIT", my general-purpose root-file "editor"..., because of... well... because of the fact that it was some sort of academic prototype!)

That was November 1978. During the intervening time, Adager grew out of DATABASE.UTILITY as an independent software product in its own right. Adager has become more like a modern automobile (even grandmothers can "drive" it safely and comfortably now!)

The general idea is very simple: In my Christmas card of 1978, I mentioned that a database transformation involves two distinct (equally delicate) operations: a genetic change in the root file and a cosmetic adjustment in the datasets. Little did I know that the next decade of my life would be dedicated to learning and implementing the painstaking technologies required by such a simple concept.

To appreciate the issues involved, let's look at a familiar example. As unbelievable as it may seem, some people with straight hair would prefer wavy hair (and some people with wavy hair would prefer straight hair). If you doubt the validity of this observation, just go to your neighborhood beauty parlor! The beauty salon provides cosmetic changes to hair that already exists. Unfortunately, the genetics remain unchanged and any new hair will stubbornly come out looking as it always did (a little more damaged, perhaps, but still basically the same).

In human terms, cosmetic engineering is "mature", whereas genetic engineering is a new (and somewhat frightening) area of concern. The opposite is true in database genetics and cosmetics. The database genes, kept in the root file, are trivial to modify (the obvious thing to do is to edit a schema and run DBSCHEMA to create a new

root file). The true challenge involves the cosmetics: How do you synchronize the existing datasets to the new root file? How do you map millions of entries from the old datasets to the newly transformed datasets in an efficient way? You cannot afford to have your database "down for maintenance" for too long. Therefore, you **MUST** do the required cosmetic changes as fast as possible.

Cosmetic changes in a database require the mapping of datasets. There are many ways to map datasets within the constraints of the HP3000. How does Adager transform a database? By means of two internal data structures which model the current and original states of the database. At the beginning, the "current" state is identical to the "original" state. As you specify changes, the current state evolves.

At any time during the specification phase, you may ask Adager to apply the changes or you may continue specifying new changes. If and when you decide to apply the changes, Adager will use the "original genetics" to read your data from the "original datasets" and will transform your data in such a way that the new datasets will reflect your accumulated specifications.

Adager will do its best to minimize wasteful operations. In most cases, this means that Adager will consolidate as many changes as possible while it scans a transformed dataset. After lots of trials (and errors), we found that certain operations were not very sociable and did not interact well with their fellows. Instead of spending (even more) sleepless nights trying to figure out a way to civilize them, we decided to segregate them, since they were like hermits anyway. They performed best by themselves and the other transformations performed best by themselves. The two segregated kinds of Adager functions are SOME types of "fixing" and MOST types of "transforming" operations. For instance, we found that it was a better idea to repack datasets either before or after transforming them (repacking is a certain type of "fixing", since it fixes inefficiencies in the access of entries within datasets). However, we also found that fixing broken free-entry lists in detail datasets could be trivially done while transforming the datasets. So, as frustrating as it is, there are no absolute rules!

Adager uses a Critical-Path methodology to decide which sequence of operations has the highest likelihood of minimizing the total elapsed time. If you prefer to use your own sequence, you can certainly ask Adager to do its functions one-at-a-time.

The syntax of your specifications to Adager is remarkably similar to DBSCHEMA's syntax. This means that you do not have to learn anything new: If you can specify a database through DBSCHEMA, you can maintain and modify that database through Adager. The similarity stops there, though, since Adager's response to illegal specifications is more forgiving than DBSCHEMA's and will allow you to correct illegal specifications on the spot (if you are in session mode).

Since Adager's internal data structures are implemented as STACKS, you can do some really neat things that would be impossible otherwise. For instance, you can simultaneously rename and relocate items and datasets without causing any confusion in Adager's logic and you can easily specify changes in a recursive manner, such as requesting a path to a non-existing dataset. In this case, Adager will let you specify the new dataset which, in turn, may have new fields that are not defined as items yet. Adager knows all of this and guides you through the required recursive maze. Once you clear up all these pending pieces of business, Adager returns you automatically to the task that began the whole recursive trip and you can continue on your merry way!

The genetics and cosmetics of IMAGE/3000 are slightly different from the genetics and cosmetics of TurboIMAGE and IMAGE/SQL. But the fundamental Adager approach remains the same: Adager handles all mutations with equal ease.

3015

**Getting the Benefits of Data Warehousing
Without the Warehouse**

by

Terrence O'Brien

Dynamic Information Systems Corporation
5733 Central Avenue, Boulder, Colorado 80301
(303) 444-4000

Getting the Benefits of Data Warehousing Without the Warehouse

Building Your Own Data Warehouse

It's been said that there are no really new ideas, just revisions of old ones. While I can't bring myself to agree with the premise, examples abound in technology of ideas that were proposed years ago, but are only now coming of age as a "new" idea. Data warehousing is a case in point.

IBM came up with the term years ago as a way to distinguish between the on-line databases optimized for transaction throughput, and the databases of more static data that can contain millions of records, optimized for fast searching and analysis. HP and other mini and UNIX vendors have capitalized on the renewed calls for strategic business use of data to herald the new age of "data warehousing" as the solution.

But rather than rushing head-long into our latest technological panacea, let's step back and look for a moment at why we would want a data warehouse to begin with. What are we trying to accomplish? Then let's look at the tools available to create that solution using the data and systems you now own.

Information, Now!

Marketing needs information on a weekly basis about the promotion success. Customer Service needs to have instant access to customer records. Manufacturing needs on-line access to documentation shared across the department. Sales needs daily access to telemarketing data and order fulfillment. And the Executive group needs ad hoc access to it all in summary form.

You could each write your own list of information needs based on your organization demands, but you can see the theme emerging: to run our businesses productively, we all need better, faster access to our on-line and historical corporate data. But it's not as simple as just putting in a faster "box" or the latest PC front-end query tool.

Large databases

The amount of data collected has grown enormously over the years. Databases of millions of records are commonplace. Some companies are talking in terms

of terabytes of data. Physically searching through these large databases for information, even with a screamingly fast processor, can take minutes, or even hours.

Disparate data

Corporate data is rarely stored in a consistent format throughout a company. Accounting may have its own dedicated system built around application tools it needs to do its job. Manufacturing has another. Sales and marketing another. Data can reside in different formats, different database structures, and even different hardware platforms. The promise of tying it all together has been an elusive goal for the last decade, and is still not in sight.

Ease of Access

Users are no longer satisfied with requesting a report from the MIS group and having it appear six to eight weeks later on their desk. They expect to be able to have direct access to that information from their desktops. They want it instantly, without having to know the arcane syntax of a cobol programmer's logic, and they want it in a format they can use with their point-and-click PC programs. The latest PC tools can be both a help and a hindrance. Explaining to the Senior Vice President he just can't do that kind of query on the system during working hours has been known to be a career-limiting move.

The Promise of Data Warehousing

So how does a data warehouse address these issues? If you've studied the traditional data warehouse models, almost all share the following characteristics:

1. A data warehouse separates the analytical information from the the on-line application system so that each can be maximized for its intended use: the on-line system for transaction throughput and the data warehouse for complex query access.
2. A data warehouse collects information from disparate database sources, pooling it in a system built for universal, easy access.
3. A data warehouse summarizes detail information into summary data and sets up common comparisons (pre-joined tables) that are stored to speed retrievals and analyses.

HP's Open Warehouse, IBM's Information Warehouse and DEC's Decision

Support solutions offer excellent tools to set up this type of warehouse. They offer consulting expertise to design your warehouse, as well as the hardware and the software tools to create it. In your research, I strongly recommend that you talk to experts from these organizations on their proposed solutions.

I'm now going to propose heresy. I'm going to suggest that you question the validity of each of these characteristics in evaluating your own data warehouse needs. And I'm going to outline some alternatives.

The "Virtual" Data Warehouse

On-line or Warehouse?

Let's start with the first data warehouse characteristic: separating on-line data from historical information. This information is usually split into two systems for one reason: speed. There has been no effective way to provide very fast, flexible access to data and maintain an efficient on-line system supporting one-hundred or more users. Lacking such a mechanism, the logical choice is to make them two separate systems.

But the choice is not as limited as it once was. There are now a handful of products that can provide multiple keyword, indexed access to data—the very kind of access needed from our data warehouse. One product, OMNIDEX, stakes its claim on being able to index data in its native database format, rather than requiring a proprietary data structure. How does this relate to our data warehouse? Because with the addition of multiple keyword indexes to *existing* data files, providing fast, flexible access to on-line data is now possible—perhaps negating the need for a separate data structure. Let me give an example to explain further.

Multiple keyword retrieval is the ability to search for records using selection criteria from any number of fields or tables. And the selection criteria can be information stored anywhere in the text, date or numeric string. Take your marketing manager that wants to track how his promotion is going. He may want to go into your prospect database and search for all records of people that have responded to a particular advertising campaign in the last month that have purchased the product. He would select by the promotion description, the response date and the order status (likely to be in another table). If the on-line database is at all large, a search such as this would traditionally have to be done off-line, or in a data warehouse setting. It could easily take a half an hour to an hour to execute because of the sorting and merging required against the large number of records. The addition of multiple keyword indexes would allow this

search to take place in seconds on the *on-line system*.

Enterprise-wide Information Access

A data warehouse can be a very efficient mechanism to overcome the fact that you have corporate data scattered across a variety of databases and platforms. In fact, I personally believe that this is the *most important* reason to set up a data warehouse. But I still suggest a closer look at the rationale before assuming that this is your ticket to universal access.

If your goal is to provide access to the data contained in all those systems, *and cross-reference information between them*, a data warehouse is a must. For example, if sales needs to look at customer records stored in one system, and compare them to their accounting status stored in another system on a different platform or database, the most efficient way to provide that comparison is to collect the pertinent information from both in a data warehouse and "homogenize", then index the information to be easily cross-referenced and joined.

If, however, you find that universal access really means giving your key users access to all information systems from a standard, easy-to-use format, but data from disparate systems does not need to be cross-tabulated; building a client/server access system to them all with multiple keyword indexing can be a more efficient choice. Client/server screens can be easily built with low-cost PC development tools such as Microsoft Visual Basic or Powersoft's PowerBuilder that can provide point-and-click access to each system. You can choose to use the PC SQL-based query tools and display systems such as spreadsheets and word processing systems, but I strongly recommend that you add multiple keyword indexes to each of the underlying databases, so that the queries most users want to accomplish can be supported. Polling store sales information nightly and displaying it in a graph that can be called up with a click of a mouse becomes possible. Reviewing accounts receivable by region can be done on-line. Manufacturing output by product. Almost any type of query or comparison can be supported in a client/server environment if the underlying indexes are in place to provide the fast access and summarization.

Summarizing Information

A key characteristic of most data warehouse systems is the fact that the data is usually not directly transferred to the warehouse from the operational system. It's often summarized and even pre-joined to make the queries more efficient. But this high level of granularity has a trade-off as well. First, when the database is refreshed with new data, all data must be re-summarized as well. Secondly, many of the pre-joined data segments compare "like" information: sales by sales rep by date, sales-by-customer-by-date, sales-by-product-by-year, etc. Much of the stored information is redundant. And third, these pre-joins are static. If a new comparison is needed, the data warehouse must be re-loaded with a new pre-join to support the comparison.

I'm not suggesting that you should not summarize, but if you can accomplish your access needs with a minimum of summarization, you provide the most flexible access with the least amount of overhead. Again, indexing all key values at the lowest granular level provides that flexibility.

So what have I described? I call it a "virtual" data warehouse: the benefits of a warehouse without the warehouse. Adding multiple keyword indexes on on-line data with client/server access can create a powerful enterprise-wide access system--at a fraction of the cost of conventional warehouse systems, and actually with less work.

Evaluate Your Needs Before You Buy

OMNIDEX, by Dynamic Information Systems Corporation (DISC) has been installed on systems ranging from state-wide insurance information systems to government procurement systems to catalog direct mail list systems to large manufacturing systems. Average retrievals are measured in seconds. And the average cost per system is under \$100,000, including installation and setup consulting. OMNIDEX is now available on the HP 3000, HP-UX, DEC VAX and SCO UNIX. Expressway offers similar indexing for DG and SUN.

In contrast, most data warehouse systems average in the multiple hundreds of thousands, and require a proprietary warehouse data structure for the indexed data.

If done correctly, they both provide enterprise-wide data access. They both tie-in to client/server technologies. They both take you into the leading edge of information technology.

The choice is yours.

The ODBMS Role in Distributed Client-Server Computing

Paper number #3016

By

Dr. Andrew E. Wade

**Objectivity, Inc.
301B East Evelyn Avenue
Mountain View, CA 94041
415.254.7100
drew@objy.com**

for

Interex '94

September, 1994

Abstract

Two trends in today's corporate world demand distribution: downsizing from centralized mainframe single-database environments; and wider integration, connecting finance, engineering, and manufacturing information systems for enterprise-wide modeling and operations optimization. The resulting environment consists of multiple databases, at the group level, department level, and corporate level, but with the need for dependencies among data in all of them. The solution is full distribution, providing a single logical view to objects anywhere, from anywhere. Users see a logical model of objects connected to objects, with atomic transactions and propagating methods, even though composite objects are split among multiple databases, each under separate administrative control, on multiple, heterogeneous platforms, operating systems, and network protocols. 32-bit address spaces are not sufficient for this level of integration, so Objectivity/DB is based on 64bits, providing access to millions of tera-objects, each of which may be many gigabytes. Support for production environments includes multiple schemas, which may be shared among databases or private, encrypted schemas, dynamic addition of schemas, and schema evolution. Integration must include legacy databases, such as RDBMSs, in this same transparent logical view of objects, and must cooperate with standards such as ODMG-93 and the OMG CORBA. Finally, the logical view must remain valid, and applications must continue to work, as the mapping to the physical environment changes, moving objects and databases to new platforms.

Downsizing

The first generation of client-server technology and traditional DBMSs were designed in a very different computing environment. The computing resources were all centralized in a large, centrally-administered mainframe or mini-computer (the server). Access to these resources were from remote dumb terminals (the clients), that had almost no computing power for their own. Hence, the architectures developed for this environment kept all the data storage and all the processing on the central computer, while the clients did nothing more than send in requests.

As PCs were added, graphical user interfaces (GUIs) were built into the client side, but all data storage and processing remained on the server. Thus, as more users were added, the system became slower and slower as their requests waited for one another in the central-server bottleneck (see Figure 1).

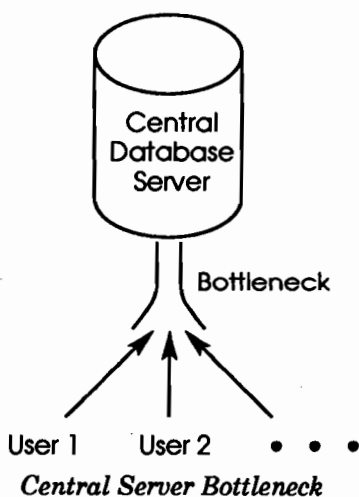
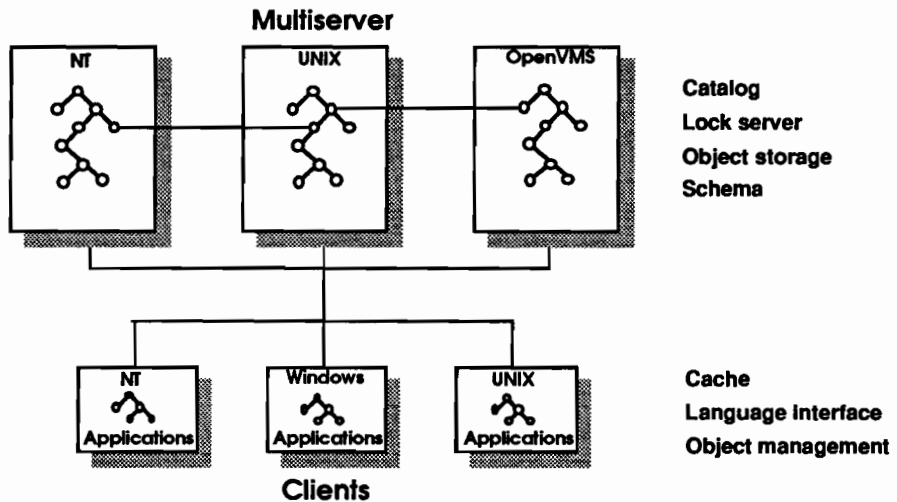


Figure 1

Today's computing environment, however, is quite different. There are still powerful servers, but there are many of them, and there are also powerful workstations on the desktop. Since the ratio of server-to-workstation power (in MIPS) is in the 3-5x range, the collection of desktop workstations represent the bulk of the computing power in the environment. Also, since these desktop machines are usually dedicated to a single user, rather than context-switching among multi-user services, they have more of their power available for that single user.

To better suit this environment, a second generation of distributed client-server technology has arisen. It allows data storage and processing to be shared across all the servers and workstations in the environment. The result is that the user sees a single logical view of objects connected to objects even though the objects reside in different databases, on different computing hardware, operating systems, and networks (see Figure 2). The DBMS manages a distributed catalogue to allow transparent location of the object when requested by name, reference, query, etc. This allows the maximum use of resources, both pre-existing resources, and new resources as they are added incrementally.



Interoperability: Single Logical View

Figure 2

Wider Integration

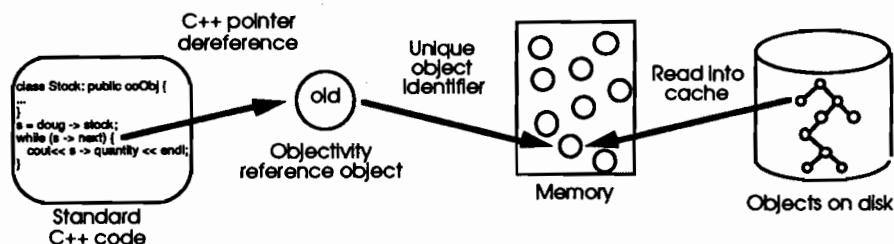
The other major trend in computing sounds almost opposite to the above downsizing. There is a need for wider integration of information resources. Instead of separate systems for order entry, human resources, engineering project management, design, and manufacturing, corporations are demanding integration across these system. For example, a new order might automatically initiate a project, reference assigned personnel, appropriate designs, and manufacturing. This move towards enterprise-wide integration allows greatly improved efficiency, automation, and the ability to model, analyze, and simulate (what-if scenarios for) the entire business.

To achieve this, the distributed client-server environment must allow multiple databases, multiple schemas, and heterogeneous computers. Each department or group, then, can create their own database, can create their own schemas, can decide who has access, etc., but can simultaneously share corporate-wide schemas, allowing their objects to connect to and be part of composite objects across the enterprise. Such composites can be accessed as single objects, with method invocations propagating among the component objects, even across databases and heterogeneous operating systems. This gives each local group the ability to work with their resources, to meet their goals with the level of autonomy they need, but still allows the increased efficiency of enterprise-wide integration.

Address Space

It seems only yesterday when 32-bit virtual-memory (VM) computing provided a major advance from previous 16- and 8-bit generations. For a single user, single-computer, local environment, 32bits or 4 GB (gigabytes) of VM is often enough. However, that size becomes tight for large applications, especially when systems use a couple of the high bits, cutting it down to a GB or half a GB, and it becomes far too little when there is a need to share information across multiple users. The advent of 64bit computers stretches this, not by a factor of 2x or 10x or 100x, but by 4 billion times, providing ample room for growth and for supporting the sharing necessary for enterprise-wide integration. With an Object Database System (ODBMS) that supports 64-bit addressing, this means 64-bits worth of objects, or millions of terra-objects ($> 10^{18}$), each of which can be any GBs.

Another address space-related issue is integrity. Pointers used in 32bit VM computing become invalid when the referenced objects move, resulting in program errors. For a single-user program, this is annoying, but the user can usually restart the program and fix it with only a small amount of information loss. However, when sharing objects across multiple users, across groups, with mission critical corporate information, pointers which are invalid (e.g., after transaction commit) would lead to database corruption. To avoid this, the object database manager can transparently insert a level of indirection in the pointer (see Figure 3). To the application, it looks the same, but now the ODBMS manager can guarantee that the reference will be safe and correct, even across commits.



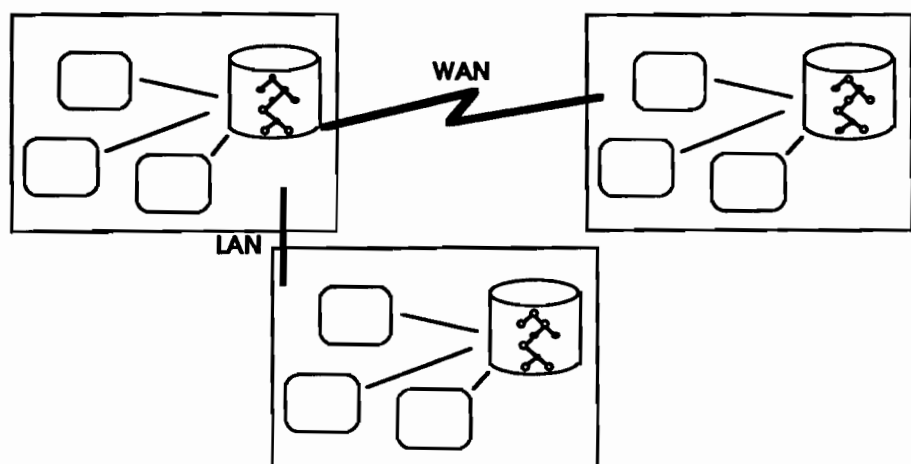
Integrity Maintained by Safe Pointers

Figure 3

Flexibility

With enterprise-wide sharing of objects, flexibility becomes more than a nicety. It becomes a necessity. This includes, for example: the ability to add and remove schemas so that groups can get their job done without requiring changes from all other groups in the organization; the ability to add new machines, with the latest price/performance, even if the hardware and operating system are different, and move some objects or databases onto those new machines, without breaking any applications; the ability to run familiar SQL and ODBC-based front-end tools; the ability to perform on-line backups without affecting applications; the ability to monitor activities, locks, detach and re-attach databases to the distributed environment, browse any objects in any database with any schema, dump objects to text language and back; etc.

When the computing environment extends beyond a single location to a Wide Area Network, the distributed ODBMS can continue to support the same single logical view and the same flexibility, but also add local autonomy (see Figure 4). In that way, when a WAN link (satellite, modem, etc.) is temporarily down, each local partition can continue to operate, with the limitation, of course, that it cannot access objects across the disconnection. This is accomplished by replicating system services, schema, catalogue, and locking information within each partition. Such partitions can be created and changed dynamically by the administrator, without breaking applications or users.



Wide Area Network (WAN) Support via Partitions

Figure 4

Standards

For such a large-scale distributed computing environment to succeed in practice, it must support a variety of common tools, languages, and users, and that requires standards. The Object Management Group's (OMG) Common Object Request Broker Architecture (CORBA) allows objects from different vendors to communicate, dispatching methods from one to the other. The Object Database Management Group's standard, ODMG 93, allows applications to work against any compliant ODBMS, and almost all vendors have committed to it. This raises the level of services from the simple CORBA dispatch of method requests, to include mission critical database capabilities such as recovery, integrity, concurrency, transactions, etc., as well as an object manager for caching and efficient support of millions of fine-grained objects. Finally, the Structure Query Language (SQL) widely-used standard is available, too, on some ODBMSs, allowing users, applications, and front-end tools to access objectbases just as they have accessed relational and other databases. Where it is available through Microsoft's Open Database Connectivity (ODBC) implementation of the SQL Access Group ISO Remote Database Access protocol, most front end tools work off-the-shelf, talking to multiple back end servers, objectbase and otherwise.

**Paper Number 3017
IMAGE/SQL At Work**

**Martin LaHiff
c/o Ella Washington
Hewlett-Packard Co.
408.447.1053**

Handouts will be provided at time of presentation

Using Replicate and NetBase for Your ALLBASE DBEs

Melanie Lallier
Quest Software, Inc.
610 Newport Center Drive, Suite 1400
Newport Beach, CA 92660

Abstract:

Replicate, also known as Warm Standby Logging, is a product from Hewlett Packard that consists of a set of routines that enables updates to one ALLBASE SQL database environment (DBE) to be duplicated elsewhere, either on other systems or in a different database on the same system. NetBase SQL shadowing consists of an export process and an import process which call routines from the Warm Standby Logging library in order to shadow an ALLBASE DBE. This session will cover:

- What is Replicate?
- How does Replicate work?
- What is NetBase SQL?
- What can Replicate and NetBase SQL do?
- Tips in implementing Replicate and NetBase SQL

What is Replicate?

Replicate, from Hewlett Packard, is a tool to facilitate shadowing changes made to the data in an ALLBASE/SQL database on one machine (master) to copies of the database elsewhere. With this tool, you can apply the same modifications, insertions and deletions which were made on the master (in the same order in which they originally occurred) to "shadow" copies of the database. By respecting the order of the transactions as well as the transactions themselves, the integrity of the shadow copies is assured. Shadow copies of the ALLBASE/SQL databases can be used to offload processing of reports from the master system and to distribute read-only users to shadow machines.

In reality, Replicate is implemented as a set of procedure calls that enable a resynchronization application, such as NetBase from Quest Software, to use the audit logging available in an ALLBASE/SQL environment to mirror changes made on a master node to shadow nodes. To understand Replicate, you must first understand the logging concept for ALLBASE/SQL.

When an update is made to an ALLBASE/SQL database environment (DBE), an audit log record is created. Each insertion, deletion, and modification to the DBE is logged. These log records may be accessed and used for duplication on the shadow systems via the Replicate routines. Because Replicate uses the audit logs extensively, Replicate is sometimes referred to as "Audit Logging" or "Warm Standby Logging".

An audit log exists for each copy of the DBE. In addition, a pointer, known as a Standby Checkpoint Record (SCR), exists to indicate where the shadow copy is with respect to the updates that have completed on the master. For example, when Warm Standby Logging (WSL) is started, the SCR is at zero. As transactions are performed on the master, they are logged. When shadowing begins, the network link is established, and the SCR is retrieved from the shadow machine. Assuming it is zero, then all transactions after zero are transmitted and applied to the shadow DBE.

Below is chart displaying the processing occurring on a master DBE and concurrent mirroring of those updates to a shadow system.

Master	Shadow	
<backup> <WSL>	<restore> <WSL>	To start, a backup of the master DBE must be made and restored on the shadow node. Once the backup is complete on the master, WSL must be enabled, and processing may resume on the master DBE.
trans#1 trans#2 trans#3	trans#1 trans#2 trans#3	Once communication is established between master and shadow, the initial SCR (0) is retrieved, and all subsequent transactions are transmitted and applied.
trans#4 trans#5	<offline for backup>	While backup occurs on the shadow, the master continues processing.
trans#6 trans#7	trans#4 trans#5 trans#6 trans#7	When the backup is finished; communication is re-established, and Replicate retrieves the SCR to know where to resume shadowing. In this case, we have trans#3, so we need all transactions after that.
trans#8 trans#9 trans#10	trans#8 trans#9 trans#10	Eventually, the shadow version becomes an exact copy of the master DBE again, as transaction activity on the master may be sporadic, while shadowing processing is constantly trying to catch up.

How Does Replicate Work?

Replicate depends on audit logs and SCRs. The audit logs contain the transactions necessary to duplicate the activity which occurred on the master DBE onto any other copy of the DBE. The SCR keeps track of where Replicate is in the process at any time. Replicate requires a minimum of two log files for both the master and shadow copies of the DBE. The log records may be maintained for different periods of time, depending on which of type of logging, archive or non-archive, is defined for the DBE.

Non-archive log files are like circular files; they contain current transactions. When a transaction is no longer current, its space in the log file can be reused for new transactions. In contrast, archive log files are semi-permanent histories of the DBE activity. They contain all the transactions committed since archive logging was enabled, and their space may only be freed for reuse by performing a special backup of the files.

Properly sizing the log files is critical. If you are using the circular-type "non-archive" log files, you do not want the SCR to point so far back in the log that it is looking for a transaction that is no longer available (because it was overwritten). If this happens, your DBEs have to be resync'ed. So, non-archive log files must be large enough to contain all the transactions being stored while the transport between the master and shadow is down. If you are using archive log files, you must have enough of them to store all of the transactions which occur between backups. If the log files are insufficient, you will have to stop processing on the master machine and build additional log files.

The second essential element to Warm Standby Logging is the SCR. The SCR contains information defining the DBE involved, the master node and shadow node, and the last committed transaction. If a DBE is being shadowed to multiple locations, each location has its own SCR so that each shadowing scenario is independent of any others. The uniqueness of the SCR within the network is critical to shadowing, and it is a result of properly enabled DBEs.

To enable WSL so that SCRs work, in the ISQL "START NEWLOG" command, you must specify a unique WSL ID and a unique Home Partition ID for each copy of the DBE. This way, when `get_scr` is called, it knows which SCR to retrieve!

For example, imagine that OURDBE is being shadowed to MYDBE and YOURDBE. If the network fails between OURDBE and MYDBE, shadowing may continue to YOURDBE. When the network link is re-established to MYDBE, Replicate needs to know where to begin transmitting updates from the audit log files for OURDBE. The SCR for OURDBE to MYDBE must therefore be different from the SCR for OURDBE to YOURDBE in order to keep both copies of the database in sync.

What Does NetBase Do?

NetBase SQL Shadowing was developed by Quest Software at the request of HP as a resynchronization application that uses the Replicate routines in the Warm Standby Logging library. NetBase enables you to quickly start shadowing your ALLBASE DBEs, without having to write a resync program. NetBase provides the networking know-how Quest has developed over the past 10 years and an easy-to-use directory that lets you implement Replicate painlessly and expand its use effortlessly.

Most resync programs hard-code which databases are to be shadowed and in what direction they are to be shadowed. Specifying a portion of a DBE to be shadowed is tricky, and whenever you want to change these details, a program modification is required. NetBase uses a directory to contain which DBEs (or tables of a DBE) are shadowed and how. When you want to change these details, you modify the directory with simple file equation type commands. The next time a user begins accessing the master copy of the DBE, the new shadowing strategy is put to use.

For the networking portion, NetBase provides an export process for the master side and an import process on the shadow system. When these two processes can communicate, the SCR is retrieved from the shadow system, and all subsequent transactions are transported from the master to the shadow. A control program is provided to facilitate starting, stopping and monitoring the import and export processes on each system.

Replicate routines are designed to shadow an entire database straight from the master to the shadow. NetBase allows you additional custom processing via user exits. You may re-route or selectively ignore transactions before they are passed into the network. You may read transactions out of the log file, without sending them across to shadow machines by setting up an export user exit in a design we refer to as a "pseudo-node." Using a pseudo-node user exit, you can log all updates or deletes (or both), or perform other analysis of the DBE activity without transmitting to remote nodes.

On the remote side, you can write user exits that ignore data, re-route data to other nodes, or do additional processing before the data is posted to the database. A third type of user exit may be called if an error was encountered during the update of the shadow.

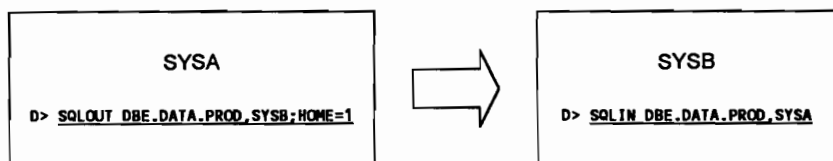
If a DBE must be resync'ed, NetBase provides a utility to reset the SCR appropriately. After restoring a valid copy of the DBE, this utility may be run to reset the SCR, and you are back to work.

What Can Replicate and NetBase Do?

The following diagrams display some of the options available for shadowing and the NetBase Directory entries required to accomplish them.

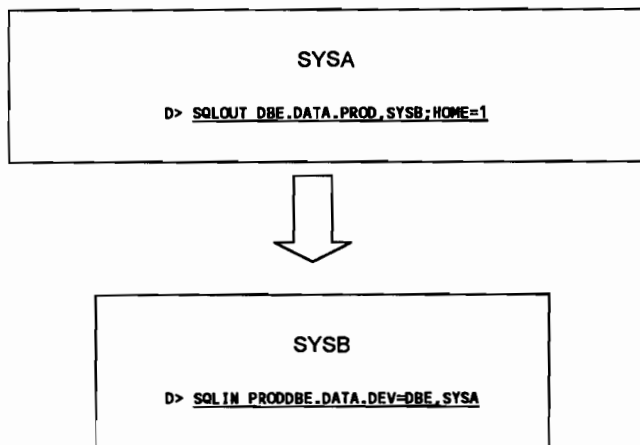
Single Direction SQL Shadowing

The simplest form of SQL shadowing is to shadow a DBE to another machine. In this case, we are shadowing DBE.DATA.PROD from node SYSA to SYSB.



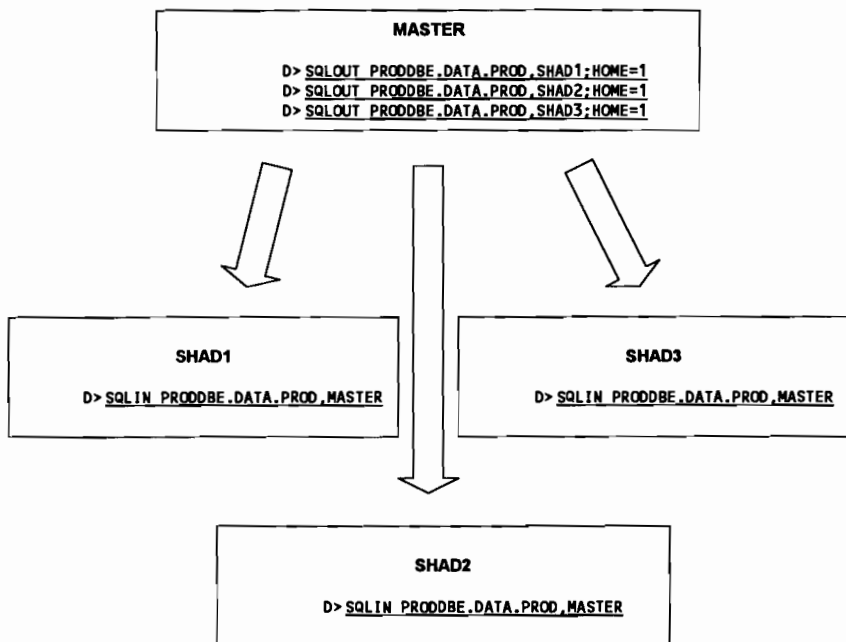
Rename a DBE

Below is an example of shadowing an ALLBASE/SQL database with one name on SYSA to a DBE on SYSB with another name. In this case, we are shadowing DBE.DATA.PROD to PRODDBE.DATA.DEV.



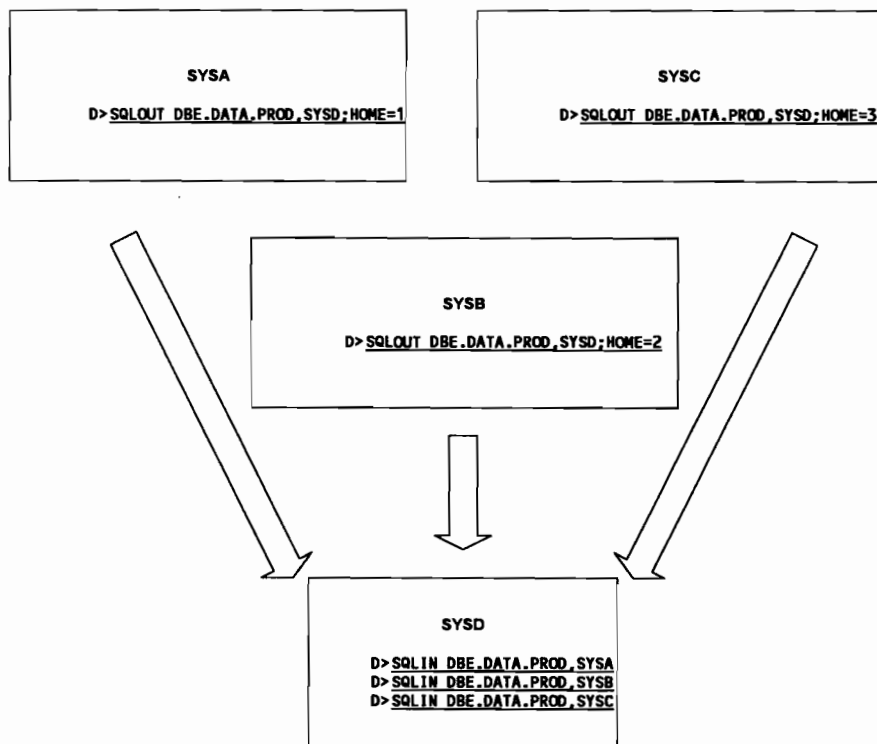
Distributed SQL Shadowing

In situations where many users need to read an ALLBASE/SQL DBE, you could implement distributed shadowing. All of the updates take place on the MASTER node and they are shadowed, via Replicate, to shadow nodes where read-only users are.



Consolidated SQL Shadowing

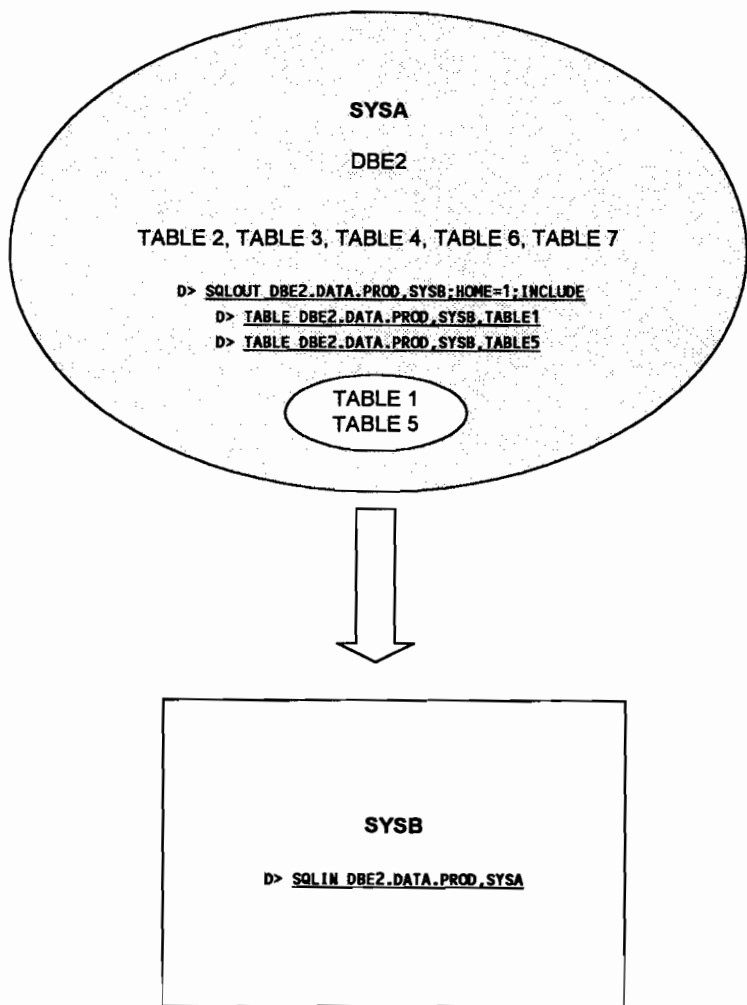
In other situations, users for different stores update the same master inventory database. In this case, consolidated SQL shadowing is useful.



Excluding Tables

In the previous examples, we have assumed that we wanted to shadow entire SQL databases. NetBase enables you to easily specify the tables you want to include or exclude.

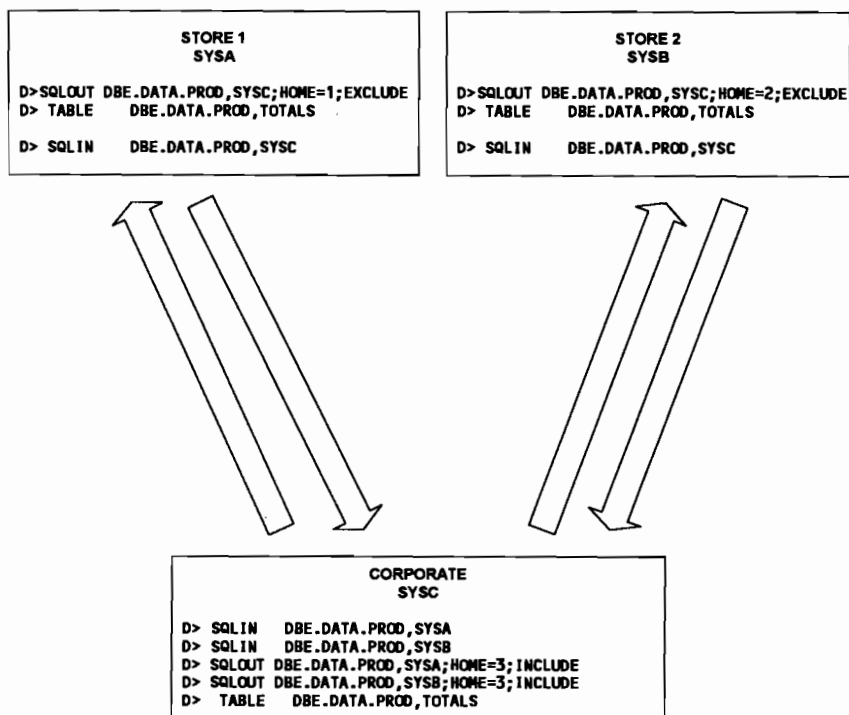
In the following example, DBE2 has a total of seven tables, only two of which we want to shadow. By specifying "INCLUDE" on the SLOUT entry, NetBase interprets the TABLE entries as the tables to be included. In this case, only tables 1 and 5 are to be shadowed.



Multi-Directional SQL Shadowing

Shadowing parts of an ALLBASE/SQL database "up" to the corporate headquarters while other tables in the same database are shadowed "down" to the stores is common.

Here are the NetBase directory entries required to shadow all of DBE.DATA.PROD except the TOTALS table from SYSA and SYSB to SYSC and the directory entries necessary to shadow the TOTALS table only from SYSC to SYSA and SYSB.



Tips on Implementing SQL Shadowing

- For each copy of each DBE, the **WSL ID** must be unique within the system.
- **Home_partition** may not be zero or duplicated elsewhere in the system. Each copy of each DBE must have a unique **home_partition ID**.
- Verify that the DBE was not created on a much older version of ALLBASE. If it was, the DBECon file is too small to add the WSL information required to enable standby logging.
- All systems should run the same version of Replicate. If not, be certain that the master system has the older version and the shadow systems have newer versions. Shadowing from old to new works; from new to old may not.
- Each system (master and shadows) must have at least two log files built to satisfy Warm Standby Logging's requirements.
- To insure that busy systems can continue to process without their connection with the shadow machine for a day or two without creating a hard-resync situation, define extra log files.
- To avoid a LOG FULL condition (which causes updates to be halted on the master), build enough log files. The required size and number depends on activity and the type of log files being built.
- NetBase SQL Shadowing requires Replicate and a network link.

Conclusion

If you have an ALLBASE SQL DBE that you want mirrored to another location, you need Replicate from Hewlett Packard. If you want a supported, turnkey solution when it comes to transporting the updates and implementing HP's Warm Standby Logging routines, you need Quest's NetBase SQL Shadowing.

Replicate Functions

- Retrieves the SCR.
- Opens the audit log.
- Reports any problems in the transmission of the transaction from the audit log to the log buffer.
- Applies the transaction on the shadow DBE.

NetBase Enhancements

- Export and import processes to handle networking issues.
- Control process to start, stop, and monitor the export and import processes which call the Replicate routines.
- Easy-to-use directory in which to specify DBEs to be shadowed, from which node and to which nodes.
- Ability to change shadowing strategies "on-the-fly" by changing directory entries being used by NetBase.
- Ability to easily shadow only specific tables within a DBE.
- Optional export user exit logic to perform additional processing of transactions prior to sending them from the master node.
- Optional post user exit logic to perform additional processing of transactions prior to applying them to the shadow DBE.
- Optional user exit logic to perform additional processing in case errors were encountered during the application of the transactions to the shadow DBE.
- Easy-to-use resync program to reset the SCR on the shadow node.

Paper Number 3019
Taking Steps to Client-Server with ODBC

M.B. Foster
M.B. Foster Associates
P.O. Box 580
50 Water
Chesterville, Ontario K0C 1H0
CANADA

Handouts will be provided at time of presentation



Paper #4000
written by Matt Hulett



1500 Dexter Avenue North
Seattle, Washington 98109
206-217-7100

The Changing Face of PC-to-UNIX Connectivity

Which Technology is Right For You?

As we age, we all get a few more wrinkles. As the face of PC-to-UNIX connectivity matures, IS professionals are faced with new wrinkles caused by increasing demands for enterprise connectivity, application integration and decentralization. Add to this a rapidly changing technology environment, and a few more gray hairs are no surprise to any IS staffer trying to bring UNIX applications down to the desktop.

I'm not speaking of access to raw UNIX-based data, but of access to business-critical applications from desktop PCs. Today, users throughout the organization are asking for access to those applications to do their jobs more efficiently.

Whether your company is considering terminal emulation, SQL client/server systems, X Window implementations, or some combination of the three to deliver UNIX-based

applications to PCs, you have to balance cost-effectiveness with functionality and reliability with innovation, depending on your environment.

Because it's easy to misstep in a rapidly changing environment, and the cost of making a wrong decision can be great, your best weapon is information. Today, I'll give you an overview about the pros and cons of each method and when each makes sense for access to UNIX applications. Unlike the "one-size-fits-all" school of thought, we believe that there is no best way, only the ways that work for *your* company's particular applications. Every method has its place; the question for you is, what's the best fit for today *and* the future?

Three Primary PC-to-UNIX Connection Technologies

According to a recent article in *Forbes* magazine, in 1980, virtually all corporate computing power resided in the central computer room. By 1987, the pendulum had swung so that 95 percent lay in desktop PCs. In the middle is UNIX, which is growing rapidly in this application downsizing environment. The IS department's challenge is to provide PC users with access to UNIX applications. There are three primary ways to deliver corporate power to the desktop.

Let's take a look at these three methods.

The ways of providing access to host applications that make the most sense in the current environment are: advanced terminal emulation, SQL client/server, and the X Window System. Please note: these are not mutually exclusive options, and you probably have one or more in place already. The point is to cut through the hype so you can determine which is most cost-effective for you today and to make effective choices for the future.

Terminal interface has been and continues to be the method used by 95 percent of mission-critical applications. *Advanced* terminal emulation changes the terminal emulation paradigm by integrating host and PC applications without expensive modifications to applications. Advanced emulation can offer multiple sessions, hide complexity from the end user, and take advantage of open systems standards.

SQL-based client/server systems are growing in popularity with the success of UNIX database offerings. Client/server applications promise to let the server do what it does best — disk input and output, CPU throughput, and non-interactive process scheduling. It can reduce network traffic, and distribute processing with the client node. This may also improve performance, while users get to work with the interface they're used to.

The X Window System, in some ways, offers the features of both terminal emulation and SQL client/server. You can maintain legacy code containing complex data relationships, and at the same time, offer a consistency of interface by presenting the information in a GUI format that is easier for the user to manipulate.

What are the various strengths and weaknesses of these three solutions, and when are they best employed? Let's start with the easiest to implement and generally lowest-cost solution — advanced emulation.

Putting the "Advanced" in Advanced Emulation

There is no free lunch, although the free terminal emulation programs provided by PC vendors would have you think so. These "thrown in" programs typically just act like dumb terminals, with the emphasis on "dumb."

Today, advanced emulation makes using corporate applications as easy as PC applications, saving training and support time. Users don't even have to know where an application resides. For example, automated scripting and named configurations can handle the connection method, login, or host command line. The icon to start order entry looks like any application icon. Screen hotspots, screen scrapers and interface bridging available with advanced emulators bring even more local interface features to UNIX applications.

Multiple sessions allow the integration of remote and local applications. Using advanced emulation, you can customize screens, assigning different sessions different colors for quick visual identification. You can also combine the functionality of various applications with advanced emulation, if you have copy and paste, DDE, and OLE features built in. Bridging between user interfaces using hotspots (buttons on terminal screens) can also be done with advanced emulation.

Advanced programmability features are one sign of advanced emulation. These features provide you with the tools to extend the life of legacy applications, using familiar programming environments. Look for integrated scripting languages, button palettes, support for OLE, DDE, Visual Basic and C/C++ interfaces and HLLAPPI to help you control the terminal/host interaction.

Advanced Emulation Advantages

In some applications, advanced emulation offers many real advantages:

- It works well with all established operating systems and existing terminals coexist with PCs.

- Advanced emulation is relatively inexpensive compared to porting legacy applications to new system. Emulators are generally cheaper than X servers, custom clients, X terminals or workstations.
- It provides access to multiple hosts and applications over serial or network connections. Since advanced emulation supports a variety of systems, it works well in heterogeneous environments, which are the norm.
- You can perform all management and maintenance on the host, so you are ensured a high degree of security. Emulators also provide increased usability without any code changes.

The Tradeoff

The traditional terminal interface is dated. Today's users are accustomed to slick new graphical user interfaces. GUI-style applications use hundreds of tools to make particular functions more convenient or easier to understand, but even basic controls such as pop-up menus or list boxes are still beyond the ability of the terminal interface. Advanced emulators enhance the interface dramatically, but behind all the makeup, it is still the terminal I/O.

An emulator also has limited knowledge about, or control over, the contents of the datastream. This lack of knowledge limits its ability to exchange data with local applications.

Best Uses of Advanced Emulation

Since all solutions are application-dependent, given this set of advantages and drawbacks, when is it appropriate to choose advanced emulation? There are four scenarios that make sense.

The first is when changing would not be cost-effective. Many times it's impractical to replace an existing application. If its use is declining, if it takes too much time and money to rewrite, or it fulfills a limited need, conversion may not be practical. Emulation can also be used as a step toward client/server or X interface systems without the concomitant resources.

The second scenario is when an application lends itself to being centralized. High-volume, on-line transaction services and high-security applications are poor candidates for migrating to LAN-based or client/server systems, since they are best controlled when entirely centralized.

The third scenario is on single-use stations. Delivery tracking, shop floor, and point-of-sale systems are more cost-effective and convenient using terminal-based connectivity.

The fourth scenario is when low bandwidth is a requirement. Terminal-based applications are most effective in low-bandwidth connections, such as modem connections, because they don't substantively add to network traffic. And while client/server computing has potential, it lacks a standard serial interface.

The Promise of SQL Client Server

Of the dozens of definitions of client/server, here we are talking about the most common and the one that receives the most press—SQL-based client/server implementations. Today, there are many commercial client/server implementations in the SQL server market. The major players are traditional UNIX database vendors such as Oracle, Informix, Sybase, Ingress, and Gupta. Front-end tools are also available from a number of vendors providing a GUI interface to SQL databases.

Object-oriented technologies, represented by the CORBA (Common Object Request Broker) specification and HP OODBMS (Object Oriented Database Management System), should eventually achieve a more complete distributed environment, but are quite a ways away today.

API standards currently extend to the transport layer only. Each DMBS has its own API and protocol for exchanging and describing SQL requests and data. Front-ends need adapters to convert these conversations to a common format.

The movement toward SQL client/server is really motivated not by what it delivers today, but by what it promises — central data control and data selection, scalability, and integration within the local user interface, file system, and applications.

The most significant advantage of the client/server model is integration with the local environment. To the user, remote applications appear and function like local applications. The remote applications even interact with other applications, and file systems and printers, as though they were local. And the client node shares the processing burden with the server. Because the server handles data selection and the interface is controlled by the client, network traffic is limited to requests and selected data. If implemented carefully, this translates into low bandwidth. (Obviously, if you have large images or encyclopedias in the database, no technique will be low bandwidth.)

IS administrators can tune servers for data throughput and transaction scheduling without worrying about character echo response, screen refresh, or other clock-sensitive interface issues. Meanwhile, the PC contributes the CPU cycles for presentation.

The Downside of SQL Client/Server

As industry pundit Robert X. Cringely says, "Open systems often aren't as open as their developers would like to think." There really is not one SQL standard. Instead, several dialects are competing for dominance.

Complicating matters for SQL client/server development is that most organizations have several PC desktop platforms. Windows, Macintosh, and OS/2 commonly coexist, along with UNIX workstations and X Window system implementations. Yet each PC operating system or workstation requires a native client.

Although many tool vendors claim to supply application portability across platforms, the most successful applications are implemented in homogeneous environments, on both the client and server side.

Careful planning and maintenance is required for client/server delivery. Successful performance depends on the careful orchestration of many parts: the server hardware, operating system, database, transports, as well as the PC hardware, display, network, operating system and transports. A less-than-optimal configuration or insufficient resources for any of the elements can create problems.

SQL itself has some limitations that can make conversion of existing applications difficult. For example, SQL is poor in relational operators. Common text based functions such as Soundex or synonym searches are not available. Certain ranges, such as date ranges, are also problematic. Workarounds to these limitations often require code on the server and the client, diminishing portability. The fact that SQL servers have different feature sets or feature uses also complicates portability.

Industry-sponsored database interfaces such as ODBC and IDAPI are a step toward portability. These provide a standard interface to SQL and other databases. Their very strengths are also drawbacks — because they are not designed for SQL-only or for a particular database, they can't take advantage of proprietary features in them. They offer a basic, but limited capability for data retrieval.

Perhaps most importantly, given all the changes required to implement SQL client/server, it tends to be the most expensive way of providing UNIX application access to the desktop. Considering these difficulties, the best advice for implementing wide-scale client/server applications is to do them in homogenous environments where you control all the variables: host, network desktop operating system, network card and software.

When SQL Makes Sense

While SQL client/server may be the solution of tomorrow, migration must first occur in limited applications. Here's a few.

For example, client/server applications are frequently used with Executive Information Support and Decision Support Systems, where information display flexibility and a graphical user interface are often desirable. A high level of integration and ease of use is desired, features offered by SQL client/server.

A single component of a larger system could also be a candidate for client/server, if it will benefit from local integration. For instance, a touch screen client could streamline stock pulling as part of an inventory system. Other candidates are

focused applications, such as document management or resources scheduling, which can be quickly put in place with common tools.

Departmental applications, with limited data distribution and where the desktops are likely to be homogenous, are also SQL-friendly applications.

With all of these implementations, a homogeneous operating environment is necessary to keep implementation costs down.

X Window — The Best of Both Worlds?

A more manageable way to update a user interface without restructuring an application for SQL client/server computing is the X Window System. X Window implementations act as highly intelligent computer terminals that use local computing power to manage the graphical user interface. The chief attraction of X is that it is an open standard providing true interoperability.

The X Window System was designed at MIT specifically to supply a platform-independent graphics and windowing protocol across networks, providing a common user interface. X clients and servers can be implemented on any hardware/operating system combination. Transport independence permits delivery via TCP/IP, DECnet, LAT, or serial connections.

Many tools are available to assist development, and X availability is almost universal. Commercial products are on the market for Windows, DOS, OS/2, and Macintosh. Plus, X terminals can coexist with X on PCs. PC X servers exist that can bring X speed and distributed graphical applications to PC users, using standard Windows conventions.

The X Window System offers several advantages:

- Legacy applications can be updated with a graphical user interface without reworking data flows and other internals. This is critical in environments where the internals cannot be easily converted to the SQL relational data model.
- Applications from different sources will always have the same appearance on the desktop, with common GUI features and conveniences.
- The market for X Window development tools is active, which means there are plenty of tools, and more coming all the time. X provides a structured, well-defined development environment, and X can be integrated with other GUI environments.
- Finally, API changes are subject to public input and customer demand, not unilateral decisions by a limited group of vendors. X is also a major part of the emerging COSE (Common Open System Environment) and CDE (Common Desktop Environment) standards.
- If using a mixture of X-terminals and PCs, the cost of deployment can be far lower than PCs or workstations for everyone, because the X-terminals do not have to be separately managed.

The Drawbacks

X has a reputation for heavy use of network bandwidth. But network usage is highly dependent on the type and implementation of each application (or X client). The traffic from a dialog-based application is significantly less. Server features such as Backing Store and Save Under save network traffic. Proper application and network planning can keep the load manageable.

X development is more difficult than that for a terminal interface. Even with an interface development tool, existing staff will need to be retrained to effectively design X interfaces.

X also lacks the diversity and sheer number of shrink-wrapped applications that are available in popular desktop environments: however, there are a few capable horizontal applications such as word processors, spreadsheets, paint and draw programs and desktop organizers.

X Window data exchange capabilities are limited. Although users can cut and paste with the local environment, similar to terminal emulation, more sophisticated methods of data exchange are still in specification and not yet available to X clients.

Because X is a display system only, unlike with Microsoft Windows, there is no consistent X help system.

And although X presentation is graphical, it does not necessarily behave like local applications. Most common control features, buttons, lists, and text boxes differ slightly from their PC counterparts.

When and Why X?

X can be the most cost-effective way to update the look, marketability, and convenience of a legacy application. Because it is well suited to heterogeneous environments, it is a good choice for companies with a wide variety of client platforms. X Window's reliability and ease-of-use make it a popular choice in almost any application. Many environments where UNIX workstations and X terminals are common find it a simple process to move the entire organization to X.

Comparing Solutions — Which is the Most Cost-Effective?

While one method may seem trendier or slicker than another, the chief criteria any organization must use when determining which one to choose is *cost-effectiveness*. Which implementation requires applications to be re-tooled? Which is quickest to implement? Which will help the largest numbers of users work effectively?

An advanced emulation solution is reasonably easy to implement, providing a new user interface yet requiring few, if any, changes to legacy applications.

SQL client/server solutions require the most work, involving restructuring the application logic and adding a new GUI in order to develop a full client/server application.

A PC X server offers some of the advantages of both of the other two methods, providing a standard user interface, regardless of where the client application resides. While there is some rewriting required, X is still easier to implement than SQL client/server.

Which PC-to-UNIX solution your organization embraces depends on which issue is most important — integration, reliability or investment required — in your particular environment. It also depends on how heterogeneous your systems are.

Advanced emulation integrates very well with existing systems and local applications. The reliability is tested, and the return on a relatively low investment is excellent. This cost-effectiveness is enhanced through the possibility of using dumb terminals for some users.

SQL client/server, while more difficult to implement, offers excellent integration: applications look and act like desktop applications. SQL client/server is also expensive, requiring either retraining of existing personnel or hiring of consultants or new staff who can rewrite the applications. Still, in a homogenous environment with the resources already in place, SQL client/server may be the best alternative.

Finally, the X Window System combines some of the features of both terminal emulation and client/server. It's also a good interim strategy for migrating to distributed processing. It provides visual consistency, reliability, and, although it typically requires a larger investment than terminal emulation, it usually costs much less to implement than a full client/server solution.

Ultimately, the choice comes down to how much it costs to bring legacy applications to the desktop. And that's dependent on your applications and environment.

4001

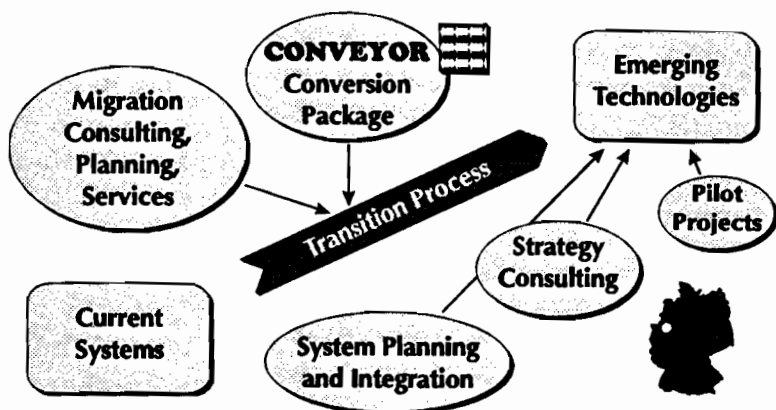
The Smooth Path to Client-Server Computing

Rolf Brandt

INFOSOFT GmbH, Hauert 12a, 44227 Dortmund, Germany

☎ +49 231 758 980

INFOSOFT was founded in 1985 and is based in Dortmund, Germany. The company is working in the area of Open Systems on the basis of new technologies and helps its customers with strategy consulting, system integration, and pilot projects. Moreover, INFOSOFT offers migration consulting, planning, and services to support the transition process to the new systems.



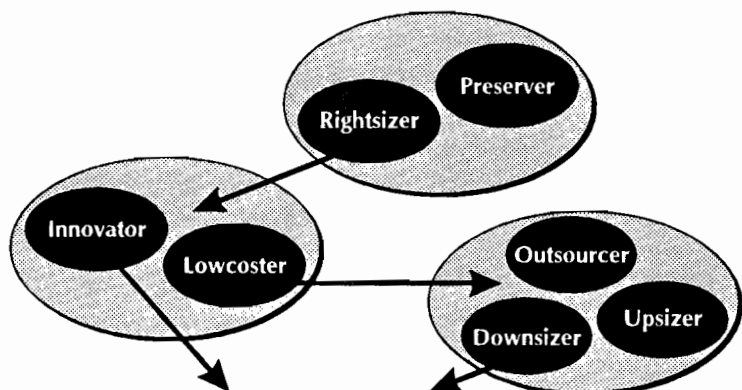
In the field of migration, INFOSOFT is specialized in automated conversion of proven legacy application software. To assist the company in its conversion activities, the CONVEYOR toolset has been developed. With this toolset, INFOSOFT has already completed 130 projects. The CONVEYOR technology allows for both re-hosting and more complex conversions covering the aspect of re-architecting.

INFOSOFT has formed strategic partnerships with leading hardware manufacturers and many companies providing essential basic

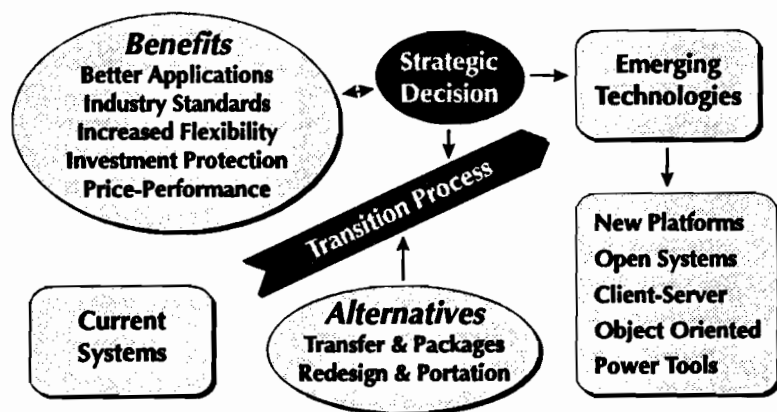
The Smooth Path to Client-Server Computing

4001-1

products. Additionally, cooperation agreements with various system integrators throughout the world have been closed. These partner companies provide conversion services with CONVEYOR in their local markets.



At present, practically all companies are considering a transition to Open Systems and progressive technologies. These considerations are mainly driven by two approaches.

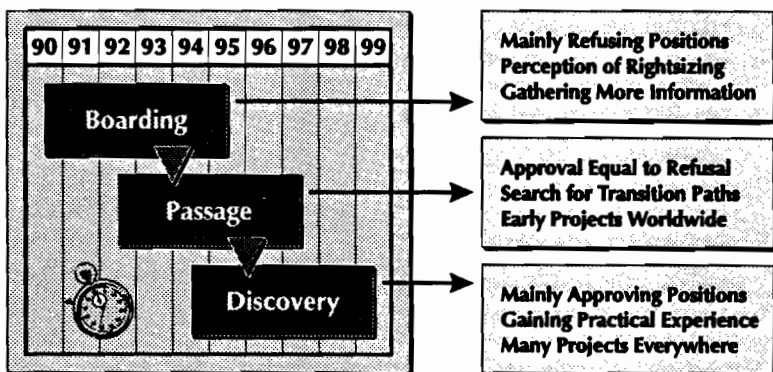


On one hand, we find companies planning to invest in innovative developments. On the other hand, cost saving aspects are dominating in many corporate entities. However, both strategies finally lead

to the same results on the basis of similar procedures and methodologies.

In nearly all cases, the transition process starts with a strategic decision. This is a strategic decision because a single reason cannot be identified. A lot of different factors play a part. The most important ones are: better applications, industry standards, increased flexibility, investment protection, price-performance. Normally, the decision is not based on a fully documented cost-benefit analysis.

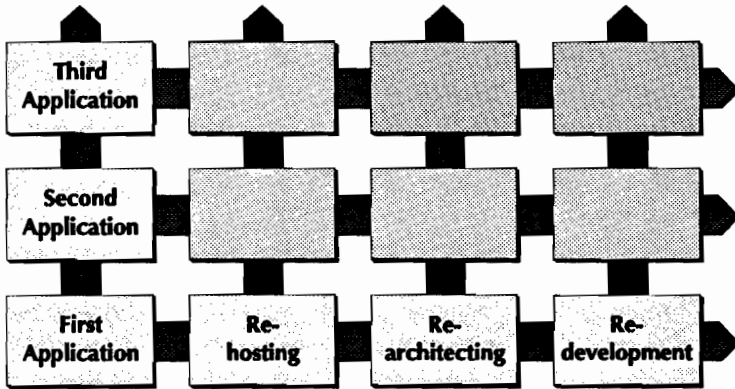
There are different alternatives for the migration of applications to the target system available which have to be assessed in detail. Standard packages in use can be transferred and additional packages can be implemented. Proven custom software can be ported. Finally, new custom applications can be developed with methods and tools reflecting the current state of technology.



In spite of the fact so many companies are planning migration and the professional magazines are full of backing articles and success stories, the transition process is actually moving very slowly ahead. Many companies are hesitating because they fear the efforts involved with migration, and the entire process appears risky to them. Moreover, it is very often not completely clear how the target environment should be configured.

On the voyage to the new systems, we can identify three phases. In the first phase, companies have perceived the rightsizing idea and mainly taken refusing positions. They started gathering more information and waited otherwise else for the things to come. This phase began in 1990 and will end in 1995.

In the second phase, approval has increased and finally equalled refusal. The search for viable transition strategies started. Throughout the world, the first early projects were carried out. This phase began overlapping with the first one in 1992 and will end in 1997. In the third phase which began in 1994 and will end in 1999, approval prevails. The companies are gaining practical experience with the new technologies in this phase. Many projects are executed everywhere.



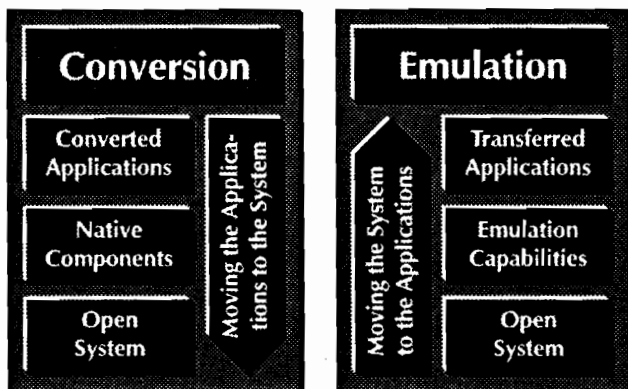
As the efforts for the transition process must not be underestimated, we recommend a two-dimensional, step-by-step approach. On one side, the applications should not be migrated at the same time but one after the other. On the other side, the technology gap should not be bridged in one piece but in re-hosting, re-architecting, and re-development steps.

During re-hosting, existing applications are moved to the target system without changing architecture, program structures and functionalities. Re-architecting includes the re-construction of the archi-

ture, for example in the direction of client-server, relational databases, and graphical user interfaces. Re-development also implies changes of program structures and functionalities.

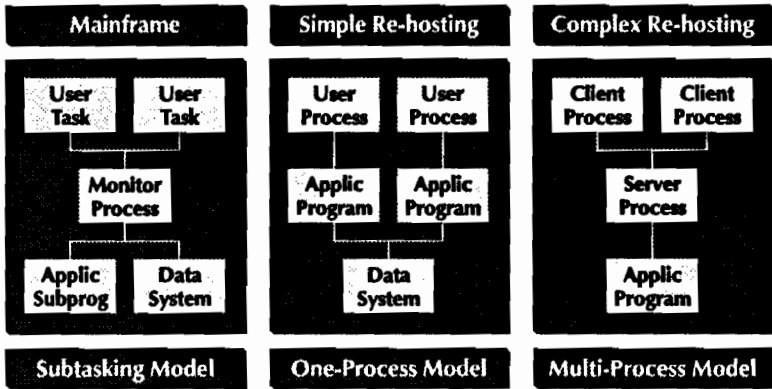
Applying the step-by-step approach however, we need to take care that the separate steps can be executed consecutively without any additional effort which can be avoided. That means this approach has to be designed very flexible, resulting in the advantage that adaptations to the continuous technology deployment can be effected much faster if required.

Additionally, considerable costs are saved through the step-by-step approach. The time and money measurements between re-hosting and re-development come to about 1:70. This is mainly caused by the fact that the time during which the old and new platforms have to run in parallel operation can be reduced dramatically if the transition process takes off with re-hosting.

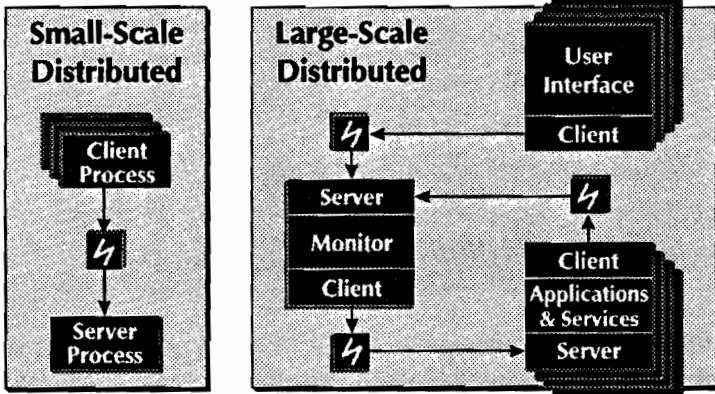


For portation of custom software in the re-hosting stage, we recommend conversion as the best approach compared to emulation. During conversion, the applications are moved to the Open Systems and basic middleware products available on them. With emulation however, the new environment is moved to the applications with the aid of a layer replicating the old environment. Emulation

does not lead into the new world of Open Systems. The next step of re-architecting becomes practically infeasible.

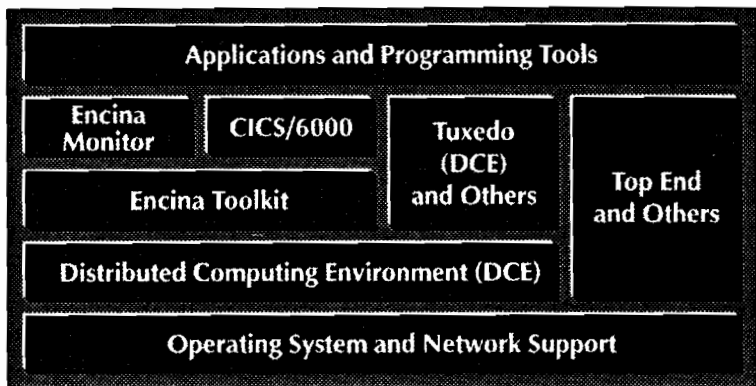


In the framework of re-hosting, the question arises whether the native facilities of the target system without any additional capabilities are sufficient or middleware products for supporting transaction processing should be used.

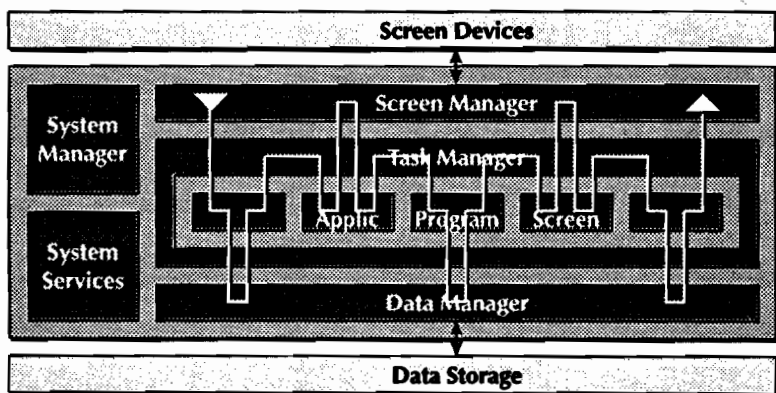


This question is often raised to an important issue because transaction monitors are normally implemented on traditional systems and many users cannot imagine online processing without monitor.

However, we cannot recommend to implement a transaction monitor on standalone Open Systems. The new transaction managers for Open Systems considerably differ from those on traditional systems.



They support distributed computing models and often do not produce any throughput improvement on standalone systems, caused by their numerous underlying middleware layers.

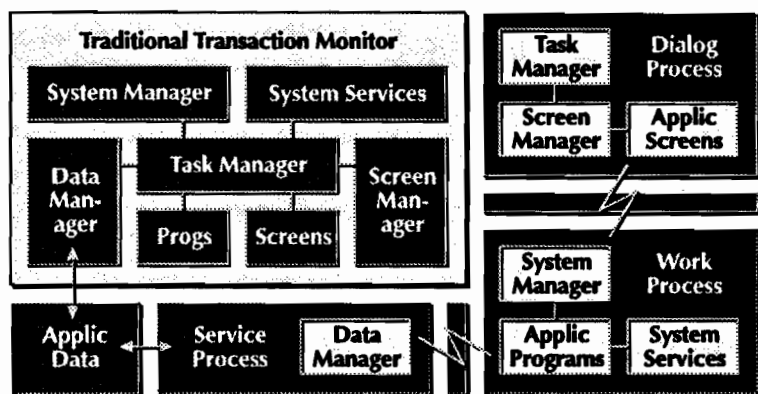


In comparable situations, the performance results of traditional monitors cannot be achieved because of the divergent architec-

tures. As far as other aspects are concerned, the new transaction managers do not provide any decisive advantages.

As the new transaction systems are designed for client-server processing, they are only a reasonable choice for distributed environments. In many cases however, a distributed computing model can be implemented in a much simpler and cheaper manner if the more or less proprietary products are used which are currently on offer and allow limited client-server solutions.

The presently emerging generalized transaction systems on the basis of DCE (Distributed Computing Environment) are only a choice for very complex computing topologies. In all other cases, they can easily lead to overkill effects. They are not yet completely mature, by the way. Many components and the integration into development environments are still missing. In production, they are rarely found at present.

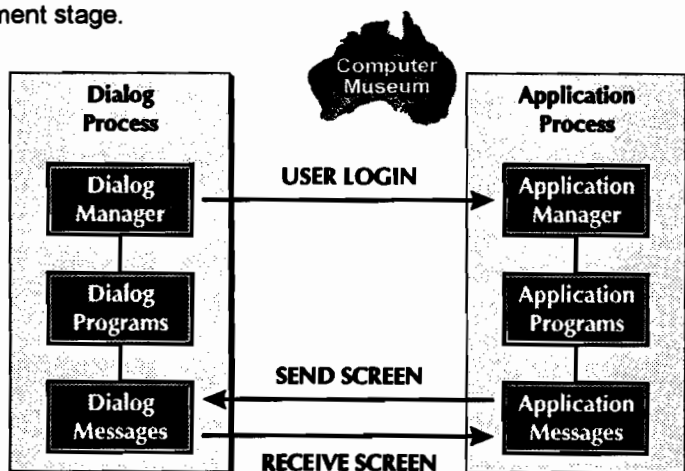


The transition to client-server processing requires a re-architecting in any case. We need to decompose the monolithic structure of the source system into its various components. These components then have to be re-composed on the basis of the architecture of the target system. We need to take care that the fundamentals for the re-composition are already provided in the re-hosting stage.

Under the control of the transaction monitor, traditional software systems are application-driven. The application programs contain the dialog logic, send datastreams to the terminals through the transaction monitor, and receive responses on the same way.

Client-server applications, however, work dialog-driven. The dialog processing is completely performed on the client systems, independent from the application programs on the server systems which are called as remote procedures. The programs do not contain any dialog logic.

Application-driven software packages cannot be converted fully automatically to dialog-driven solutions because of the inverted control flows and other structural differences. To achieve this goal, a re-construction of the architecture is not sufficient. Additionally, the various components on which it is built also have to be re-constructed. As the application logic is affected during this procedure, manual interventions are required. In this manner however, we already get to the last phase of the transition process, the re-development stage.



If we want to gather experience in the world of client-server processing without entering the re-development stage too early, we can also succeed with re-architecting techniques. In this case however,

we have to make a compromise. We can only build an architecture in which dialog-driven client processes asynchronously communicate with application-driven server processes.

In this model, the application processes have to be started at first. The dialog processes can then inform the application servers of their existence, receive (dialog) messages from them, and send (dialog) messages back to them. For the message receipt, the dialog and application processes wait at defined interrupt points in turn.

Integrating Workstations, PCs and Macs

Paper 4002

Brad Fezell
Manager of Information Systems
Dril-Quip, Inc.
13550 Hempstead Highway
Houston, TX 77040
(713) 939-7711

The Desktop Integration Challenge

There are three primary desktop computing platforms in use by the typical American corporation today -- IBM compatible PCs, the Apple Macintosh and several flavors of UNIX workstations. The majority of PCs in the corporate workplace are networked today and of course the Apple Macintosh and UNIX workstations have always had networking capabilities built-in.

Only recently, however, have significant tools become available to integrate these platforms in a heterogeneous fashion. The task of desktop integration sometimes involves elegant tools with cute user interfaces, but at other times, the ugly retro-fit nature of integrating dissimilar architectures is difficult to hide.

Desktop integration can be as simple as transferring files via diskette or so complex and transparent that the user has no clue that their word processor is actually running on someone else's CPU. Each environment has its own unique requirements. Some departments just want to share a printer among Mac and PC users, others want to transparently interchange documents between PCs, Macs and UNIX workstations. Still others, in addition to printer and file sharing, want to run UNIX applications on their PCs and Mac applications on their UNIX workstations.

Ingredients for Success

The systems integrator is one who, in our current context, integrates heterogeneous systems. A systems integrator's job is one that requires a great deal of planning and discipline because there are so many variables involved when several operating systems and hardware platforms are involved. Throw in two or three different network operating systems and topologies and you can see why the discipline and planning are so important.

I believe that traditional MIS disciplines are the most important ingredient in the desktop integration success recipe. Those of us with traditional MIS backgrounds have always placed a premium on the disciplines of reliability, accessibility and serviceability. We are greatly concerned with things like up-time, security, backup and disaster recovery. If you want your project to be implemented such that the maintenance requirement after installation is minimized and the chance of success is maximized, employ these traditional MIS principles from the start.

Parallel Paths

The number of products available to accomplish our goal is staggering. Every week I read about new desktop integration products. In some categories, there is a clear cut winner but in others you may have to do some of your own testing to determine which product works best in your environment.

I will present the products I am familiar with, and others that I am aware of, but this is by no means a comprehensive list. I will also give some insights on real world implementation where I can. This section is organized by type of machine and direction of integration.

UNIX Workstation ⇒ PC

Insignia SoftWindows with Netware Connectivity

Runs DOS and MS Windows applications on UNIX workstations. SoftWindows is a software based MS-DOS emulator that runs on both UNIX and Macintosh workstations. Insignia recently changed the product's name from SoftPC to SoftWindows, but it is essentially the same product with MS Windows included.

The newest version of SoftWindows adds Netware connectivity which is a boon for shops that are integrating workstations and Netware PCs. When connecting to a Netware file server you must use the Ethernet_II frame type if you are using an HP 700 workstation. If you are not binding this frame type to your Netware file server's ethernet card, you will need to add a couple of lines to your Netware server's autoexec.ncf file. On my file server, the added lines read as follows:

```
LOAD HP386E32.LAN SLOT=1 FRAME=ETHERNET_II NAME=ENET_II  
BIND IPX TO ENET_II NET=2
```

Having added these lines, you can reboot the Netware server and run SoftWindows on your workstation and connect to the Netware file server just as an ordinary PC would.

I have tested several DOS and MS Windows applications from the Netware file server and had no problems at all. However, running MS Windows on a workstation is bound to be a disappointment for most users because of its performance. I tried several MS Windows applications on my 715/50 but none of them, not even MS Windows Write, ran with acceptable speed. I also tried these applications on a 735 and was quite pleased with the speed, but how many of us have 735s on our desk? Character mode DOS programs run quite well though, and I highly recommend SoftWindows for running DOS programs if you can not find a UNIX alternative.

SoftWindows has competition coming, but for now, most reviewers still consider it to be the fastest, most compatible and reliable way to run DOS and MS Windows applications on UNIX.

SunSoft WABI

Runs certain MS Windows applications on UNIX workstations. This product is designed to translate MS Windows calls into X-Window calls therefore allowing unmodified versions of MS Windows programs to run in the WABI environment on UNIX workstations. Initially, WABI made claims of running MS Windows applications at native X-Windows speed, but the reality is that even SoftWindows is faster. In addition to problems with speed, many programs require MS Windows to be installed on the workstation in addition to WABI and still others do not work at all.

While WABI may not be ready for prime time, it is still a great idea that is badly needed. Major software vendors like Lotus have halted development on mainstream products like 1-2-3 for UNIX citing profitability concerns. If more software vendors follow suite we could see a deficiency of mainstream personal productivity applications for UNIX. What we desperately need in the UNIX workstation world are the same low cost applications that are available to MS Windows users. If WABI or other such products will answer this need with adequate speed and compatibility, our problem will be solved.

Novell Netware NFS and Flex/IP

Exports PC file systems to UNIX workstations. Provides bi-directional print services. There may be no better solution for transparent UNIX workstation access to PC directories and bi-directional printing. When I say bi-directional printing I mean that you can print from a PC to any UNIX printer or from a UNIX workstation to any networked PC printer. Flex/IP is a subset of Netware NFS that only provides the print services.

This product runs on a Netware file server as a set of Netware Loadable Modules and once installed, allows you to mount Netware directories on UNIX workstations or any machine running NFS. UNIX workstation users can store files with long names up to 256 characters by virtue of the NFS name space that NFS installs on the Netware volumes that you choose during installation.

Some applications such as WordPerfect and Lotus 1-2-3 use the same file formats between both their DOS and UNIX versions so you can share files between DOS and UNIX users without translation.

Netware NFS also provides FTP file transfers between DOS and UNIX machines as well as an X-Window client for managing the Netware console from a UNIX workstation. The X-Window client performs the same function as the RCONSOLE utility that is used on PCs. Essentially, it allows you to do the same things that you would do at the Netware file server console from your workstation.

This is a product I highly recommend for users who want to share files and printers between PC and workstation users because of its effective simplicity and low maintenance requirement.

DESQview/X

Exports DOS applications to UNIX or Mac X-servers. DESQview/X is, as far as I know, a one of a kind product. To my knowledge there is not another product that will let you run a DOS program on a PC and display it in an X-Window on another machine.

DESQview/X may be a contender for the best DOS on UNIX solution when character mode DOS applications are what you need. The product is a full featured X-Window implementation on DOS that can be used by PC users in the same fashion that X-Windows is used by UNIX workstation users. X-Windows has the ability to run a program on one machine and display it on another which is a great feature.

A powerful PC with lots of RAM and disk space could be configured as a DOS application server to UNIX workstation users in lieu of SoftWindows. For instance, I have run eleven WordPerfect 5.1 for DOS sessions on a DESQview/X PC and displayed them all to a UNIX workstation and maintained respectable application speed.

My experience running MS Windows under DESQview/X was not encouraging but I do not want to discourage you from trying it. Theoretically, you can run MS Windows on DESQview/X and display it on a UNIX workstation.

UNIX Workstation ⇒ Macintosh

MAE – Macintosh Application Environment

Runs Macintosh applications on UNIX workstations. Apple has apparently scored a hit with this System 7 on UNIX emulator. I recently tested this product and was quite impressed. The speed on my HP 9000 715/50 is comparable to a Mac LC and compatibility with off the shelf Mac applications is very good. One major shortcoming is no network support. Any Mac application that expects a network will have a problem with MAE. This actually disqualifies MAE as an integration tool in the purest sense because no real Macs can be involved, but even so the product is worth mentioning.

If your shop is heavy into Macs and UNIX workstations and you need to run Mac apps on your workstations, this product could be your answer.

PC ⇒ UNIX Workstation

Reflection/X for MS Windows

Displays X-Window applications on PCs. Walker Richer & Quinn have long been making terrific emulation and connectivity products and Reflection/X follows the WRQ tradition of quality, function and form. Reflection/X essentially turns your PC into a smart x-terminal.

At this point I should give a brief explanation of the x-windows client/server model. The server in x-windows is the program controlling the display(s), keyboard and mouse. A client is the program initiated and controlled by the user. The client is written to interact with the server for access to the input/output devices. Reflection/X is considered an x-server product although some x-clients are also included with the package.

Because it runs under MS Windows, Reflection/X will run along side other MS Windows programs. You can choose HP Vue, Vue Lite or xdm just as you would on a workstation or x-terminal. X-Window applications can be displayed under the local window management of MS Windows or under host window management and you can assign icons to your favorite X-Windows applications. Reflection/X also comes with a drag-n-drop FTP client that lets you transfer files between your UNIX and PC hosts.

The next time you need an additional workstation, consider Reflection/X or one of its competitors. The performance of today's PCs rival that of dedicated x-terminals and the additional capabilities of an MS Windows PC make it an interesting alternative that might save you some money.

There are many other MS Windows based x-servers on the market of which Hummingbird Communication's eXceed/W is the most popular.

DESQview/X

Displays X-Window applications on PCs. We discussed DESQview/X earlier as a DOS application host to other X-Window servers. DESQview/X is also a full featured DOS based x-server that does the same thing Reflection/X does only without MS Windows.

The down side of DESQview/X as an x-server is poor video performance and options. There are few supported video cards and resolutions compared to MS Windows based x-servers.

PC Based NFS products (there are a host of them). PC-NFS, WRQ's NFS, etc.

Exports UNIX file systems to PCs. I have not tested any of these products so I can only speak to the concept of accessing UNIX file systems from PCs. Sun's PC-NFS has been around for a long time and HP-UX supports it with its pcnfsd daemon. Walker Richer Quinn also has a new NFS product for PCs which should integrate well if you are using other WRQ products.

If you have only a few PCs in a mostly workstation environment, one of these products might be a good choice for sharing the UNIX file systems with your PCs.

Macintosh ⇒ UNIX Workstation

MacX

Displays X-Window applications on Macs. This product is an X-Window server produced by Apple and does essentially the same thing that Reflection/X and other PC based x-servers do.

WRQ's Reflection for Macintosh

Provides Terminal Emulation. For hard wired or modem serial communications, Reflection for Macintosh provides good terminal emulation for character based UNIX applications. If you need TCP/IP networked communications and file transfer, Reflection for Macintosh works with our next product, WRQ's Telnet for Macintosh.

WRQ's Telnet for Macintosh

File Transfer for Macs. Telnet for Macintosh allows you to connect to a UNIX host via TCP/IP network connection using Reflection for Macintosh and the MacTCP init included with Mac System 7. There is also FTP file transfer ability.

Macintosh ⇒ PC

Novell Netware for Macintosh

Provides access to Netware volumes from Macs. This is a solid product that allows Mac and PC users to share files and printers. Most of the features of the previously discussed Novell NFS product are present with Netware for Macintosh. Mac users can store files with long file names on Netware file servers and print to Netware printers.

Some shops use this product to provide mass storage subsystems such as hierarchical management systems (HMS) to both Mac and Netware users. There are many such systems available for Netware servers. This gives both types of users access to optical jukebox storage systems of 20 to 200 or more gigabytes in size. In a graphics intensive environment such as document imaging, this can be a great benefit.

Insignia SoftWindows with Netware Connectivity

Runs DOS and MS Windows applications on Macintosh workstations.
Please refer to the discussion above on Insignia SoftWindows.

Conclusion

New products arrive almost every week to bridge the gap between different desktop systems. As you consider a strategy to bring your systems together, keep the big picture in focus. Try to address as many issues as possible with one vendor's products in order to keep your configurations simple and your variables down.

I encourage you to remember the disciplines of reliability, accessibility and serviceability as you implement these solutions because you will be applying a greater degree of complexity to your environment. Following standards and de facto standards always pays dividends. Remember, our main concern should be that our systems remain on line and our users have access to the data they need.

Integrating PC Lans and Unix Servers
Ann Hewitt
Hewlett-Packard

Hewlett-Packard is a recognized leader in integrating personal computers and Unix servers. HP's strategy is based on the need for PC users to access resources and applications on the HP9000 as though they were local to the user. Resources that users would access include databases, shared peripherals, and applications. To be successful in the client-server marketplace, HP needs to provide the capability to integrate PC's and servers independent of the Network Operating System.

The key components of HP's strategy are to:

- * Integrate all HP-UX application servers into all common PC NOS environments.
- * Continue to provide a variety of solutions to our customers. Today HP provides Netware/9000, LM/X 1.4, LM/U 2.2, PC ARPA services, as well as third party solutions including Banyan ENS, PC-DCE, PC-NFS and PacerShare for PC integration with the HP9000.
- * Extend HP leadership in PC LAN integration by providing support for partners using the client-server application programming interfaces (API's) found in Netware/9000 and LAN Manager.
- * Investigate new technology offerings such as Microsoft's Windows NT and determine where NT might augment our current offerings.

CURRENT PRODUCTS:

NetWare for Unix: Customers who need connectivity to HP9000s for their Netware clients may find Netware/9000 an effective solution for application access and printer sharing. There is a wide range of Unix applications including all the leading relational database products that customers using Netware today might need to integrate with. However, Netware/9000 is not a replacement for an Intel-based server. It is not possible to get the type of DOS file and printer service performance available using the Intel platform from Netware/9000.

In addition to file and print sharing capabilities, NetWare on the HP9000 offers limited support for the Netware Application Programming Interfaces (APIs) such as Sequence Packet Exchange (SPX). These APIs are the foundation for client/server application development and are of particular importance to developers and end users alike. The SPX interface for both client and server are shipped as part of NetWare for the HP9000. However, HP does not provide support for architectural, design, or programming problems related to creating an application using the Netware APIs. API manuals are shipped by special request.

LAN Manager 1.4: LAN Manager 1.4 was jointly developed by Hewlett Packard and Microsoft. LAN Manager/X is an advanced, full featured network operating environment that enables the HP9000 Series 800 and 700 to operate as file and resource servers to OS/2 and MS-DOS Pcs. LAN Manager/X gives PC users access to the power, resources and security of Unix. PC applications can continue to be used from the familiar PC environment. With LAN Manager/X, users can create, delete and modify files on HP-UX servers from MS-DOS and OS/2 PC workstations. PC users are also able to share devices such as printers and plotters.

LM/U 2.2: HP LAN Manager version 2.2 (abbreviated as LM/U 2.2) provides the latest version of LAN Manager for Unix systems on a HP-UX platform. LM/U offers several features beyond the LM/X 1.4 offering: domain administration, advanced security, file replication and windows-based management. However, performance considerations limit LM/U 2.2 to casual file and printer sharing. For customers requiring the LM/U 2.2 features, performance expectations should be set accordingly.

PC ARPA and Network Services: HP PC ARPA and Network Services provide a PC running MS-DOS with HP ARPA Services to host computers in a multivendor environment or to HP application servers. These services supply a complete and sophisticated set of communication facilities between the PC and the minicomputer, complementing LAN Manager or NetWare solutions. With HP ARPA services, a PC running MS-DOS can communicate in a multivendor environment using the networking services defined by the Department of Defense Advanced Research Project Agency (ARPA) and Berkeley Software Distribution (BSD).

THIRD PARTY SOLUTIONS:

BANYAN: Enterprise Network Services from Banyan is the first implementation of Banyan's ENS services on a RISC/Unix platform. ENS for the HP 9000 Series 800 provides functionality equivalent to native VINES Release 5.52 (StreetTalk Directory Services, Security, Management and file and print services.) DOS, Windows OS/2.x and Unix workstations have full access to the Banyan services as well as common access to mission-critical HP-UX applications. For ordering information call Tom Schmidt, Channel Sales Representative, at 415-391-5655.

PC-DCE: PC-DCE, available from Gradient Technologies Inc., provides a full implementation of DCE for the personal computer using DCE peer support and Microsoft Windows (rev. 3.0 or higher) as the software environment on the PC. PC-DCE provides an excellent bridge between the Distributed Computing Environment and the world of Windows based PC applications. The PC-DCE Core Set offers DCE RPC Threads, Security and Naming support. The Extension set adds support for the distributed File System (DFS), Time Service and Remote Print Services. For more information on PC-DCE contact Gradient Technologies at tel: 508-562-2882.

PACERSHARE: PacerShare, from Pacer software, is an implementation of Apple's AppleShare file server software and allows an HP 9000 to act as a file server for Macintosh PC's. Users can fully integrate their Macintosh and Unix system files and applications using PacerShare. For detailed information, contact Pacer Software at tel: (619) 454-0565.

PC-NFS: NFS (Network File System), developed by Sun Microsystems, provides transparent access to remote systems on the network. HP bundles NFS with the HP-UX operating system, however customers may purchase PC-NFS for NFS connectivity to the server. PC-NFS allows a PC to act as an NFS client and access or share files and resources on an NFS server. For the PC, Hewlett Packard references SunSoft Inc. at tel: (800) USA-4-SUN or FTP Software at tel: (617) 246-0900.

SELECTING A PC INTEGRATION SOLUTION:

Information needs to flow freely throughout an organization. With a variety of offerings from Hewlett Packard, HP's customers are able to integrate Netware, LAN Manager, or Banyan PC's as needed. HP provides the flexibility to choose the network integration solution best suited to business needs and the existing investment. If the PC clients tend to require information from the HP 9000 infrequently or only need file transfer and/or virtual terminal capability, HP ARPA Services for the PC would be adequate. Using a Unix platform for file and print services is not a new concept. If you are faced with integrating DOS, Windows, or OS/2 workstations into an existing multi-vendor environment choose the PC NOS solution which closely fits the customers requirements.

If a customer is already using a PC Network Operating System, (i.e. Lan Manager, Netware or Banyan) it does not make sense to introduce a new protocol set to share data and resources. The appropriate layered HP-UX NOS should be selected allowing for performance expectations.

PC-NFS is a solution for those environments where NFS is already the strategic integration solution.

PC-DCE is a solution for customers focusing on DCE integration. It allows the PC to interoperate with all other DCE compliant systems as a trusted peer and bridges the PC world and DCE networks.

For Macintosh connectivity, Hewlett Packard references Pacershare from Pacer Software. There are a number of third party solutions that also provide Macintosh connectivity. For a list of additional solutions, contact the Sales Response Center.

WORLDWIDE SALES AND SUPPORT

Hewlett-Packard's support organization has established more than 400 support offices in more than 90 countries. Hewlett Packard will provide comprehensive support and service for all PC NOS offerings on the HP-UX platform.

Conclusion:

Hewlett-Packard is committed to client-server computing in a multivendor environment. HP's experience in PC-server integration products, plus our long history of OLTP applications create a base of experience necessary for success. HP's dominance in PC server integration depends on our ability to provide market-leading PC integration solutions to our HP9000 product family. In addition, it is well recognized within Hewlett-Packard that our success also rests on the strength of our partnerships with the leading VAB application providers. HP is committed to working together with our VABs to provide best-in-class applications that utilize PC workstation clients and HP Business servers.

Paper # 4004

Integrating PCs into the UNIX Environment

written by Matt Hulett



1500 Dexter Avenue North
Seattle, Washington 98109
206-217-7500

Introduction

One of the biggest issues facing organizations in making the move to open systems is integrating Microsoft Windows and UNIX. Microsoft Windows is now the most popular desktop operating system, and users, familiar with its graphical user interface and intuitive operations want the same ease of use when working with host-and server-based applications. Yet UNIX, which is rapidly becoming the operating system of choice for the server platform, is far from intuitive. To provide integration between the two distinct operating systems, many IS administrators have adopted the X Window System.

X allows IS departments to provide users with the Windows interface and the functionality they expect, without having to re-port existing applications. X has other significant advantages, too—especially its ability to work across a variety of platforms. This high degree of interoperability makes X the most versatile choice in tying the Windows and UNIX platforms together.

Implementing X on the desktop involves several considerations. These include evaluating PC X servers, selecting a TCP/IP stack, and evaluating offerings in terms of additional capabilities such as VT320 emulation and TCP/IP client applications.

Defining X

The X Window System is best described as a hardware- and network-independent windowing and graphics protocol. It was developed from two 1980's research projects, "W" at Stanford and "Project Athena" at the Massachusetts Institute of Technology (MIT), funded jointly by MIT, Digital Equipment Corporation (DEC), and IBM. In 1986, Digital offered the first commercial version of X. The X Consortium was formed in 1987 with MIT, Tektronix, IBM, DEC, Sun, HP, and AT&T taking part.

X provides interoperability through two basic components: a client and a server. X clients reside on hosts (frequently UNIX servers, though they can reside on any host equipped to run over TCP/IP).

X servers typically reside on the PC where they provide the graphics display, though a server can run on a variety of machines, including an X terminal. (Although this seems contrary to the usual definitions of client and server, it is not inconsistent if you consider the client as the application requesting the windowing and graphics services and the server as the application providing the particular services requested. Using this definition, where the client and server reside is irrelevant.) An X server controls the display and the input devices, such as a keyboard or mouse. It monitors commands from the client and input from the user (key-strokes and mouse clicks), which it passes on to the client.

In the world of X, the user does not care what host an application resides on. A user can simultaneously run applications on machines that are alien to each other. Each application appears in its own window on screen, and users can move data between them. PCs, Macintosh computers, workstations, minicomputers, and mainframes can all operate under a common user interface using X.

The X Advantage

X offers many advantages as a technology for accessing host-based applications because of its hardware and vendor independence. Its flexibility allows a distributed computing scheme where applications can run on any machine and be displayed on any machine. This makes X ideal in heterogeneous (mixed) computing environments.

X supports every UNIX variant, including OSF/1, ULTRIX, HP-UX, Sun OS, Solaris, AIX, BSD, AT&T's System V, and others. X also supports proprietary operating systems such as Digital's VMS, IBM's MVS and VM, and even Microsoft's NT Advanced Server.

X was also designed to be network independent. Although X is usually used over TCP/IP, it will also run over DECnet, direct and modem connections, Ethernet, Token-Ring, and other network systems. This independence is ideal for bringing a variety of hosts together for enterprise-wide computing.

PC X servers have dramatically changed X connectivity, overtaking X terminals as the device of choice for delivering X to the desktop. PCs are now so powerful and network-capable that running two windowing systems—Microsoft Windows and X—is a viable solution, assuming the PC is a 386 or higher, equipped with 4 Mb of RAM.

The main reason organizations have chosen PC X servers as their integration system is to leverage their investment in their installed base of PCs. Users who once needed access only to word processing, spreadsheets, and other locally-based applications now often need access to mission-critical X applications, too, such as those used for process control, databases, and financial auditing, among others. Instead of adding another device to the desktop, such as an X terminal, a PC X server package can be added as a low-cost solution.

The savings on hardware and programming costs is not the only benefit. PC users are often reluctant to change their current environments. The chief advantage to users is that installing a PC X server gives them additional capabilities without having to learn a new platform. The graphical user interface of X is similar to the GUI in Microsoft Windows applications, letting PC users move easily between the two environments. A Microsoft Windows-based PC X server lets users cut and paste a graphic from an X-based drawing package and paste it into another Windows application, such as Word for Windows. The comfort factor X provides users translates into less training and support time.

Selecting a PC X Server

How can you evaluate a PC X server? Some important criteria include how well a particular X server incorporates a Windows "look and feel," what is required for installation and configuration, what connection utilities it includes, and how easy it is for users to launch client applications. Basically, a PC X server should provide the advantages typical of applications designed for Microsoft Windows, simplifying operations for users, so they don't have to master UNIX commands to work on X applications.

The Role of TCP/IP

One essential for using the X Window System for Windows-to-UNIX integration is TCP/IP. This protocol suite is the de facto standard for interoperability among dissimilar systems. TCP/IP was developed by the Department of Defense for use on its ARPANET network linking universities, research institutions, defense contractors, and federal agencies—the collection of networks that eventually became the basis for the Internet.

The TCP/IP protocol suite includes not only the TCP/IP network transport, but also many utilities, applications, and protocols, covering such diverse areas as file transfer, electronic mail, network management and remote printing.

TCP/IP is a mature technology that has been proven in innumerable implementations. Today, it has been adopted by all UNIX vendors and virtually all networking vendors. TCP/IP comes with a suite of utilities and applications for network management, troubleshooting, and connectivity. TCP/IP ships with the UNIX operating system. Yet there are many different Windows implementations of TCP/IP competing for the desktop. Finding the right solution requires some careful evaluation.

Choosing a TCP/IP stack

The best way to select a TCP/IP stack for the PC is to compare products based on memory consumption, reliability, performance, and any additional applications a particular stack provides.

Memory consumption

There are three ways to implement TCP/IP on the PC—TSRs (terminate-and-stay resident programs), DLLs (dynamic link libraries) and VxDs (virtual device drivers). Each implementation has strengths and weaknesses.

Before Windows was so widely adopted, TCP/IP stacks for DOS-based PCs were written as TSRs. Residing in real mode addressable memory (from 0 to 1024K), TSRs remain in memory until they are either manually unloaded or until the system is rebooted. As additional TSRs are loaded to run other applications, the limited memory space below 1024K becomes crowded. Users who want to run larger, more complex applications on the PC can be quite frustrated by the amount of memory that a TSR-based stack takes.

Microsoft Windows gave TCP/IP developers a way around the memory constraints imposed by DOS-based stacks. Windows-based stacks are either DLLs or VxDs, or a combination of the two. Both DLLs and VxDs are considered protected mode applications, allowing them to run above the 1024K threshold, freeing conventional memory. Anyone considering integrating Windows PCs with UNIX will want to take advantage of these Windows-based solutions. (Note that even TSR implementations can be Windows Sockets compatible, so this designation alone does not ensure that the TCP/IP implementation itself is Windows-based (a VxD or DLL design).

VT320 emulation

Some TCP/IP stacks also include VT320 emulation, but again, all implementations are not equal. Terminal-based connectivity is still the method used by 95 percent of all companies to connect to mission-critical applications. High-end VT320 emulators provides options that are just not available with rudimentary VT emulators. Sophisticated VT320 software packages not only accurately emulate VT320 terminals, they also provide advanced features such as graphical keyboard mapping, toolbars, and scripting languages that allow users easier access to host data.

Powerful TCP/IP client applications

When integrating Windows desktops, it is also important to look for a stack that includes powerful TCP/IP client applications. A great deal of added functionality can be provided by various utilities and applications that are bundled with the core protocol. Such upper-level applications provide essential tools for bridging the Windows and UNIX environments.

Important functionality to look for:

- SNMP MIB II support for network management
- LPD/LPR for remote and local printing
- FTP client and server for fast file transfers and for providing peer access to PC files
- SLIP/CSLIP for remote serial connections
- BOOTP and RARP for IP addressing capabilities

The important thing to remember about these applications is that they need to be easy to use. Most Windows users do not understand—or care to learn—command lines in either DOS or UNIX. A good graphical user interface is essential for every piece of the TCP/IP implementation. Printing to server-attached printers needs to be straightforward, and FTP will not seem friendly to a Windows user unless it's drag-and-drop file transfer. The key word is simplicity.

Conclusion

The basic issue with Windows-to-UNIX integration is the need to access the services and applications of the UNIX server from the PC. Because it is hardware- and network-independent, the X Window System provides the most versatile way of connecting a variety of dissimilar systems.

PC X servers for Microsoft Windows provide a big advantage to IS personnel, who can extend the life of legacy applications without having to port them to other operating systems. To get the most from X technology, an IS manager will want to carefully evaluate the TCP/IP software chosen for the PC, as well as the PC X server. Additional capabilities, such as VT320 emulation and TCP/IP utilities and applications, can make a tremendous difference in how smoothly Windows-to-UNIX integration is achieved—and welcomed—throughout the organization.

PAPER # 4006

**Printing with NetWare for UNIX v. 3.11
(NetWare v. 3.11 for the HP 3000 and HP 9000)**

by Cathy Gunn

**Hewlett-Packard Co., 19420 Homestead Rd.
Cupertino, CA 95014
408-447-2949**

NetWare® for UNIX® is the "Portable" version of Novell's *Native* NetWare products that runs on Intel-based systems. NetWare for UNIX provides NetWare services and transports on a multi-purpose operating system such as MPE/iX or HP-UX.

HP's NetWare for the HP 3000 and HP 9000 provide a connectivity solution for companies that want to share resources between multi-user MPE/iX or HP-UX server environments and NetWare networks.

This paper describes what NetWare print services are provided with NetWare v. 3.11 for the HP 3000 (NetWare 3000) and NetWare v. 3.11 for the HP 9000 (NetWare 9000), and how to configure host print services.

NetWare Print Services

NetWare for UNIX print services are compatible with Native NetWare print services. NetWare print utilities such as PCONSOLE, PRINTCON, and PRINTDEF as well as remote PSERVER.EXE and RPRINTER.EXE are supported. NetWare print queues are used to store print jobs until serviced by a print server.

Host print services allow NetWare clients to print to host printers, although on Native NetWare 3.1x, the PSERVER.NLM is used, while NetWare for UNIX uses the PSERVER and RPRINTER daemons.

NetWare Print Queues

Queues are used to store the print job files until the print jobs can be serviced by a print server. A queue is a subdirectory of the NetWare SYS volume, SYSTEM directory. The print files in the queue are written to disk. Once a print job file has been serviced, the file is deleted from the queue and the disk.

NetWare Print Servers

Novell has produced four different platform versions of the print server as shown in table 4006-1:

Table 4006-1

NetWare PSERVERS

Version	Description
PSERVER.EXE	Runs on a DOS workstation as a dedicated print server. Can be used with NetWare (v2.x, v3.x, v4.x and NetWare for UNIX).
PSERVER.NLM	Runs on a Native NetWare server v 3.x (NLMs are not supported on NetWare for UNIX servers.)
PSERVER.VAP	Runs on a NetWare file server running NetWare v2.1x and above or on a NetWare router (bridge) running NetWare v2.1x or above)
PSERVER Daemon	Runs on the host system over NetWare transports.

All versions of the PSERVER (listed above) are compatible with each other and can be running on the same network. Any version of the PSERVER can take a print job from a NetWare queue and spool the job to a printer.

NetWare Remote Printers

Novell has produced two versions of the remote printer (RPRINTER) as described in table 4006-2. Both RPRINTER versions are compatible with each other and can attach to the same print server.

Table 4006-2

NetWare RPRINTERS

Version	Description
RPRINTER.EXE	Runs on a DOS workstation as a TSR. NetWare clients can access the local printer (cabled to an LPT port of the workstation) as if the printer were attached to a file server.
RPRINTER Daemon	Runs on a host system that has initialized the NetWare Transports. Spools print jobs from the NetWare queues to the host system queues.

NetWare for UNIX Host Print Services

The host PSERVER and RPRINTER daemons provide host print services to NetWare clients. The SCONSOLE host utility is used to configure host printing.

PSERVER Daemon

The PSERVER daemon runs on the host system. The host system does not have to be running NetWare services, but requires the NetWare transports.

The PSERVER daemon uses the same configuration files that are common to all versions of the PSERVER. These files are created and modified with the NetWare client PCONSOLE utility and are located in the NetWare SYS volume, SYSTEM directory.

The NLM, VAP and DOS version of the print server were designed to use both a local printer facility and a remote printer facility because these versions can own the printer ports of the computers they are loaded on and the printers cabled to those ports.

The PSERVER Daemon on NetWare for UNIX was designed to use only remote printers. The PSERVER Daemon cannot use the local printer facility because it runs as an application on the host system, and does not own the printers. Therefore, all printers attached to the PSERVER daemon are printers that use the remote printing facility (RPRINTER daemon).

RPRINTER Daemon

The RPRINTER daemon can be loaded on the same host platform with the PSERVER daemon. The RPRINTER daemon is the interface into the host computer's printing system. RPRINTER spools jobs from NetWare queues into the host system queues.

Both the DOS RPRINTER and the RPRINTER daemon can attach as remote printers to the PSERVER daemon and service jobs.

Path of a Host Print Job

The path of a NetWare print job output to a host printer on the NetWare for UNIX server is shown in figure 4006-1:

1. A DOS workstation spools the job out to a local printer port (i.e., LPT1).
2. NetWare captures the job and sends the job to a queue on the NetWare for UNIX server.
3. The PSERVER daemon polls the queue. When PSERVER finds a job, it spools it out to one of its printers (using the RPRINTER daemon).
4. The RPRINTER daemon spools the print job into the host queue.
5. The host queue then spools the job to its printer for printing.

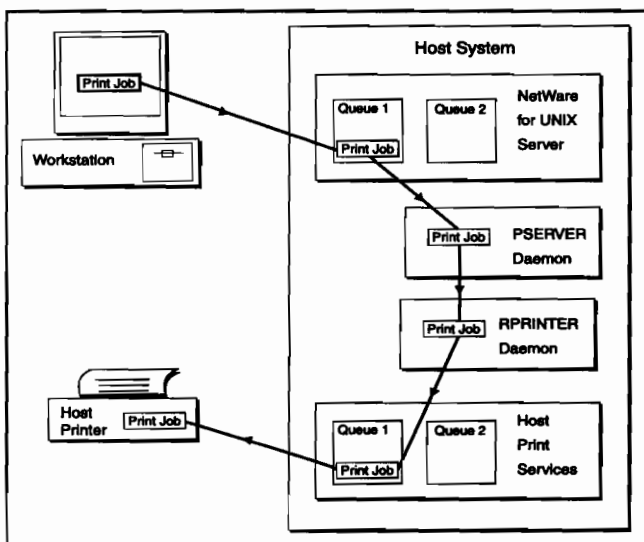


Figure 4006-1
Print Job Output to Host Printer

Host Printing Configuration Tasks

Figure 4006-2 shows an overview of set up and configuration for NetWare printing using host printers with NetWare v. 3.11 for the HP 9000 or Netware v. 3.11 for the HP 3000. For this example, it is assumed the host system is running as a file server (requiring both the NetWare transports and services).

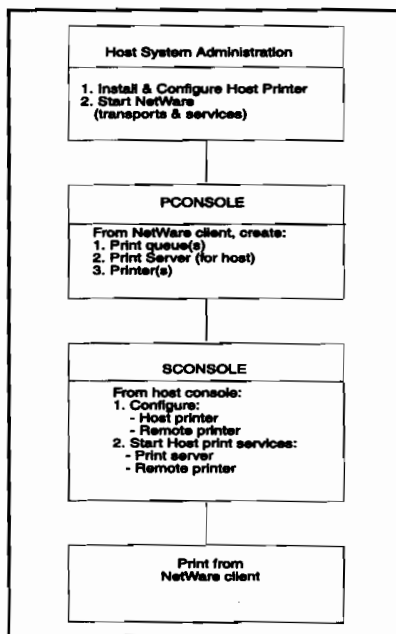


Figure 4006-2
Overview of Tasks for Configuring Host Printing

Configure Two Printers Example (NetWare 9000)

Figure 4006-3 shows a NetWare for the HP 9000 server with two host printers named **Laserjet1** and **Laserjet2** that will be configured for use by NetWare users.

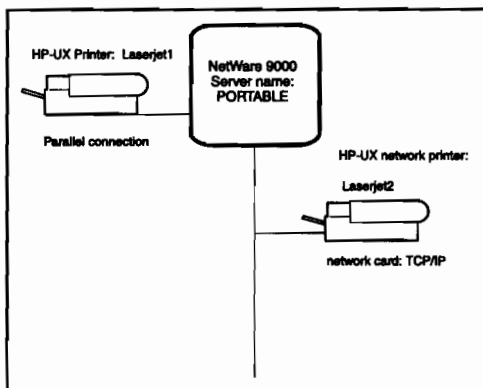


Figure 4006-3
Host Printers Example

Before you Configure Printing

Before you start the procedures described for host printing configuration, you need to do the following:

- Have already configured the host printers using the system administration utilities provided with HP-UX. For this example the printers are: **Laserjet1** and **Laserjet2**

- Create print queue names for the printers: For example:

Host Printer	NetWare Queue name
Laserjet1	LASERJET1Q
Laserjet2	LASERJET2Q

- Create a PSERVER for the host. For example: **HOST_PSERVER**.
- Create NetWare printer names associated with a slot number for each printer. This printer name is for descriptive use only. The slot number is used in both PCONSOLE and SCONSOLE for the host printer mapping to NetWare. Up to 16 printers are supported. The slot numbers are numbered 0 to 15. For example:

Printer	Printer Name	Slot Number
Laserjet1	Host_Laserjet1	0
Laserjet2	Host_Laserjet2	1

These items are used in the configuration steps that follow. The configuration requires you use two utilities. First, PCONSOLE from a NetWare client, then SCONSOLE on the host.

PCONSOLE Configuration

First, make sure your NetWare 9000 server has been started. Log onto the system as root and enter:

```
/usr/netware/bin/startnps
/usr/netware/bin/startnw
```

1. From a NetWare client, log into the NetWare 9000 server as SUPERVISOR:

```
F:\> LOGIN SUPERVISOR
```

2. Run the NetWare print utility:
F:PUBLIC\> PCONSOLE

Select "Print Queue Information"
Press the <Insert> key and type **LASERJET1Q** for the print queue name. Press the <Insert> key again and create the second print queue **LASERJET2Q**.
3. Press <Escape> key and choose "Print Server Information".
4. Press <Insert> and type the name of the host **PSERVER**:
HOST_PSERVER.
5. After creating the print server, select it (**HOST_PSERVER**) and choose:
"Print Server Configuration" then,
"Printer Configuration"
6. Select slot number 0 and type a name for the NetWare printer:
HOST_LASERJET1. This name is for description only. The slot number must match the host configuration.
7. For "Type" press <Enter> and select: Remote Other/Unknown.
Accept the default values for the other parameters.
8. Repeat steps 6 and 7 for the second printer, select slot 1 and type the printer name: **HOST_LASERJET2**.
9. Press <Escape> then yes to save changes, then press <Escape> again.
10. Select "Queues Serviced by Printer" Select the printer
HOST_LASERJET1 and select the queue **LASERJET1Q**. Enter 1 for the NetWare priority. Press <Enter>.
11. Repeat step 10 for the second printer: Select the printer
HOST_LASERJET2 and select the queue **LASERJET2Q**. Enter 1 for the NetWare priority. Press <Enter>.

12. Press the <Escape> key to back up screens and when prompted to exit PCONSOLE, highlight yes.

SCONSOLE Configuration

For the SCONSOLE configuration, you need:

- Host printer name (in SCONSOLE called: host print queue name).
Example: **Laserjet1** and **Lasjerjet2**.
 - Create a descriptive printer name (in SCONSOLE called: remote printer name) for each host printer. These names can be different than the NetWare printer name(s), but the slot numbers must match the PCONSOLE configuration. Example: **host_1j1** and **host_1j2**.
 - Items from PCONSOLE configuration: print server name: **HOST_PSERVER**, and Slot numbers for the printers: **0** and **1**.
1. Log into your system as root and run SCONSOLE:
`/usr/netware/bin/sconsole`
 2. From the SCONSOLE main menu select the following items:
 2. Configuration
 2. Services Configuration
 2. Printer Configuration
 1. Host Printer Configuration
 2. Add a Host Printer to a System

3. From the Add a Host Printer to a System screen, enter the items as prompted in the table below for the first printer, then the second printer:

Table 4006-3

Add a Host Printer Configuration

Question	For Host Printer LaserJet1	For Host Printer LaserJet2
<i>Descriptive printer name:</i>		
Remote Printer Name?	LJHOST1	LJHOST2
<i>Enter the host printer name at the prompt:</i>		
Host Print Queue Name?	LaserJet1	LaserJet2
Host Print Queue Priority?	1	1
Host Print Queue Form Name?	Press <Return>	Press <Return>

4. From the Host Printer Configuration screen, return one level to the Printer Configuration Menu and select the following items:
- 2. Remote Printer Configuration
 - 2. Add a Remote Printer

5. From the Add a Remote Printer screen, enter the information from table 4006-4 (for each printer):

Table 4006-4

Add a Remote Printer Configuration

Question	For Host Printer Laserjet1	For Host Printer Laserjet2
<i>From PCONSOLE configuration:</i> Printer Server Name?	HOST_PSERVER	HOST_PSERVER
<i>From PCONSOLE configuration:</i> Printer Slot Number?	0	1
<i>Descriptive printer name from step 3 above:</i> Remote Printer Name?	LJHOST1	LJHOST2

To enable printing, you must start the print server (PSERVER) and RPRINTER daemons:

1. From SCONSOLE, return to Main and select:
 1. Administration
 1. Start/Stop Netware
2. Assuming the transports and services for NetWare have already been started, start the host PSERVER:
 3. Start Print Server
3. Enter the following when prompted:

Enter Primary File Server name: **PORTABLE**

Enter Print Server name: **HOST_PSERVER**

Enter password to log into server **PORTABLE**: <password>

The password is the password assigned when the print server was created in PCONSOLE (if any).

4. From the START/SHUTDOWN MENU, start the remote printer (RPRINTER daemon). Select:
 4. Start Remote Printer
5. Log into a NetWare client and test printing. Print an ASCII file using NPRINT. For example:
F:\> NPRINT **filename** q=**laserjet1q**

Configuring Two Printers Example (NetWare 3000)

Configuring host printing for NetWare users with NetWare 3000 is very similar to the NetWare 9000 configuration. The configuration steps in PCONSOLE are identical. The SCONSOLE configuration items are tailored to the MPE host printer environment, and are slightly different than NetWare 9000.

NetWare 3000 provides a default (SCONSOLE) printer configuration for the host PSERVER which can service one printer:

```
NetWare Print Server: NWIX_PSERVER
MPE/iX Device Name:   lp
MPE/iX Priority:      8
Host Printer Name:    hostlp
Slot number:          0
```

Figure 4006-4 shows two host printers (lp and lpp) that can be configured for use by NetWare clients. Since one of the printers is configured as MPE/iX device name: lp, the default host printer configuration can be used for this printer.

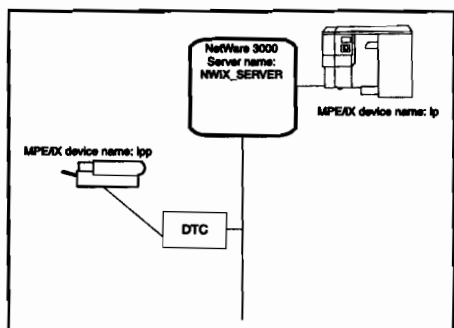


Figure 4006-4
MPE Host Printers Example

The following example uses the default printer configuration for the first printer, and also shows how to configure a second printer.

PCONSOLE Configuration

You can follow the same steps used in the previous example except:

- Make sure NetWare 3000 has been started. Log into the system as `manager.sys,netware` and use the `nw start` command to start both the NetWare server and transports:

```
:hello manager.sys,netware  
:nw start
```

- For the PCONSOLE configuration, this example uses the following values for the print server, queue names and printer names:

- NetWare print queue names for the MPE/iX devices:

MPE/iX Device Name	NetWare Queue name
lp	HOSTLPQ
lpp	HOSTLPPQ

- New PSERVER name: **HWIX_PSERVER**.
- For the NetWare printer names and slot numbers use:

MPE Device name	NetWare Printer Name	Slot Number
lp	MPE_HOSTLP	0
lpp	MPE_HOSTLPP	1

These printer names are for descriptive use only. The slot number is used in both PCONSOLE and SCONSOLE for the host printer mapping to NetWare. Up to 16 printers are supported. The slot numbers are numbered 0 to 15.

SCONSOLE Configuration

For the SCONSOLE configuration, you need:

- MPE/iX device names. For this example: **lp** and **lpp**.
- Create a descriptive printer name (in SCONSOLE called: host printer name) for each host printer. These names can be different than the NetWare printer name(s), but the slot numbers must match the PCONSOLE configuration. For example: **MPEHOST_LP** and **MPEHOST_LPP**.
- Items from the PCONSOLE configuration: print server name: **HWIX_PSERVER**, and the slot numbers for the printers: **0** and **1**.

SCONSOLE Configuration

1. Log into the system and run SCONSOLE:

```
:hello manager.sys,netware
:run sconsole
```

2. To review the default MPE printer configuration items, from Main, select:

```
2. Configuration
2. Service Configuration
2. Print Services Configuration
1. Host Printer Configuration
1. List Host Printer(s) on System
```

The default configuration shows:

```
                List Host Printer(s) on System

Host Printer Name      MPE/iX Device Name      MPE/iX output Priority
-----
mpehost_lp             lp                        8
```

Press RETURN to continue....

3. From the Host Printer Configuration menu, return one level to the Print Services Configuration menu and select:

```
2. Remote Printer Configuration
1. List Remote Printer Assignments
```

The default configuration shows:

```
                List Remote Printer Assignments

PServer Name          Printer Slot Number      Host Printer Name
-----
NWIX_PSERVER          0                        mpehost_lp
```

Press RETURN to continue....

Printing with NetWare for UNIX v. 3.11

4. Add the second printer (**lpp**) to the configuration. From the Remote Printer Configuration menu, return one level to the Print Services Configuration menu and select:

1. Host Printer Configuration
2. Add a Host Printer to a System

On the Add a Host Printer screen, enter the following values when prompted. This example assumes an environment file has been configured in MPE/iX for printer **lpp** is **env1.group.sys** (separated from the device name by a /).

Add a Host Printer to System

Host Printer Name: **mpahost_lpp**
MPE/iX Device Name: **lpp/env1.group.sys**
MPE/X Output Priority: **8**

5. From the Host Printer Configuration menu, return one level to the Print Services Configuration menu and select:

2. Remote Printer Configuration
2. Add a Remote Printer

On the Add a Remote Printer screen, enter the following values when prompted:

Add a Host Printer to System

Print Server Name: **HWIX_PSERVER**
Printer Slot number: **1**
Host Printer Name: **mpahost_lpp**

6. Follow the prompts to save your changes and exit from **SCONSOLE**.

7. To start the PSERVER and RPRINTER daemons, assuming you are logged in to the system as manager.sys,netware enter:
`:nw start print`
8. Log into a NetWare client and test printing. Print an ASCII file using NPRINT. For example:
`F: /> NPRINT filename q=hostlppq`

Trademarks:

NetWare is a registered trademark of Novell, Inc.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

This paper is derived from the book: *A Guide to NetWare for UNIX*, HP Press, Prentice-Hall, 1994, by Cathy Gunn

Paper Number 4007
Understanding and Using X Windows

Matt Hulett
Product Manager
WRQ
2815 Eastlake Avenue East
Seattle, WA 98102
206.324.0407

Handouts will be provided at time of presentation



Client-server With Your HP3000: A Case Study

David Wright

*Tufts Associated Health Plans, Inc.
333 Wyman Street
Waltham, MA 02254
(617) 466-9400*

Introduction

Today's competitive pressures and rapidly changing business needs are forcing MIS departments to develop applications faster - and with fewer resources. Users want to access and use corporate information in the same way they work with PC-based information. They are tiring of those difficult-to-use character and block mode screens and want more flexible ways to manipulate and manage information.

Our company's challenge was to find ways to fulfill these user requirements without a substantial investment in time and resources. With the support of upper management and our users, we developed our first online transaction processing system using client-server technology, and our experiences follow.

Over the past year, many articles proclaiming the benefits of client-server computing and the capabilities of IMAGE/SQL have appeared. One of the advantages of this technology is the abundance of application development products from which to choose. Conversely, selecting the appropriate hardware and software to satisfy your particular needs can be a daunting task. In our first development effort, we chose a combination of Visual Basic, Open Database Connectivity (ODBC), IMAGE/SQL and ALLBASE/SQL.

When developing flexible applications with client-server technology, you don't have to abandon your TurboIMAGE investment or convert data to a relational database. With IMAGE/SQL, it is now relatively easy to install a graphical interface to access your legacy data and empower your users with applications that increase productivity and improve effectiveness. To get started, however, you need management and user support, determination, and a carefully selected pilot project.

Selecting a Pilot Project

Tufts Associated Health Plans, Inc., is a family of innovative managed care companies providing health care coverage to more than 350,000 members in New England. We have several networked HP3000 Series 900 computers running applications written primarily in Cobol, VPlus, TurboIMAGE, and Cognos' PowerHouse.

In searching for a suitable pilot project, we wanted an application that was meaningful but relatively small in scope, not business-critical and able to withstand delays in delivery. Whenever you do something for the first time, it always takes longer than you initially anticipate. Client-server development is no exception.

After a search of our project backlog, we selected a project in which the users wanted to replace an existing PC-based application with one that had improved functionality and a "Windows look and feel". In analyzing the functional requirements of this system, we discovered that much of the data being maintained in the PC database also existed in our TurboIMAGE databases. Obviously, this was a prime candidate for IMAGE/SQL giving us the opportunity to tap interactively into our TurboIMAGE data from a PC application. Furthermore, IMAGE/SQL enabled us to transfer the remaining PC data into an ALLBASE/SQL database on our HP3000, resulting in one logical database. Our first client-server application was born.

Design Objectives

Software tools used to construct this application were selected according to their ability to meet our application design and development objectives, and to introduce us to client-server computing. In architecting this application, our principle design objectives were as follows:

1. Support an Open Architecture.

Software used in our client-server application should support industry standards and trends while minimizing the reliance on proprietary software solutions; for example, use of popular programming languages, relational data access techniques, and industry standard Application Programming Interfaces (APIs). Proprietary components and vendor-specific extensions were avoided wherever possible.

2. Integration of Data.

Data integration was a critical concern and distributed data presented many technical issues that we chose to avoid. Distributed transactions, database locking, and performance issues were minimized by locating all data sources on our familiar and reliable MPE/iX operating system.

Since this application needed a database in which to store new data, ALLBASE/SQL in combination with IMAGE/SQL provided an ideal solution. The union of these two databases enabled us to use our TurboIMAGE data without copying to a relational environment in order to implement a client-server application. Furthermore, this solution melded two disparate databases into one logical database with which the application developer could easily work.

3. Avoid Duplication of Data.

A convenient and oftentimes expedient solution to building a new application is to create an application-specific database. Doing so frequently results in duplication of data. Our objective was to use existing data in its native location rather than creating a reformatted copy for the purposes of a new application, as was done in the past.

4. Leverage Existing Computing Investments.

Wherever possible, existing investments in hardware and software were used to facilitate our migration to client-server computing. A hybrid solution, which combined current practices with new technology, provided a gradual transition to client-server computing. Moreover, it enabled us to leverage our current investments while becoming experienced with the new technology. For example, using IMAGE/SQL and ALLBASE/SQL on our HP3000 enabled us to take advantage of our existing skill sets in PowerHouse and Cobol. These tools could be used to integrate the new relational database with our existing TurboIMAGE-based applications.

5. Use of Graphical User Interfaces (GUIs).

The prevailing standard for interactive access to computer applications is via a graphical user interface. GUIs provide features that enable users to interact more easily and intuitively with their application, and there is a vast array of products from which to choose when building graphically oriented applications. Also, our users wanted interfaces to their business applications that were similar to their office products (e.g., Word & Excel).

6. Minimize Programming Effort.

Visual programming tools were used to expedite the Windows applications development process. Use of Visual Basic enabled us to develop the application faster while avoiding complex, low level, Windows programming tasks. In addition, industry-standard SQL was used in lieu of proprietary database access techniques. These approaches simplified programming and helped shield our application developers from operating system, networking, and database complexities.

7. Limit the number of software products.

Client-server applications become increasingly complex as more software products are introduced. An objective, therefore, was to limit the number of vendors and software layers that could contribute to unnecessary complexity. Additional software layers introduce more points of failure, contribute to performance slowdowns, and add to the complexity of software integration and support.

The Application

A department within our company was using a PC application to manage its database of health care providers. This system supported various query capabilities, as well as the production of camera-ready provider directories that were sent off-site for volume printing.

Some of the deficiencies and limitations of this system were:

1. Required extensive updating of redundant data
2. Consisted of data that was inaccessible to other departments
3. Lacked proper data validation procedures
4. Was cumbersome to operate
5. Contained data that was inconsistent with our TurboIMAGE data
6. Imposed limits on the storage of data
7. Did not have adequate security features

Until this project started, IMAGE/SQL and ALLBASE/SQL were new to us. We realized, however, that open systems and relational technology were becoming significant factors in supporting the information technology needs of our business effectively.

The new application was designed to read data from our TurboIMAGE databases while storing information in a new ALLBASE/SQL database. Rather than use ALLBASE/SQL to store new data, we had the option of adding datasets to our TurboIMAGE database. However, we chose to use the inherent benefits of ALLBASE/SQL and to avoid some of the current limitations of IMAGE/SQL (e.g., row-level locking, utilizing third-party indices, and database constraints).

Figure 1 provides a list of some of the advantages of using relational database technology. We foresee that over time most of our TurboIMAGE data will need relational access and the combination of IMAGE/SQL and ALLBASE/SQL appear to address this long-term requirement.

Figure 1. Relational Databases

- Dynamic database changes (e.g., add/drop tables, indices, fields, security, constraints, space, etc.) can be made easily and while the database is in use.
- Industry standard data access and manipulation language (SQL). SQL is relatively easy to learn and use, supports set versus record manipulation, is flexible and very powerful.
- Data types: variable character, float, decimal, date, time, binary, binary large objects (BLOBs).
- Native data manipulation capabilities. Import/export data, transfer and reorganize data within the database, etc.
- Database views provide logical views of the database. For example, predefined table joins which simplify data access, views which mask data, and derived/calculated columns.
- Various indexing techniques are available (hashed and B-tree indices). IMAGE/SQL B-tree indexing is only available through the use of third-party products (e.g., OMNIDEX).
- Data concurrency (locking) is handled automatically based on the type of table - no programming is necessary. Database, table, and row level locking is available.
- Database space can be increased automatically. Manual table enlargement procedures are not needed.
- Database constraints. For example, referential integrity constraints prevent rows from being added to a table until a corresponding key value exists in a referencing table, unique constraints enforce unique key values within a table, and check constraints verify certain criteria - such as a valid date - before a row can be added or updated.
- COMMIT and ROLLBACK capabilities. Logical transactions are controlled using these two SQL statements. With TurboIMAGE, each field/record is updated when an update command is issued. Using SQL, updated data is not actually available to other users until a COMMIT WORK statement is issued. ROLLBACK WORK discards all updates since the last COMMIT WORK was issued.
- Query optimization is used to find the most efficient means of accessing the requested data. Having to specify the access path is not necessary.
- Database security can be defined and changed dynamically, even while the database is in use.
- Stored procedures and rules can be centralized within the database.

Advantages of ALLBASE/SQL:

- Native HP relational database (HP3000 & HP9000)
- ALLBASE/SQL, via IMAGE/SQL, represents the only relational interface to TurboIMAGE data.
- ALLBASE/SQL data can be located with other legacy data thereby improving integration.

Creating a New Application Architecture

Application partitioning is the process of segmenting an application into discrete components that enable them to be executed independently and, frequently, on separate computing platforms. A typical client/server application contains three primary layers.

1. Presentation
2. Application Logic
3. Data Access

Our application was developed using a two-tiered client-server architecture, comprised of presentation and application logic residing on the client, with the server responsible for data access. Software development tools selected for this application were particularly suited to a two-tiered architecture.

A three-tiered architecture typically requires software development on both the client and the server. Since this approach introduces many additional complexities, such as remote procedure calls, we chose to avoid it in our first client-server development effort.

Visual Basic and the Front-end

Microsoft Visual Basic is a powerful development environment that exploits the key features of graphical applications development. It has quickly become the most popular client-server development tool for developing Windows applications. Using Visual Basic, full-featured Windows applications can be built with relative ease, as compared to writing Windows programs in C.

Visual Basic (VB) contains so many built-in features that creating a program is more like arranging a set of objects and events. With VB, you can write efficient programs that are every bit as professional as commercial applications. Furthermore, executable VB programs do not require licensing fees, since Microsoft permits them to be distributed without cost.

For this project, VB comprised the presentation and application logic layers of the application. Database access logic consisted of SQL statements initiated from VB and submitted to the server for processing. Our HP3000 acted as a database server by processing these statements and returning the results to VB for display and further manipulation.

Middleware and Open Database Connectivity (ODBC)

The term middleware describes a vast array of software that enables client and server processes to communicate with each other. It can be referred to as anything that is not part of the client or server application. This application

component is critical in forming a link between computing resources, networks, databases, and operating systems. Middleware, critical to the success of any client-server application, generally consists of APIs, drivers, gateways, translators, and networking software.

Microsoft ODBC, a middleware component, is a database access specification based on SQL. It provides a set of functions linking Microsoft Windows applications to back-end databases. This database interface enables an ODBC-compliant Windows application to access databases that support the ODBC specification.

Currently, two database APIs are available from HP to access ALLBASE/SQL and IMAGE/SQL data from a PC: ODBC and the Gupta C API. Since a third-party product is needed to enable Visual Basic to work with the Gupta API, we selected ODBC for our application. Furthermore, ODBC has gained widespread acceptance as the de facto standard for Windows database access since it provides a transparent interface to a growing list of disparate databases.

HP3000 Database Server

IMAGE/SQL provides relational read and write access to TurboIMAGE data using industry-standard SQL. IMAGE/SQL includes TurboIMAGE, ALLBASE/SQL, and a database administration tool that links them together. This combination of software is marketed as IMAGE/SQL, although the underlying database is TurboIMAGE.

In order to obtain relational access to TurboIMAGE, an ALLBASE/SQL database must be built and your TurboIMAGE database must be "attached" to it. Building an ALLBASE/SQL database requires the creation of an ALLBASE/SQL DBEnvironment (DBE). A DBE is a collection of files that contain configuration information, system tables, user tables, indices, authorities, log files, etc., that support the ALLBASE/SQL database as well as relational access to TurboIMAGE. In effect, HP has used the ALLBASE/SQL infrastructure as the interface to TurboIMAGE; in my opinion, a very clever idea.

Attaching and Detaching the Database

A database administration tool provided by HP, called IMAGESQL, loads information about a TurboIMAGE database into a new or existing ALLBASE/SQL DBE. This process is referred to as an "attach" of the TurboIMAGE database to an ALLBASE/SQL DBE. Each TurboIMAGE database that requires relational access must be attached to a DBE. During the attach process, dataset names are mapped to ALLBASE/SQL tables, field names are mapped to column names, data types are mapped to ALLBASE/SQL data

types, and security is mapped to ALLBASE/SQL views. This information supports the translation of SQL statements to TurboIMAGE database calls. TurboIMAGE mapped tables contain no data and facilitate only the interface to a TurboIMAGE database. This same DBE may also contain ALLBASE/SQL tables with user data.

Before TurboIMAGE structural information can be loaded into a DBE, slight alterations must be made in order to conform to SQL syntax rules. Certain characters used in dataset names and field names (specifically, + - * / ? ' % &) are automatically converted to an underscore (_). For example, a TurboIMAGE dataset name CLAIM-DETAIL would be converted to CLAIM_DETAIL and the field YES/NO? would be changed to YES_NO_. TurboIMAGE arrays are also converted to separate column names within an ALLBASE/SQL table. For example, a TurboIMAGE array called ARRAY(4) would be mapped into four separate columns; ARRAY_1, ARRAY_2, ARRAY_3, ARRAY_4.

A TurboIMAGE database must be detached from the DBE before structural changes are made. For example, before adding or removing fields, paths, or datasets, the database must be detached from the DBE. This is necessary because the ALLBASE/SQL DBE is not dynamically updated when structural changes are made to the TurboIMAGE database. Capacity changes do not require detaching from the DBE.

When a TurboIMAGE database is detached from a DBE, relational access is no longer available to that database because all information regarding mapped tables is dropped from the DBE. Also, all customized mapping information (e.g., new users and split columns) is dropped and must be added when the database is reattached. We maintain an ISQL script file that contains customized SQL statements. It is executed each time the TurboIMAGE database is reattached to the DBE.

Setting up the Environment

Figures 2 and 3 list the operating environment that supported this application. Once we had the appropriate combination of software and configuration settings resolved, the project proceeded without any notable setbacks. Setting up the remaining components was quite simple once we understood what had to be done. A brief summary of the remaining setup procedures follows.

Using IMAGESQL.PUB.SYS, we attached our TurboIMAGE database to a new ALLBASE/SQL DBE. Our database was structurally very large and we had to enlarge the log file size with SQLUTIL. Once the database was attached, a series of ODBC database tables and views were added to our DBE. ODBC views were

loaded once by copying the views file, supplied by HP, to the server, and installed using HP's interactive SQL utility (ISQL).

Figure 2. Operating Environment

Client

A 386/33mhz PC with 8MB RAM, DOS 5.0 and Windows 3.1.

HP PC API ODBC Driver (version X.G0.03).

Provided by HP, this software enables a front-end SQL application, in this case Visual Basic, to communicate with the back-end ALLBASE/SQL and IMAGE/SQL databases.

WRQ 3000 Connection (version 2.01).

TCP/IP software from WRQ provides the communication protocol between the PC and the HP3000. A network operating system is not needed by this application. By the time this article reaches publication, a DLL version of the WRQ 3000 connection (Reflection Network Series for Windows - RNS/WIN 4.0) should be available.

Our VB developers used Visual Basic Professional Edition 3.0.

HP3000 File Server:

MPE/iX 4.0

TurboIMAGE (version C.04.19).

IMAGE/SQL (version B.F0.25).

ALLBASE/SQL (version A.G0.21)

HP ThinLAN 3000/iX

We found that a few additional configuration settings were necessary and these are listed below.

1. Using SQLUTIL, set the Maximum Time-out and Default Time-out for the DBE to 30 seconds. If a database access is waiting on a lock, it will not wait longer than 30 seconds. Otherwise, you need to terminate the user from the DBE; rebooting does not clear the lock.
2. Since IMAGE/SQL applies set-level locks, we set the DBE DBOPEN mode to 5 using IMAGESQL. With this setting, no TurboIMAGE locks are applied. Consequently, SQL updates can not be made to the database.
3. Isolation levels allow you to control the degree of concurrency by regulating the extent to which one user can affect another user in a multi-user

environment. We set the isolation level in our ODBC.INI file to Read Committed (RC) - see Figure 4. RC holds a read-level lock until the transaction is committed. Read Uncommitted holds no locks and is not recommended in an update environment. We experienced several locking problems when attempting to use the two other isolation levels: Repeatable Read (RR) and Cursor Stability (CS).

Figure 3. Operating Environment Matrix

Platform	Application Layer	Software Components	Files
PC	Presentation and Application Logic	Microsoft Visual Basic	EXE, VBX, ...
	Database API (Middleware)	Microsoft ODBC Driver	ODBC.DLL ODBC.INI
		HP ODBC ALLBASE/SQL Driver	ALLBASE.DLL
	Network Protocol (Middleware)	3000 Connection from WRQ (TCP/IP)	WRQTCP.EXE WRQHPSO.EXE WSOCKETS.DLL ...
HP3000	Network Protocol (Middleware)	ThinLAN 3000/iX (TCP/IP)	
	Data Access	ALLBASE/SQL (SQL Interpreter, Query Optimizer, and data access routines)	
	Data	ALLBASE/SQL & IMAGE/SQL	Database Files

In our development efforts we discovered rather quickly that VB's data controls impacted application response negatively. By using a few diagnostic tools, described later in the troubleshooting section, we could examine VB's extensive dialogue with the server. This dialogue is generated by Microsoft's Access

database engine, integrated with VB, in an attempt to simplify VB programming. Data controls work fine with Access. Unfortunately, their built-in features resulted in extensive overhead causing poor performance with ALLBASE/SQL and IMAGE/SQL.

To resolve the data control problem, we entertained the idea of using a third-party product to make the connection between VB and HP's PC API. We quickly discounted this idea based on our objective to limit the number of software layers and vendor products. Our desire was to find the shortest route to the database while keeping programming effort and interfacing issues to a minimum.

Figure 4. ODBC.INI file:

```
[ODBC Data Sources]
PARTSDBE=HP ALLBASE/SQL

[PARTSDBE]
Driver=C:\WINDOWS\SYSTEM\ALLBASE.DLL
Description=HP PARTSDBE Database
LastUser=#MPEIX/HP3000:PARTSDBE.DBE,SK#DAVE,USER.DEV
DefaultIsolation=RC

[Default]
Driver=C:\WINDOWS\SYSTEM\ALLBASE.DLL
Description=ODBC Default Connection
LastUser=#MPEIX/sysname:db.e.group.account,sk#sessid,user.account
DefaultIsolation=RC
```

Secondly, we considered using direct ODBC function calls. However, this solution would have required purchasing Microsoft's ODBC Software Development Kit (SDK) and considerably more programming. A middle-of-the-road solution was developed using a technique called SQL pass-through. Basically, this method requires the developer to prepare each SQL statement within VB and pass it through the Access engine, untouched, to the ALLBASE/SQL database driver. Performance was much improved.

We distributed the application to each user by creating a setup disk with the Visual Basic Application Setup Wizard. Each user had the application loaded on his or her PC which eliminated our dependence on our network operating system (i.e., NetWare).

Sample Program

A sample VB program - see Figures 5 and 6 - was written to demonstrate how easy it is to develop a GUI front-end to your HP3000 data. Its primary purpose is to assist you in understanding how the data access process works. A copy of this program is available on CompuServe as a PKZipped file in the HP Systems Forum, General/Uploads library section. If you wish to use it, create an ALLBASE/SQL PARTSDBE database on your system. Assuming you have ALLBASE/SQL or IMAGE/SQL, creating this DBE (about 12,000 sectors) is as simple as this: 1. build a group in which to store the DBE, 2. CHGROUP to the new group, 3. type SQLSETUP.SAMPLEDB.SYS to create the new DBE (enter option 1), and (4) load the ODBC views using ISQL. Once the DBE is created, you can run ISQL to peruse the database. For example:

```
:ISQL
isql=> CONNECT TO 'PARTSDBE';
isql=> SELECT * FROM PURCHDB.VENDORS;
```

You can also attach a TurboIMAGE database to this DBE and query it using ISQL. This program can be easily modified to access your TurboIMAGE data provided you first set up the DBE and then modify the SQL statements to reference a TurboIMAGE table of your choice.

Figure 5. Sample Screen

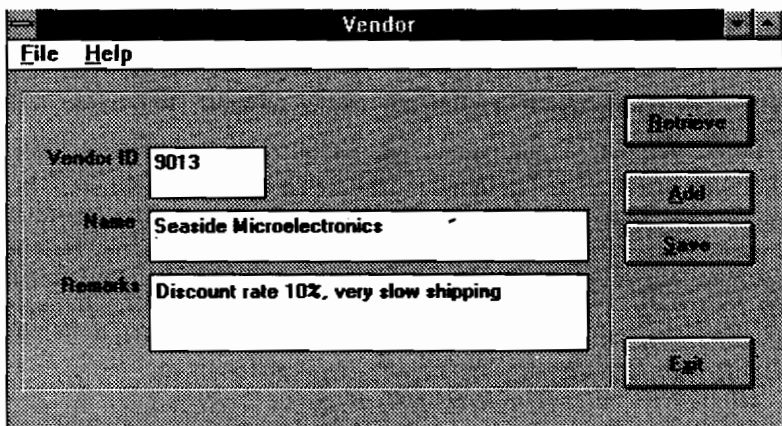


Figure 6. Sample Visual Basic Program

```
Option Explicit
Dim DB As Database
Dim DS As Dynaset
Dim sSQL As String
Dim nRows As Integer
Dim bAddNewRecord As Integer

Sub cmdAdd_Click ()
    cmdRetrieve.Enabled = False
    cmdAdd.Enabled = False
    cmdExit.Enabled = False
    cmdCancel.Visible = True
    txtVendorNumber = " "
    txtVendorName = " "
    txtVendorRemarks = " "
    txtVendorNumber.SetFocus
    bAddNewRecord = True
End Sub

Sub cmdCancel_Click ()
    cmdRetrieve.Enabled = True
    cmdAdd.Enabled = True
    cmdExit.Enabled = True
    cmdCancel.Visible = False
    txtVendorNumber.SetFocus
End Sub

Sub cmdExit_Click ()
    End
End Sub

Sub cmdRetrieve_Click ()
    mousePointer = 11

' Prepare SQL statement
sSQL = "select vendornumber, vendorname, vendorremarks from purchdb.vendors "
sSQL = sSQL + "where vendornumber = " + txtVendorNumber

' Execute SQL statement. Results returned to Dynaset.
Set DS = DB.CreateDynaset(sSQL, 64)

mousePointer = 0
If DS.RecordCount = 0 Then
```

```

    MsgBox "Vendor Not on File"
    Exit Sub
End If

txtVendorName.Text = DS("vendormname")
txtVendorRemarks.Text = DS("vendorremarks")
txtVendorNumber.SetFocus
End Sub

Sub cmdSave_Click ()
' Turn on hourglass.
    mousePointer = 11

If bAddNewRecord = False Then

' Prepare SQL statement.
    sSQL = "update purchdb.vendors set "
    sSQL = sSQL + "VendorName = " + txtVendorName + ", "
    sSQL = sSQL + "VendorRemarks = " + txtVendorRemarks + " "
    sSQL = sSQL + "where VendorNumber = " + txtVendorNumber

' Execute SQL statement and commit.
    nRows = DB.ExecuteSQL(sSQL)

    GoTo Done
End If

' Prepare SQL statement.
    sSQL = "select vendornumber from purchdb.vendors "
    sSQL = sSQL + "where vendornumber = " + txtVendorNumber

' Check if record already exists.
    Set DS = DB.CreateDynaset(sSQL, 64)

If DS.RecordCount > 0 Then
    MsgBox "Vendor Is Already On File"
    txtVendorNumber.SetFocus
    GoTo ExitSub
End If

' Prepare SQL statement.
    sSQL = "insert into purchdb.vendors values ("
    sSQL = sSQL + txtVendorNumber + ", "
    sSQL = sSQL + """" + txtVendorName + """, "
    sSQL = sSQL + ""';';';';';';';'"

```

```
sSQL = sSQL + "" + txtVendorRemarks + ""
```

```
' Execute SQL statement that updates database.  
nRows = DB.ExecuteSQL(sSQL)
```

Done:

```
cmdRetrieve.Enabled = True  
cmdAdd.Enabled = True  
cmdExit.Enabled = True  
cmdCancel.Visible = False  
txtVendorNumber.SetFocus  
bAddNewRecord = False
```

ExitSub:

```
' Turn off hourglass.  
mousePointer = 1  
End Sub
```

Sub Form_Load ()

```
Set DB = OpenDatabase("", False, False, "ODBC;DSN=PARTSDBE;UID=")  
End Sub
```

How the Application Works

The application is invoked from the user's PC by clicking on a Windows icon. Once it is loaded, a connection is made to the HP3000 server. Connecting to the server is accomplished automatically when a database is opened. By calling the VB OpenDatabase function, Microsoft's ODBC driver manager (ODBC.DLL) searches your ODBC.INI file (see Figure 4) for the specified data source name (e.g., PARTSDBE). When the data source is found, the associated database driver is loaded - in this case ALLBASE.DLL - and a connection to the server is initiated.

The ALLBASE/SQL ODBC driver, supplied by HP, is a Windows Dynamic-link Library (DLL) located in the WINDOWS\SYSTEM directory on the user's PC. This driver translates ODBC requests from our VB application into the appropriate calls for the ALLBASE/SQL database engine and returns the result back to the application via the ODBC driver manager.

Connecting to a server is the responsibility of the ALLBASE/SQL ODBC driver. It initiates a sockets connection to our HP3000 as long as the HPDARPA_MANAGER.SYS batch job is running. This batch job is initiated by typing ANSTART ARPA from MANAGER.SYS. It is a listener process that services the entire system. Its function is to listen for a connection request from a

client and then spawn a database server process. The server process connects to the database and services all requests from the client. The server process runs under the listener process.

Unlike HP's original connection architecture that initiated a separate batch job for each client connection, this process uses one batch job to service all client connections. In our environment, the original technique was not viable since each client connection consumed a batch job slot. One downside to the new connection architecture - available only with ALLBASE/SQL G0 - is the inability to determine who is connected to the server. The SHOWPROC command can be used to indicate the number of client connections but it does not show who they are or when they connected.

Once the connection is established, the VB application communicates with the database server via SQL statements. ALLBASE/SQL processes these statements and returns either a result set (list of data) or an update status.

As illustrated in Figure 6, two techniques are used to submit SQL statements to the database. A SELECT statement is initiated via the CreateDynaset method in VB. In this situation, a Dynaset is used to store the results of your query. SQL statements that change data are initiated via the ExecuteSQL method. ExecuteSQL does not return data and can not be initiated with the CreateDynaset method. Both techniques use SQL pass-through (option 64) which passes the SQL statement to the ODBC database driver. With these two methods, you can read and write ALLBASE/SQL and TurboIMAGE data.

With respect to software development on the HP3000, none was needed to support our application. From an application's perspective, the HP3000 functioned entirely as a database server.

Troubleshooting

Troubleshooting problems in a client-server environment can be quite challenging. Without the proper tools and an organized way of isolating a problem, you may be unable to get the answers you need. There are many software and hardware components and each one can affect the operation of your application.

Throughout development, we tested our SQL statements first with ISQL on the server to see how they performed. Many times we had to change them to improve performance. Second, we tested them in our VB application and if that didn't suffice we would resort to a few PC utilities to identify the bottleneck/problem. A tool we found very helpful and obtained from Microsoft's ODBC SDK is called ODBC Test; also referred to as Gator. It enables you to

test the ODBC interface separate from your application. By initiating SQL statements from within this program, you can determine whether the application is causing the problem or whether it is in the ODBC, or middleware, layers.

If you need a more detailed look at the conversation being conducted between the client and server, an ODBC tracing tool called DRDBSP (Dr. DeeBee Spy) can be used to capture this information to a log file for your review. By initiating this program before you start your application, it will log all ODBC calls to a log file for later inspection. Another similar tool is the HPDCRECORD feature which records ALLBASE/SQL messages sent and received across the network. It, also, records information in a log file which must be converted before reading. DRDBSP and HPDCRECORD are included on the ODBC setup from HP. In addition, an ALLBASE/SQL utility called SQLMON was helpful in monitoring DBE locks, table capacities and various other status-related information.

The above and other topics are described in more detail in the HP PC API User's Guide for ALLBASE/SQL and IMAGE/SQL. I highly recommend reading it from cover to cover.

There is one more troubleshooting tip I'd like to mention. We found the use of CompuServe extremely helpful in getting answers to our questions from HP (HP Forum), Microsoft (MS BASIC Forum), and various other users of the network. It is also a great place to exchange information and ideas. If you don't have a CompuServe account, I recommend getting one before starting your client-server project.

Additional Points of Interest

Infrastructure

A few assumptions were made regarding hardware and software necessary to implement a client-server application. Don't even consider installing your application without a stable and reliable network in place. Ensure that you have the necessary networking and CPU capacity before implementing your application.

Prototyping

Since this was our first Windows-based client-server project, we constructed a prototype that helped demonstrate the features and functionality of a graphically oriented system. It was essential that our users understood the nature of a windows event-driven application. Our VB prototype served the purpose and it also formed the foundation of the final product. In addition, it served as a proof-of-concept exercise and a means of testing various supporting software components.

Training

Fortunately, we had a few experienced VB programmers available for this project. If your staff is unfamiliar with Windows applications development, send them to a VB course. It is money and time well spent. VB is easy to learn, but picking up a manual and expecting someone to develop an application may be setting an unfair expectation. Refer to Figure 7 for a list of additional reference material.

Since our developers were unfamiliar with IMAGESQL, ALLBASE/SQL, and ODBC, HP was very helpful in providing us with the information and support we needed. Read the HP manuals. Most of them are well written and contain a wealth of information. Also, use the HP Response Center if you have questions or need help.

Other members in our MIS department also needed training in order to support the application once it was installed. Make sure your PC, networking, Operations, and Technical groups have been educated to support the application.

Support

Don't forget about database backups, capacity checking, and database maintenance. These issues are often overlooked in client-server projects because applications of this type are considered easy to install and maintain. Administrative responsibilities must be determined before you implement.

Documentation

Various forms of documentation are needed for your first client-server project. Others expected to follow in your footsteps will need a map to guide them. We produced a functional specification complete with screen prints (from our prototype) and functional descriptions, user and administration guides, on-line help, and a technical specification that described the application in detail.

A product called RoboHELP (from Blue Sky Software Corp., La Jolla, CA 619-459-6365) was used to simplify the process of creating and maintaining a Windows Help file. This product integrated with Microsoft Word and facilitated the creation and integration of an online help subsystem.

Test, Test, Test

I can't emphasize this phase of the project enough. Don't underestimate the impact your client-server application can have on your client, network, and server. Test the application in a setting as close to your production environment as possible. For example, do concurrency testing, reboot your PC, shut down the DBE, volume test; in short, try to find the weak links in the client-server chain.

Remember, a client-server application works only as well as its least effective component.

Communication

Keep others in your company apprised of your progress. Internal marketing is helpful since some may view this technology as a means to render their expertise obsolete. Conduct internal training seminars and application demonstrations. Everyone can benefit and being informed can provide comfort to those not directly involved.

Feedback to HP

HP has learned that listening and responding to its customers have made it a dominant player in the computing market. If you are not satisfied with something, let them know. We found that working with the Response Center, and in some cases "the factory", we got answers to our questions and enhancements to the software. Feedback works and HP is listening.

Summary

Client-server applications are able to off load processing from host computing systems by distributing tasks to other processors. This improved division of labor enables each computing platform to do what it does best. Consequently, developing a client-server application with your HP3000 and your PC workstations has been greatly simplified with the availability of IMAGE/SQL.

If your company is like ours, we have a large investment in our TurboIMAGE data. However, in order to take advantage of the leading client-server technologies available today, relational access to our legacy data is necessary. Migrating this data to a relational structure is no longer necessary. IMAGE/SQL has made TurboIMAGE data accessible to the abundance of front-end tools that facilitate your client-server development efforts. Give it a try. You'll be impressed at what can be accomplished with these new capabilities and how your users and developers will respond to this evolving method of applications development and desktop integration.

Figure 7. Reference Materials

HP:

Getting Started With HP IMAGE/SQL
HP IMAGE/SQL Administration Guide
HP PC API User's Guide for ALLBASE/SQL and IMAGE/SQL
Up and Running with ALLBASE/SQL

Windows:

Windows Design: The Windows Interface - An Application Design Guide from Microsoft Press; ISBN 1-55615-439-9. This book provides guidelines for developing user interfaces for applications that run in the Windows environment. It describes the components of the Windows user interface and explains design principles for software developers of Windows-based applications.

The Visual Guide to Visual Basic For Windows, Ventana Press; ISBN 1-56604-063-9

Visual Basic How-To, Waite Group Press; ISBN 1-878739-42-5

Visual Basic Programmer's Guide to the Windows API, ZD Press; ISBN 1-56276-073-4.

Paper 4009
Processing Multiple Inputs With The X Toolkit Intrinsic

Donald J. Parker
Software Engineering Specialist
E-Systems, Inc.
P.O. Box 6056; CBN 190
Greenville, TX 75403-6056
903/457-7770

ABSTRACT

It is a well known and documented fact that the X Toolkit Intrinsic (Xt) provides a very good framework for building Graphical User Interface (GUI) applications. But what may not be known is that it also provides a very good foundation for developing applications that process data from other sources of input such as external devices, networks, and/or other applications. This paper will explore the design of the X Toolkit and demonstrate how it is ideally suited for applications that must react to unpredictable external events. It will analyze the requirements and design of a multi-input application and, with an example program, describe how the X Toolkit can be used to develop an application that reacts not only to GUI events, but also to events from other external sources. Although expert knowledge is not required, the paper does assume the reader is familiar with the basic concepts of the Xt Intrinsic and how it is used to build GUI applications.

INTRODUCTION

Almost all applications require some sort of input data in order to accomplish the task they were designed to do. Simple programs may only require one source of input, such as a data file, to do their job. More complicated applications may need to process data from a multitude of different sources, each in its own unique format. Input data can come from a variety of sources including Graphical User Interfaces, external devices (terminals, disks, tapes, networks, etc.), other applications (message queues, pipes, sockets), or from within the operating system (signals, interval timers). Multi-input applications generally fall into two categories: synchronous and asynchronous. Synchronous applications are sequential in nature and generally process input data one source at a time. This sort of application is fairly straightforward to implement because the logic within the program is predictable and usually flows in a "top to bottom" fashion. An asynchronous application, on the other hand, is one that monitors several sources of input and reacts to whatever data is available. The flow within an application of this type is very unpredictable and is said to be *event driven* because the flow is driven by input events. Designing an application of this nature presents more challenges than the sequential model because of the additional complexities involved in handling multiple sources of input simultaneously. Although there are several methods that could be used to solve this problem, this paper will show how the X Toolkit Intrinsic can be used to provide a very elegant solution to the problem.

DESIGN OF THE X TOOLKIT INTRINSICS

The primary purpose of the X Toolkit Intrinsic is to provide a simplified approach to programming GUI applications. In this capacity, it serves as a framework that allows programmers to create user interfaces by combining an extensible set of user interface components. These components are known as *widgets*, which consist of an X window along with some procedures that operate on the window. The Intrinsic provides a small core set of widgets,

but defines an architectural model for widgets that also allows programmers to extend the Toolkit by creating new types of widgets.

The X Toolkit provides application programmers with a specific model for writing applications. All X applications are normally designed to be event-driven. However, the X Toolkit builds on this event model to create a *dispatch-driven* programming model. Xlib applications usually have a switch statement inside the event loop. The `switch` statement looks at the type of each event and performs some action based on the information in the event. If the application uses multiple windows, the program also has to determine in which window the event occurred and take that into consideration. For applications with many windows, the `switch` statement can become quite long and very complex. The X Toolkit hides this process and dispatches all events to the appropriate user interface components. So instead of writing one large switch statement to handle the logic of the program, programmers who use the X Toolkit write small procedures to deal with specific events that occur within each component. This approach allows programs to be written in a much more declarative style (when A happens, do B) that greatly simplifies the logic of most applications[1]. With this model, most Xt applications follow a similar structure and perform several basic steps. These are:

- Initialize the Xt Intrinsic \Rightarrow `XtAppInitialize(3X)`
- Create widgets \Rightarrow `XtCreateWidget(3X)`
- Register callbacks \Rightarrow `XtAddCallback(3X)`
- Realize widgets \Rightarrow `XtRealizeWidget(3X)`
- Enter the event loop \Rightarrow `XtAppMainLoop(3X)`

Most Xt applications are completely event-driven, and are designed to loop indefinitely responding to events. The dispatch-driven architecture of the X Toolkit makes it easy for an application to process these events. To receive events, an application simply registers callback procedures for the applicable widgets. The event loop then retrieves events from the X event queue and dispatches them to the appropriate callback procedures associated with the widget in which the event occurred. This architecture is illustrated in Figure 1.

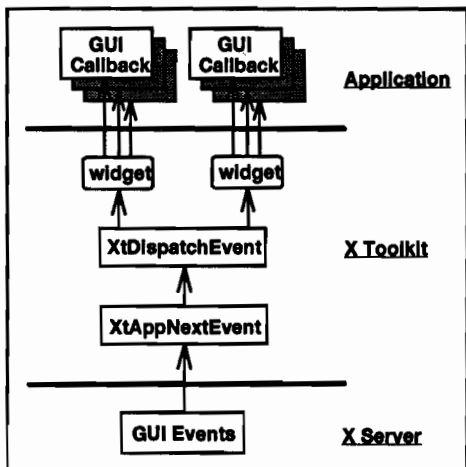


Figure 1. Dispatch-Driven Architecture of X Toolkit

An interesting feature of the X Toolkit is that way down at the lowest level of the event processing loop, GUI events are considered to be nothing more than packets of information from a source of input (the X Server). The internals of the `XtAppNextEvent(3X)` procedure is designed so that it has the capability to process input from not only the X Server, but other sources of input as well. The magic that makes this possible is provided by the `UNIXTM` `select(2)` system call, which has the unique ability to monitor multiple sources of input. Therefore even though the X Toolkit is used primarily to process GUI events, the dynamic event-driven architecture that it is built upon makes it ideally suited for an application that processes data from multiple sources of asynchronous input.

SOURCES OF ASYNCHRONOUS INPUT

With the plethora of inputs available in today's modern computer environments, it may seem like an overwhelming task to design an application that is able to multiplex data from many different sources. But after careful examination, the sources of asynchronous input can be generalized into the following basic categories:

- GUI Events
- Timer Events
- Idle Events
- Inter Process Communication (IPC) Events

GUI Events are those events generated by the person interacting with the application, such as clicking a button on the mouse or entering text from the keyboard. Timer Events are those events generated when a specific time has been reached or a certain amount of time has elapsed. These type of events are often used by an application to perform operations that must be done on a periodic basis or to time-out a previously initiated action. Idle Events are those events that occur while the application is waiting for input; these events allow an application to do background processing during the "dead" time between other external events. IPC Events are those events generated by external devices or other applications. This is a fairly broad category that includes not only the *intra*-host communication that occurs between programs on the same computer, but also the *inter*-host communication that occurs between programs on different computers connected by a network. (In actuality, GUI Events are really nothing more than IPC Events, but because they play such a major role in most applications, GUI Events will be considered important enough to be in a category by themselves.)

AN EXAMPLE MULTI-INPUT APPLICATION

To demonstrate how the different types of events might be processed, a rather simple multi-input application will now be introduced. Although it is not intended to be a fully functional program, it will hopefully provide enough insight into how to apply these multi-input concepts to a real world application. The example program is designed to control an AM/FM radio connected to the RS-232 serial port of a `UNIXTM` workstation. The interface to the radio is via a Graphical User Interface which allows a user to tune the radio; the GUI is updated whenever status messages are received from the radio. The application must also be able to respond at any time to a "master control" application that might wish to take control of the radio or query it for status information. And finally, the program must periodically log statistical information about the radio to log files and be able to upload the log files to the master controller when requested. Figure 2 shows a block diagram of the example application.

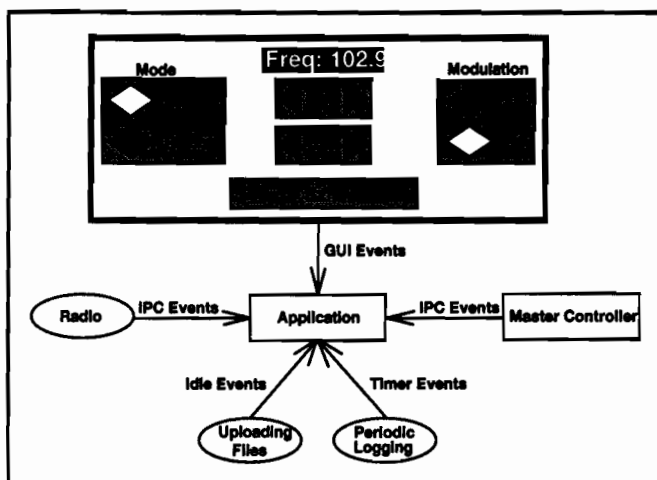


Figure 2. Block Diagram of Example Application

From this brief description, it can be determined that the program will require input from several sources: the Graphical User Interface, the radio connected to an RS-232 serial port, the master control program, and timer events from the operating system. From the previous discussion about the basic categories of input, the RS-232 and master control inputs are both considered IPC events. Since the uploading of log files could be a time consuming operation, this processing will be done during the idle time of the application. Now that the sources of input have been identified, the next step is to determine how the application should process data from the multiple sources.

PROCESSING GUI EVENTS

Since the primary purpose of the X Toolkit is to process GUI Events, it is quite easy to handle these events. The principal way this is done is with the `XtAddCallback(3X)` [2,3] procedure which declares "when the user activates this widget, invoke this callback procedure". The prototype for `XtAddCallback(3X)` and its corresponding callback procedure is as follows:

```
void XtAddCallback(
    Widget          widget,      /* widget */
    String          name,        /* callback name */
    XtCallbackProc callback,    /* callback procedure */
    XtPointer       clientData   /* client specific data */
);

typedef void (*XtCallbackProc)(
    Widget          widget,      /* widget */
    XtPointer       clientData,  /* client specific data */
    XtPointer       callData     /* callback specific data */
);
```

While it is not the purpose of this paper to show how to build the complete GUI, and it is assumed the reader is already somewhat familiar with this process, the following sample code shows how a portion of it could be done for the example application:

```

void guiTuneLeftCB(          /* tune radio callback */
    Widget      widget,      /* widget */
    XtPointer   clientData,  /* client specific data */
    XtPointer   callData    /* callback specific data */
)
{
    int         radioFD = (int)clientData;
    radioTune   tuneMsg;     /* radio tune message */

    /* Build a "tune" message and send it to the radio */
    ...
    write(radioFD, &tuneMsg, sizeof(tuneMsg));
    ...
}
void main()
{
    ...
    topLevel = XtAppInitialize(appContext,...);
    ...
    widget = XtVaCreateManagedWidget("tuneLeft",
        xmArrowButtonWidgetClass,
        parent, NULL);
    XtAddCallback(widget, XmNactivateCallback, guiTuneLeftCB,
        radioFD);
    ...
    XtAppMainLoop(appContext);
}

```

A similar callback procedure would be setup for each interactive component of the GUI. Using the X Toolkit, the application is able to very easily declare what should happen whenever the user interacts with any component of the interface. The important concepts to realize are that (1) the mechanisms provided by the X Toolkit allow the user to drive the application and (2) the callback procedures make up the meat of the application.

PROCESSING TIMER EVENTS

Timer Events have many purposes. Typically they are utilized for operations that must be done on a periodic basis. They are also very useful when an application needs to time-out a previously initiated request. In this type of situation, an application sends a request for some service to a client, enables a time-out, and expects either the client to respond with an answer or a time-out to occur. If a time-out occurs, the application can re-send the request or report an error to the user. Fortunately, processing Timer Events is also handled very nicely by the X Toolkit with the `XtAppAddTimeOut(3X)` [2, 3] procedure. This procedure allows the application to declare "after x number of milliseconds have expired, call a specified procedure. The prototype for `XtAppAddTimeOut(3X)` and its corresponding callback procedure is as follows:

```

XtIntervalId XtAppAddTimeOut(
    XtAppContext  appContext, /* application context */
    unsigned long interval,   /* in milliseconds */
    XtIntervalCallbackProc callback, /* callback procedure */
    XtPointer     clientData` /* client specific data */
);

typedef void (*XtIntervalCallbackProc)(
    XtPointer     clientData, /* client specific data */
    XtIntervalId *id         /* pointer to interval ID */
);

```

The *interval* argument specifies the time interval to delay. The *callback* argument is a pointer to a procedure to call whenever the specified *interval* expires. The *clientData* argument is whatever the application wants to pass to the *callback* procedure. The value returned by `XtAppAddTimeOut(3X)` is a unique ID generated by the X Toolkit to identify each specific timer; it is used by `XtRemoveTimeOut(3X)` whenever an application needs to disable a timer. A timer is automatically disabled whenever it expires. If an application wishes to restart an expired timer, it must call `XtAppAddTimeOut(3X)` again with the appropriate arguments. Specifying an *interval* of zero (0) results in the *callback* procedure being called as soon as the application returns control to `XtAppMainLoop(3X)`. For the example application, `XtAppAddTimeOut(0)` could be utilized as follows:

```
#define LOG_INTERVAL 60000 /* log stats every 60 seconds */

void logStatsTimerCB(
    XtPointer    clientData,
    XtIntervalId *id
)
{
    XtAppContext    appContext = (XtAppContext)clientData;
    FILE *fp;

    /* log radio statistics to log file */
    fp = fopen("logfile", "w");
    fprintf(fp, ...);
    ...
    fclose(fp);

    /* re-enable periodic log feature */
    XtAppAddTimeOut(appContext, LOG_INTERVAL, logStatsTimerCB,
                    appContext);
}

void main()
{
    ...
    topLevel = XtAppInitialize(&appContext, ...);
    ...
    XtAppAddTimeOut(appContext, LOG_INTERVAL, logStatsTimerCB,
                    appContext);
    ...
    XtAppMainLoop(appContext);
}

```

PROCESSING IDLE EVENTS

Idle Events serve many purposes. They are typically used to perform background processing while an application is idle waiting for an event. Since many applications spend most of their time waiting for input, idle events allow useful work to be performed during this otherwise "dead" time. Idle events can be used to perform quick, one-time operations such as creating pop-up interfaces that are not immediately needed by the application, as well as long, continuous operations that may take a significant amount of time to perform. For the later case, the operation would need to be partitioned into several small steps; each step would then be executed during an idle event within the application until either the operation was completed, restarted, or aborted. Fortunately, processing Idle Events is also handled very nicely by the X Toolkit with the `XtAppAddWorkProc(3X)` [2, 3] procedure. This procedure allows the application to declare "when the application is idle, call a specified procedure. The prototype for `XtAppAddWorkProc(3X)` and its corresponding callback procedure is as follows:

```

XtWorkProcId XtAppAddWorkProc(
    XtAppContext    appContext, /* application context */
    XtWorkProc      callback,   /* work procedure */
    XtPointer       clientData  /* client specific data */
);

typedef Boolean (*XtWorkProc)(
    XtPointer       clientData, /* client specific data */
);

```

The *callback* argument is a pointer to a procedure to call whenever the application is idle waiting for an event. The *clientData* argument is whatever the application wants to pass to the *callback* procedure. The value returned by `XtAppAddWorkProc(3X)` is a unique ID generated by the X Toolkit to identify each specific work procedure; it is used by `XtRemoveWorkProc(3X)` whenever an application needs to disable the work proc. If the work procedure returns `TRUE`, then the X Toolkit will remove it and it will not be called again (unless re-enabled by the application). But if it returns `FALSE`, it will be called repeatedly every time there is idle time, until the application calls `XtRemoveWorkProc(3X)`. Multiple work procedures can be registered, and they will be performed one at a time. The most recent one added has the highest priority and will be called first. Work procedures should also perform their operations quickly; typically they should execute for no more than a fraction of a second. If they are too long, the application's response time will suffer because other events cannot be processed while the work procedure is executing. For the example application, work procedure could be utilized as follows:

```

void uploadLogFileWP(
    XtPointer       clientData /* client specific data */
)
{
    LogFileInfo     *pLogInfo = (LogFileInfo *)clientData;
    char            fileName[256];

    if (pLogInfo->fileNo < pLogInfo->maxFiles) {
        sprintf(fileName, "logfile.%d", pLogInfo->fileNo);
        UploadFile(fileName); /* upload next log file */
        pLogInfo->fileNo++;   /* increment file counter */
        return(False);      /* Keep work proc enabled */
    }
    else {
        return(True);       /* all files uploaded */
    }
}

void mcInputCB(...) { /* master ctl input handler */
    ...
    case UPLOAD_FILES:
        XtAppAddWorkProc(appContext, uploadLogFileWP,
                        &logFileInfo);
    ...
}

```

PROCESSING IPC EVENTS

Processing the IPC Events is not so obvious because these events are coming from an external device and another application. Since one of the requirements of the application is that it must be able to process input from any of these sources at any time and in any order, the program cannot block while waiting for input from any one of them. One way of solving the problem

would be to use a *polling* technique in which the program would run in a continuous loop checking each source of input for data. Each source of input would be required to be setup so that any attempt to read events from the source would not cause the program to block or hang. If this were not done, then the program would hang until an event arrived and would prevent the application from processing any events from the other sources. The code to implement a polling loop might look something like the following:

```
main()
{   int          radioFD;    /* radio file descriptor */
    int          mcFD;      /* master ctl file desc */
    ...
    topLevel = XtAppInitialize(appContext, ...);
    ...
    /* Open other sources of input in "non-blocking" mode */
    radioFD = open("/dev/tty00", O_RDWR | O_NOBLOCK);
    mcFD = socket(AF_INET, SOCK_DGRAM, 0);
    fcntl(mcFD, F_SETFL, O_NOBLOCK);
    while (1) {             /* Poll for input */
        /* Check for GUI Events */
        if ((mask = XtAppPending(appContext))) {
            XtAppProcessEvent(appContext, mask);
        }
        /* Check for radio IPC Events */
        if ((len=read(radioFD, event, sizeof(event))) > 0) {
            ProcessRadioEvent(radioFD, event, len);
        }
        /* Check for master controller IPC Events */
        if ((len=recvfrom(mcFD, event, sizeof(event),
                          0, &client, &clientLen)) > 0) {
            ProcessMCEvent(mcFD, event, len);
        }
    }
}
```

While polling would work, it has many inherent disadvantages. The biggest problem is the program consumes tremendous amounts of CPU time because it spends most of its time spinning in an endless loop. It also hogs the CPU and keeps other programs from running because it is nearly always in a run state. A `sleep(2)` could be added to the bottom of the loop to keep it from hogging the CPU but then the response time of the application is affected and it also becomes more likely that an event might be lost. Polling also doesn't fit very well into the *dispatch-driven* model of the X Toolkit.

A better solution to the problem would be to use an *interrupt* technique whereby the program would enable each source of input to interrupt whenever data was available. The program would then be able to go into a wait state until the operating system woke it up whenever activity occurred on any of the sources. Fortunately, the X Toolkit provides a procedure which supports this capability very well, `XtAppAddInput(3X)` [2, 3]. This procedure allows an application to setup an event handler for each unique source of input. The prototype for `XtAppAddInput(3X)` and its corresponding callback procedure is as follows:

```
XtInputId XtAppAddInput(
    XtAppContext    appContext, /* application context */
    int             source,     /* source of input */
    XtPointer       condition,  /* input condition */
    XtInputCallbackProc callback, /* callback procedure */
    XtPointer       clientData /* client specific data */)
```



```

);

typedef void (*XtInputCallbackProc)(
    XtPointer      clientData, /* client specific data */
    int            *source,    /* pointer to input source */
    XtInputId     *id         /* pointer to input ID */
);

```

The *source* argument specifies the data source that is to be monitored. The *condition* argument tells Xt the type of activity that is to be monitored on the *source*. The accepted values are *XtInputReadMask* which indicates the program wants to be notified whenever data can be read from the *source*, *XtInputWriteMask* which means the program wants to be notified when it is OK to write to the *source*, and *XtExceptionMask* which means the program wants to be notified whenever some exceptional condition exists on the *source*. Even though the values imply the *condition* is a mask, in reality it is not. If a source is to be monitored for more than one type of condition, then *XtAppAddInput(3X)* must be called once for each condition that is to be monitored. The *callback* argument is a pointer to a procedure to call whenever the specified *condition* occurs on the *source*. The *clientData* argument is whatever the application wants to pass to the *callback* procedure. The value returned by *XtAppAddInput(3X)* is a unique ID generated by the X Toolkit to identify each specific source of input; it is used by *XtRemoveInput(3X)* whenever an application no longer wants a source to be monitored by Xt. It should also be noted that the *source* and *id* arguments of the input callback procedure are passed as pointers, which means they need to be de-referenced when used in the callback code (i.e. use **source* instead of *source*).

There are certain restrictions on the types of sources that can be monitored by the X Toolkit (at least within a UNIXTM environment). Specifically, the *source* argument must be a valid *file descriptor* which is typically obtained by the *open(2)* or *socket(2)* system calls. Most UNIXTM systems provide numerous IPC mechanisms including message queues, semaphores, shared memory, sockets, pipes, and FIFO's (or named pipes). However, message queues, semaphores, and shared memory cannot be used as the *source* argument to *XtAppAddInput(3X)* because the functions that create them (*msgget(2)*, *semget(2)*, and *shmget(2)*) do not return a *file descriptor*. The reason for these restrictions is because way down deep in the heart of Xt, the *select(2)* system call is being used to monitor the sources of input, and it only works with file descriptors. While this is not a serious limitation, it is something programmers need to be aware of when deciding what type of IPC mechanism will be used in a multi-input application.

For the example application, getting a valid source for the RS-232 will not be a problem because it will use the *open(2)* system call to obtain a *file descriptor* associated with the serial device file (i.e. *"/dev/tty00"*). However, careful consideration must be given to the type of IPC mechanism that will be used for communicating with the master controller program. From the above discussion, message queues, semaphores, and shared memory can be immediately eliminated because they don't use *file descriptors*. That leaves either named pipes or sockets. Named pipes are fairly easy to use but they can only be used for *intra-process* communication, that is, the communicating applications must be resident within the same computer. Sockets, on the other hand, can be used for both *intra-* and *inter-process* communication within a network of computers which makes them very versatile. Therefore, in order to make the application be able to communicate with the master controller regardless of where it might be located on the network, a socket will be used. More specifically, an *internet domain datagram* socket will be used because it is somewhat simpler than a *stream* socket. (As a side note, one of the idiosyncrasies of a *stream* file descriptor is that it has no record boundaries. What this implies is

that even though a transmitting application may send 100 bytes of data, the recipient may not receive the full 100 bytes at once; the data may be received one byte at a time, or in chunks at a time. Eventually all 100 bytes will be received, but they may not all get there at the same time! Users of *stream* file descriptors need to be aware of this behavior). So for the example application, `XtAppAddInput()` could be utilized as follows:

```

void radioInputCB(
    XtPointer      clientData, /* radio input handler */
    int            *source,    /* client specific data */
    XtInputId      *id,       /* ptr to source of input */
    XtInputId      *id,       /* ptr to input ID */
)
{
    radioEvent     event;     /* input event */
    int            len;       /* event length in bytes */

    len = read(*source, &event, sizeof(event));
    /* process event just received from the radio */
    ...
}

void mcInputCB(
    XtPointer      clientData, /* master ctl input handler */
    int            *source,    /* client specific data */
    XtInputId      *id,       /* ptr to source of input */
    XtInputId      *id,       /* ptr to input ID */
)
{
    mcEvent        event;     /* input event */
    int            len;       /* event length in bytes */
    struct sockaddr_in client; /* client address */
    int            clientLen; /* client addr len in bytes */

    len = recvfrom(*source, &event, sizeof(event),
                  0, &client, &clientLen);
    /* process event just received from the master controller */
    ...
}

void main()
{
    int            radioFD;    /* radio file descriptor */
    int            mcFD;      /* master ctl file desc */
    struct sockaddr_in myAddr; /* socket address */

    topLevel = XtAppInitialize(&appContext, ...);
    ...
    radioFD = open("/dev/tty00", O_RDWR);
    XtAppAddInput(appContext, radioFD, XtInputReadMask,
                  radioInputCB, NULL);
    ...
    mcFD = socket(AF_INET, SOCK_DGRAM, 0);
    myAddr.sin_family = AF_INET;
    myAddr.sin_addr.s_addr = INADDR_ANY;
    myAddr.sin_port = port;
    bind(mcFD, &myAddr, sizeof(myAddr));
    XtAppAddInput(appContext, mcFD, XtInputReadMask,
                  mcInputCB, radioFD);
    ...
    XtAppMainLoop(appContext);
}

```

This solution provides a very clean method of processing IPC Events since it is integrated into the *event-dispatching* model of the X Toolkit. The biggest advantage is that the application does not tie up the CPU with idle spin loops because it only runs when input is available. This results in the CPU being used much more efficiently, allowing other programs to run during the time the application is blocked waiting for input from one of its sources. This method is also more extensible because it is very easy to add other sources of input to the application.

IS IT SOUP YET?

The example application is now almost complete. All the sources of asynchronous input are being funneled into the central dispatching mechanism of the X Toolkit: GUI events are being dispatched to widget callbacks, IPC events are being distributed to input handlers, and Timer Events are being sent to interval callbacks. The flow of events within the application up to this point is illustrated in Figure 3.

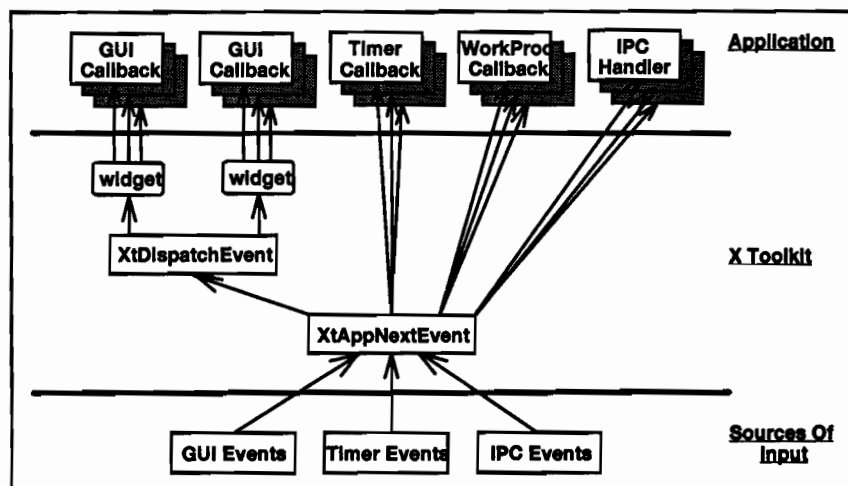


Figure 3. Flow of Events within Multi-Input Application

DISPATCHING IPC EVENTS

Although the application up to this point could be considered finished, the dispatching of IPC Events still needs some refining in order to make it more symmetric with the dispatching of GUI Events. While the X Toolkit provides some very nice hooks that allow an application to register procedures to be called whenever input is available from alternate sources, it is still up to the application to read the event and do something with it. Typically an application of this sort reads the event, determines the type of event just received from a unique field within the event, and then distributes the event to another procedure for further processing. So the issue that remains to be resolved is how to distribute IPC Events after they have been read by the appropriate IPC Handler. One way to resolve the problem would be to have a *switch* statement inside each IPC Handler that would look at the type of each IPC Event and perform some action based on the information in the event. This method is illustrated in the following code fragment:

```

void radioInputCB(
    XtPointer      clientData, /* radio input handler */
    int            *source,    /* client specific data */
    XtInputId     *id         /* ptr to source of input */
)
{
    XtInputId     *id         /* ptr to input ID */
    radioEvent    event;     /* input event */
    int            len;       /* event length in bytes */

    len = read(*source, &event, sizeof(event));
    switch(event.type) {
        /* determine type of event */
        case radioFREQUENCY_EVENT: /* update freq label in GUI */
            ...
            break;
        case radioMODULATION_EVENT: /* update GUI mod toggle */
            ...
            break;
        ...
    }
}

void mcInputCB(
    XtPointer      clientData, /* master ctl input handler */
    int            *source,    /* client specific data */
    XtInputId     *id         /* ptr to source of input */
)
{
    XtInputId     *id         /* ptr to input ID */
    mcEvent       event;     /* input event */
    int            len;       /* event length in bytes */
    struct sockaddr_in client; /* client address */
    int            clientLen; /* client addr len in bytes */

    len = recvfrom(*source, &event, sizeof(event),
                   0, &client, &clientLen);
    switch(event.type) {
        /* determine type event */
        case mcREMOTE_MODE_EVENT: /* set radio in remote mode */
            ...
            break;
        case mcLOCAL_MODE_EVENT: /* set radio in local mode */
            ...
            break;
        case mcSTATUS_EVENT:     /* get radio status */
            ...
            break;
        ...
    }
}

```

The disadvantages of this solution are that the switch statement can become quite long and very complex, and it is not very dynamic. A better solution would be to implement a scheme that is modeled after the dispatch-driven architecture of the X Toolkit, which consists of two parts: event registration and event dispatching. Event registration would allow the application to receive IPC Events by registering callback procedures for each source of input. Event dispatching would then be responsible for invoking the appropriate callback procedures associated with a source whenever an IPC Event was received. Although the X Toolkit does not provide any built-in procedures for dispatching events from alternate sources (other than XtAppAddInput(3X) and XtAppAddTimeOut(3X)), it is fairly simple to develop a set of routines that does provide this functionality. The two primary procedures needed are one that allows an application to register an IPC Event with a callback procedure, and another for dispatching an IPC Event to the appropriate callback procedures. In order to make it easy to distinguish the new routines from the normal Xt procedures, an Ext prefix will be used to

designate them as *Extended Xt procedures*. The event registration procedure will be called `ExtAddIPCCallback()`; the prototype for it and its corresponding callback procedure is as follows (sample code for all the *Ext* procedures is included at the end of this paper):

```
void ExtAddIPCCallback(
    int          source,      /* source file descriptor */
    unsigned long type,      /* event type */
    ExtIPCCallbackProc callback, /* callback procedure */
    XtPointer    clientData /* client specific data */
);

typedef void (*ExtIPCCallbackProc)(
    int          source,      /* source file descriptor */
    XtPointer    clientData, /* client specific data */
    XtPointer    callData    /* callback specific data */
);
```

This procedure is very similar to the `XtAddCallback(3X)` procedure. The *source* argument is the file descriptor associated with the source of input. The *type* argument is the type of IPC event the application is interested in receiving and depends on the format of the data in the event. Most events contain a key field which indicates the *type* of the event. Even though the *type* argument is an unsigned long, the actual field in the event can be of type *char*, *short*, *int*, or *long*; the rule is that if it can be cast to an unsigned long, then it is ok (which means it cannot be of type *float*). The *callback* argument is a pointer to the procedure to call whenever an IPC Event of type *type* is received on the *source*. The *clientData* argument is whatever the application wants to pass to the *callback* procedure. For the example application, `ExtAddIPCCallback()` could be utilized as follows:

```
void radioFrequencyCB(          /* radio frequency callback */
    int          source,      /* source file descriptor */
    XtPointer    clientData, /* client specific data */
    XtPointer    callData    /* callback specific data */
)
{
    Widget          widget = (Widget)clientData;
    ExtAnyIPCCallbackStruct *pCbs =
        (ExtAnyIPCCallbackStruct *)callData;
    radioEvent      *pEvent = (radioEvent *)pCbs->event;

    sprintf(string, "%f", pEvent->frequency);
    XtVaSetValues(widget,
                  XmNlabel, XmStringCreateSimple(string),
                  NULL);
}
...
void mcRemoteModeCB(          /* remote mode callback */
    int          source,      /* source file descriptor */
    XtPointer    clientData, /* client specific data */
    XtPointer    callData) /* callback specific data */
{
    int          radioFD = (int)clientData;
    ExtSocketIPCCallbackStruct *pCbs =
        (ExtSocketIPCCallbackStruct *)callData;
    mcEvent      *pEvent = (radioEvent *)pCbs->event;
    write(radioFD, pEvent, pCbs->length);
}
...
```

```

void main()
{
    int          radioFD;    /* radio file descriptor */
    int          mcFD;      /* master ctlfile desc */
    struct sockaddr_in myAddr; /* socket address */
    ...
    topLevel = XtAppInitialize(&appContext, ...);
    ...
    /* Initialize IPC Events */
    radioFD = open("/dev/tty00", O_RDWR);
    XtAppAddInput(appContext, radioFD, XtInputReadMask,
                  radioInputCB, NULL);
    ExtAddIPCSource(radioFD);
    ExtAddIPCCallback(radioFD, radioFREQUENCY_EVENT,
                     radioFrequencyCB,
                     freqWidget);
    ExtAddIPCCallback(radioFD, radioMODULATION_EVENT,
                     radioModulationCB,
                     modWidget);
    ...
    mcFD = socket(AF_INET, SOCK_DGRAM, 0);
    myAddr.sin_family = AF_INET;
    myAddr.sin_addr.s_addr = INADDR_ANY;
    myAddr.sin_port = port;
    bind(mcFD, &myAddr, sizeof(myAddr));
    XtAppAddInput(appContext, mcFD, XtInputReadMask, mcInputCB,
                  NULL);
    ExtAddIPCSource(mcFD);
    ExtAddIPCCallback(mcFD, mcREMOTE_MODE_EVENT, mcRemoteModeCB,
                     radioFD);
    ExtAddIPCCallback(mcFD, mcLOCAL_MODE_EVENT, mcLocalModeCB,
                     radioFD);
    ExtAddIPCCallback(mcFD, mcSTATUS_EVENT, mcStatusCB,
                     radioFD);
    ...
    /* Process inputs and dispatch events */
    XtAppMainLoop(appContext);
}

```

Now that the IPC Events are registered, a procedure called **ExtDispatchIPCEvent()** will dispatch the events to the appropriate callback procedures. This procedure will first determine the source of the event and then look on a list associated with the source for callbacks that are interested in the event. Each callback that is interested in the event will be called and passed as arguments the source, any data specified by the application when the callback was registered, and callback specific information about the IPC event. The prototype for this procedure and the callback specific data is as follows:

```

void ExtDispatchIPCEvent(
    int          source,    /* source file descriptor */
    ExtAnyIPCCallbackStruct *pCbs /* callback structure */
);

typedef struct {
    unsigned long    type;    /* Generic IPC callback data*/
    void            *event;  /* event type */
    int              length; /* pointer to the event */
} ExtAnyIPCCallbackStruct;

```

Just like the *callData* structure passed to a widget callback procedure is widget dependent, so is the *callData* structure that is passed to an IPC callback procedure. The **ExtAnyIPCCallbackStruct** structure shown above is the minimum amount of information that is passed to an IPC callback function as the *callData* argument. The exact format of this data will depend on the input handler but should contain at least the *type*, *event*, and *length* fields (see the attached listings for an example of what a callback structure would look like for a socket input handler). For the example application, **ExtDispatchIPCEvent()** could be used as follows:

```
void radioInputCB(                /* radio input handler */
    XtPointer      clientData, /* client specific data */
    int            *source,    /* ptr to source of input */
    XtInputId     *id         /* ptr to input ID */
)
{
    radioEvent     event;      /* input event */
    ExtAnyIPCCallbackStruct cbs; /* IPC Callback structure */

    cbs.length = read(*source, &event, sizeof(event));
    /* dispatch event just received from the radio */
    cbs.type = event.type;      /* type of event received */
    cbs.event = &event;        /* pointer to input event */
    ExtDispatchIPCEvent(*source, &cbs);
}

void mcInputCB(                  /* master ctl input handler */
    XtPointer      clientData, /* client specific data */
    int            *source,    /* ptr to source of input */
    XtInputId     *id         /* ptr to input ID */
)
{
    mcEvent        event;      /* input event */
    struct sockaddr_in client;  /* client address */
    int            clientLen;  /* client addr len in bytes */
    ExtSocketIPCCallbackStruct cbs; /* IPC Callback struct */

    cbs.length = recvfrom(*source, &event, sizeof(event),
                          0, &client, &clientLen);
    /* dispatch event just received from the master ctl */
    cbs.type = event.type;      /* type of event received */
    cbs.event = &event;        /* pointer to input event */
    cbs.client = &client;      /* clients address */
    cbs.clientLength = clientLen; /* client addr len in bytes */
    ExtDispatchIPCEvent(*source,
                        (ExtAnyIPCCallbackStruct *) &cbs);
}

```

The advantages of this approach are many. The most obvious is that it fits very well into the existing Xt dispatch-driven programming model, which allows programs to be written in a much more declarative style (when this event occurs, do this). It also greatly simplifies the logic of the applications, and results in a very modular design because small procedures are written to deal with specific events that occur within each component. The method is also much more dynamic and easily extended if and when it becomes necessary to add new event types to the application.

The example application is now complete. All the sources of asynchronous input are now being processed and dispatched to the appropriate callback procedures. The advantages of this solution are that it is very dynamic and can be very easily extended whenever it becomes necessary to process other sources of input. It fits very nicely into the easily understood X Toolkit *dispatch-*

driven model which allows the application to declare "when this event happens, do this action". Figure 4 shows the architecture of the solution.

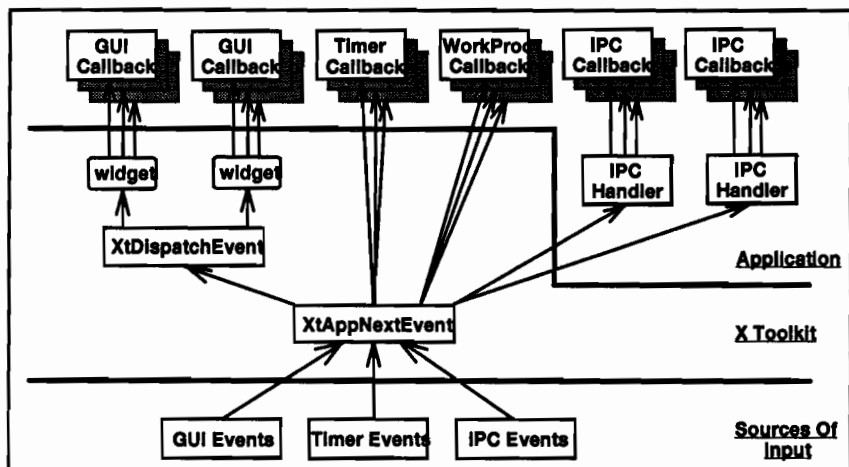


Figure 4. Multi-Input Event Dispatching Architecture

CONCLUSIONS

For several years now, the X Toolkit has proven itself as a very good framework for building Graphical User Interface applications. And now with very little effort, it has been shown how easily it can be used as a foundation for building multi-input applications. The *dispatch-driven* framework of the X Toolkit Intrinsics maps very well to the problem domain of a multi-input application:

- GUI Events \Rightarrow XtAddCallback(3X)
- Timer Events \Rightarrow XtAppAddTimeOut(3X)
- Idle Events \Rightarrow XtAppAddWorkProc(3X)
- IPC Events \Rightarrow XtAppAddInput(3X)

The concepts presented here allow for a more object-oriented design because all the details of processing each unique source of input can be encapsulated inside input handlers. Although not formally discussed in this paper, one could also take the concepts a step further to create *IPC Widgets* for each unique source of input. Using these techniques, events from multiple sources of input are seamlessly integrated into the dispatch-driven framework of the X Toolkit. Hopefully this paper has shed some light on how to solve a problem that may have previously been a "thorn in the flesh"

ACKNOWLEDGMENTS

Thanks to Randy Davis, Bill Frawley, Chuck Poche, and Lee Spencer for their editorial comments. Special thanks to Shelly, Ryan, and Brooke for their patience!

REFERENCES

- [1] Douglas A. Young. *The X Window System: Programming and Applications with Xt, OSF/Motif Edition*. Prentice Hall, 1990.
- [2] Adrian Nye and Tim O'Reilly. *X Toolkit Intrinsic Programming Manual, OSF/Motif Edition*. O'Reilly & Associates, Inc., 1990.
- [3] Staff of O'Reilly & Associates, Inc. *X Toolkit Intrinsic Reference Manual, Second Edition for X11 Release 4*. O'Reilly & Associates, Inc., 1991.

UNIX is a trademark of UNIX System Laboratories.

SAMPLE IMPLEMENTATION OF EXT PROCEDURES

The following listings contain C code for a sample implementation of the Ext (Extended Xt) procedures referenced in the paper. It consists of a header file, `Ip.c.h`, and a source file, `Ip.c`.

```
/*
 * Ip.c.h
 */
#ifdef IPC_H
#define IPC_H

#include <X11/Intrinsic.h> /* X Toolkit Intrinsic stuff */

/*
 * IPC typedefs
 */
typedef struct {
    unsigned long type; /* Generic IPC callback data */
    void *event; /* event type */
    int length; /* pointer to the event */
} ExtAnyIPC_CALLBACK_STRUCT; /* event length (in bytes) */

typedef struct {
    unsigned long type; /* Socket IPC callback data */
    void *event; /* event type */
    int length; /* pointer to the event */
    struct sockaddr_in *client; /* client who sent the event */
    int clientLength; /* client length (in bytes) */
} ExtSocketIPC_CALLBACK_STRUCT;

typedef void (*ExtIPC_CALLBACK_PROC)(
    int source, /* source file descriptor */
    XtPointer clientData, /* client specific data */
    XtPointer callData /* callback specific data */
);
```

```

typedef struct {
    unsigned long type; /* IPC Callback record */
    ExtIPCCallbackProc callback; /* event type */
    XtPointer clientData; /* callback procedure */
} ExtIPCCallbackRec, *ExtIPCCallbackList;

typedef struct {
    ExtIPCCallbackList callbacks; /* array of callbacks */
    int numCallbacks; /* num. callbacks in use */
    int numSlots; /* num. callbacks available */
} ExtIPCSourceInfo;

/*
*****
*
* IPC Function prototypes
*
*****
*/
extern void ExtInitIPCSource(
    void
);

extern ExtIPCSourceInfo *ExtAddIPCSource(
    int source /* source file desc */
);

extern void ExtRemoveIPCSource(
    int source /* source file desc */
);

extern ExtIPCSourceInfo *ExtGetIPCSource(
    int source /* source file desc */
);

extern ExtIPCSourceInfo *ExtDupIPCSource(
    int source, /* source to duplicate */
    int newSource /* new source */
);

extern void ExtAddIPCCallback(
    int source, /* source file desc */
    unsigned long type, /* event type */
    ExtIPCCallbackProc callback, /* callback procedure */
    XtPointer clientData /* client specific data */
);

extern void ExtRemoveIPCCallback(
    int source, /* source file desc */
    unsigned long type, /* event type */
    ExtIPCCallbackProc callback, /* callback procedure */
    XtPointer clientData /* client specific data */
);

extern void ExtDispatchIPCEvent(
    int source, /* source file desc */
    ExtAnyIPCCallbackStruct *pCbs /* callback structure */
);

#endif /* IPC_H */

```

```

/*****/
/* Ipc.c */
/*****/
#include <stdio.h>
#include <stdlib.h>
#include "Ipc.h"

#ifdef MAXFUPLIM
#define FD_SETSIZE MAXFUPLIM /* max file descs */
#endif

#define VALID_FD(source) \
( (0 <= (source)) && ((source) < FD_SETSIZE) )

#define VALID_SOURCE(source) \
( VALID_FD(source) && (L_sourceTable[(source)] != NULL) )

#define IPC_TABLE_SIZE \
( sizeof(L_sourceTable) / sizeof(L_sourceTable[0]) )

/* IPC table indexed by source */
static ExtIPCSourceInfo *L_sourceTable[FD_SETSIZE];

/*
*****
* DESCRIPTION
* This function initializes the IPC package.
*
*****
*/
void ExtInitIPCSourceInfo(void)
{
static int init = 0; /* IPC table init flag */
int i; /* index/loop counter */

if (!init) { /* if first time */
for (i = 0; i < IPC_TABLE_SIZE; i++) {
L_sourceTable[i] = NULL; /* ..init all entries */
}
init = 1; /* ..only do this once */
}
}

/*
*****
* DESCRIPTION
* This function creates an IPC source info structure for the
* specified source.
*
*****
*/
ExtIPCSourceInfo *ExtAddIPCSource(
int source /* source file desc */
)
{
ExtIPCSourceInfo *pSI = NULL; /* ptr to source info */

ExtInitIPCSourceInfo(); /* init IPC package */
}

```

```

if ((VALID_FD(source)) && /* if source is valid, */
    (L_sourceTable[source] == NULL)) {
    pSI = L_sourceTable[source] =
        (ExtIPCSourceInfo *)calloc(1, sizeof(ExtIPCSourceInfo));
}
return(pSI);
}

/*
*****
*
* DESCRIPTION
* This function destroys an IPC source info structure for the
* specified source. If no other source is linked to the
* source (i.e. it has not been "dup'ed"), then the source info
* structure will be freed.
*
*****
*/
void ExtRemoveIPCSource(
    int source /* source file desc */
)
{
    static char *funcName = "ExtRemoveIPCSource";

    ExtIPCSourceInfo *pSI; /* ptr to source info */
    int i; /* index/loop counter */
    int inUseCount; /* in use counter */

    if ((pSI = ExtGetIPCSource(source)) == NULL) {
        fprintf(stderr, "%s: info not found for source %d\n",
            funcName, source);
    }
    else {

        /* Check to see if anybody else is using this source */
        for (i = inUseCount = 0; i < IPC_TABLE_SIZE; i++) {
            if ((i != source) &&
                (L_sourceTable[i] == L_sourceTable[source])) {
                inUseCount++;
            }
        }

        if (inUseCount == 0) { /* if source not in use */
            free((char *)pSI->callbacks); /* ..free callback array */
            free((char *)pSI); /* ..free source info */
        }

        L_sourceTable[source] = NULL; /* mark the source free */
    }
}

/*
*****
*
* DESCRIPTION
* This function returns a pointer to the source information
* associated with the specified source.
*
*****

```

```

*****
*/
ExtIPCSourceInfo *ExtGetIPCSource(
    int          source          /* source file desc */
)
{
    return(VALID_SOURCE(source) ? L_sourceTable[source] : NULL);
}

/*
*****
* DESCRIPTION
*   This function "dup's" the information for a specified source
*   and links it to a new source.  The result is that both the
*   original source and the new source point to the same source
*   information.
*
*   This function is used primary by sources that are associated
*   with "connection-oriented" sockets, which use two sources:
*   one for the "listen" socket and another for each "connected"
*   socket.
*
*****
*/
ExtIPCSourceInfo *ExtDupIPCSource(
    int          source,          /* source to duplicate */
    int          newSource       /* new source */
)
{
    ExtIPCSourceInfo *pSI = NULL; /* ptr to new source */

    if ((VALID_SOURCE(source)) &&          /* if sources are valid *//
        (VALID_FD(newSource)) &&         /* ..then "dup" the info *//
        (L_sourceTable[newSource] == NULL)) {
        pSI = L_sourceTable[newSource] = L_sourceTable[source];
    }
    return(pSI);
}

/*
*****
* DESCRIPTION
*   This function adds an IPC callback to the list of callbacks
*   for the specified source.
*
*****
*/
#define SLOTS          16          /* slots per chunk */

void ExtAddIPCCallback(
    int          source,          /* source file desc */
    unsigned long type,          /* IPC event type *//
    ExtIPCCallbackProc callback, /* callback procedure *//
    void         *clientData     /* client specific data *//
)
{
    static char *funcName = "ExtAddIPCCallback";
    ExtIPCSourceInfo *pSI;      /* ptr to source info */

```

```

        int          start = 0;          /* where to start search */
        int          numSlots;          /* num. slots to alloc */
        int          bytes;             /* num. bytes to alloc */
        int          i;                 /* counter/index */

/*
*****
* If the callback array for this file descriptor is full, then
* reallocate another chunk of memory.
*****
*/
if ((pSI = ExtGetIPCSource(source)) == NULL) {
    fprintf(stderr, "%s: info not found for source %d\n",
            funcName, source);
    goto error;
}
if ((pSI->callbacks == NULL) ||
    (pSI->numCallbacks == pSI->numSlots)) {
    numSlots = ((pSI->callbacks == NULL) ?
                SLOTS : pSI->numSlots + SLOTS);
    bytes = sizeof(ExtIPCCallbackRec) * numSlots;

    if ((pSI->callbacks = (ExtIPCCallbackList)
        realloc(pSI->callbacks, bytes)) == NULL){
        fprintf(stderr,
            "%s: cannot allocate callback list for source %d\n",
            funcName, source);
        goto error;
    }
    start = numSlots - SLOTS;          /* init new memory chunk */
    bytes = sizeof(ExtIPCCallbackRec) * SLOTS;
    memset((char *)&pSI->callbacks[start], 0, bytes);
    pSI->numSlots = numSlots;
}
/*
*****
* Find the first empty slot in the callback array, save the
* caller's information, and increment the "in use" counter.
*****
*/
for (i = start; i < pSI->numSlots; i++) {
    if (pSI->callbacks[i].callback == NULL) {
        pSI->callbacks[i].type = type;
        pSI->callbacks[i].callback = callback;
        pSI->callbacks[i].clientData = clientData;
        pSI->numCallbacks++;
        break;
    }
}

error:
    return;
}

/*
*****
*
* DESCRIPTION
* This function removes an IPC callback from the list of
* callbacks for the specified source.
*****

```

```

*
*****
*/
void ExtRemoveIPCCallback(
    int          source,          /* source file desc */
    unsigned long type,          /* IPC event type */
    ExtIPCCallbackProc callback, /* callback procedure */
    void         *clientData     /* client specific data */
)
{
    static char      *funcName = "ExtRemoveIPCCallback";

    ExtIPCSourceInfo *pSI;      /* ptr to source info */
    int              i;         /* index/loop counter */

    if ((pSI = ExtGetIPCSource(source)) == NULL) {
        fprintf(stderr, "%s: info not found for source %d\n",
            funcName, source);
    }
    else {

        /* Remove all occurrences of callback from callback list */
        for (i = 0; i < pSI->numSlots; i++) {
            if ((pSI->callbacks[i].type == type) &&
                (pSI->callbacks[i].callback == callback) &&
                (pSI->callbacks[i].clientData == clientData)) {
                pSI->callbacks[i].callback == NULL;
                pSI->numCallbacks--;          /* decr "in use" counter */
            }
        }
    }
}

/*
*****
*
* DESCRIPTION
* This function dispatches an IPC event to the appropriate
* callback by searching the callback list associated with the
* specified source.
*
*****
*/
void ExtDispatchIPCEvent(
    int          source,          /* source file desc */
    ExtAnyIPCCallbackStruct *pCbs /* callback structure */
)
{
    static char      *funcName = "ExtDispatchIPCEvent";

    ExtIPCSourceInfo *pSI;      /* source info */
    int              count;     /* matches found counter */
    int              i;         /* index/loop counter */

    /*
    * Search the callback list associated with the source and invoke
    * the callbacks that are interested in the event.
    */
}

```

```

if ((pSI = ExtGetIPCSource(source)) == NULL) {
    fprintf(stderr, "%s: info not found for source %d\n",
           funcName, source);
}
else {
    for (i = count = 0;
         i < pSI->numSlots && count < pSI->numCallbacks;
         i++) {
        if ((pSI->callbacks[i].type == pCbs->type) &&
            (pSI->callbacks[i].callback != NULL)) {
            (*pSI->callbacks[i].callback)(source,
                                         pSI->callbacks[i].clientData,
                                         pCbs);
            count++;
        }
    }
}

/* If no callbacks were found, report unregistered event. */
if (count == 0) {
    fprintf(stderr,
           "%s: unregistered IPC event received: "
           "type=%ld, length=%ld\n",
           funcName, pCbs->type, pCbs->length);
}
}
}

```


Paper Number: 4010

Using Windows NT as a Desktop in an Internetworked, Client/Server Environment

Robert C. Jones
Hewlett-Packard
2015 South Park Place
Atlanta, Georgia 30339
(404) 850-2887

Hewlett-Packard Corporate IT, in conjunction with the HP United States Field Operations, began an evaluation of Windows NT as a potential PC client for HP's internal PC environment in 1993. This paper will examine some of the experiences of HP IT in using Windows NT as a client within Hewlett-Packard's internetworked, geographically dispersed TCP/IP environment. Three areas will be examined:

- Windows NT as a simultaneous client and server
- Windows Sockets applications in Windows NT
- Remote management of Windows NT clients

Certainly, one of the strongest features of Windows NT is its built-in networking. Windows NT includes TCP/IP, Lan Manager, Windows Sockets, and a DCE-compatible RPC mechanism within the basic product (as well as other protocols and network APIs). Windows NT seems to have been designed from the ground up with client/server computing in mind.

Windows NT as a Simultaneous Client and Server

The ability of Windows NT to act as a client and a server at the same time is certainly a fairly revolutionary concept in the PC world (although hardly earth-shattering in the Unix workstation world). It should be noted that all versions of Windows NT have client and server capability.

File Sharing

In HP's current Windows 3.1/Lan Manager environment, most file sharing between PCs is conducted through intermediary LM/X file servers. NT can, essentially, remove reliance on the middleman for file sharing. Data can easily be shared among Windows NT users. Creating a network share in Windows NT is a

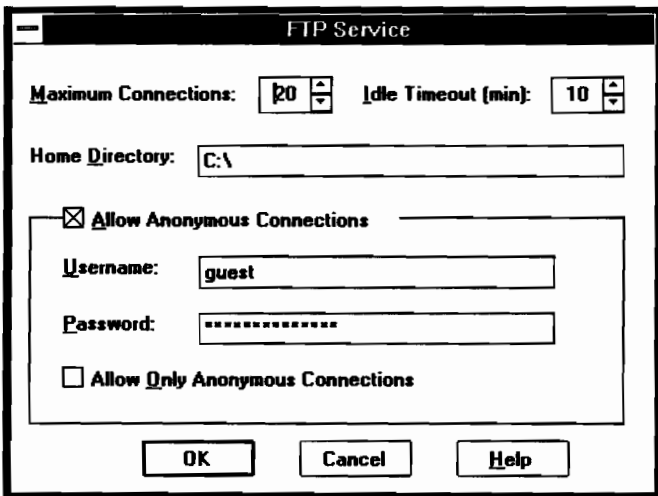
simple point and click operation. (For an example of the "Shared Directory" screen, see the bitmap under "CD-ROM Server"). Once the share has been created, other NT or Lan Manager clients on the network can connect to the NT system through the familiar NET USE convention.

NET USE X: \\NTDESKTOP\HP

Once the user has issued the NET USE command, simply changing to the X: drive will give them access to all of the files and directories associated with share "HP" on NT machine NTDESKTOP.

FTP Server

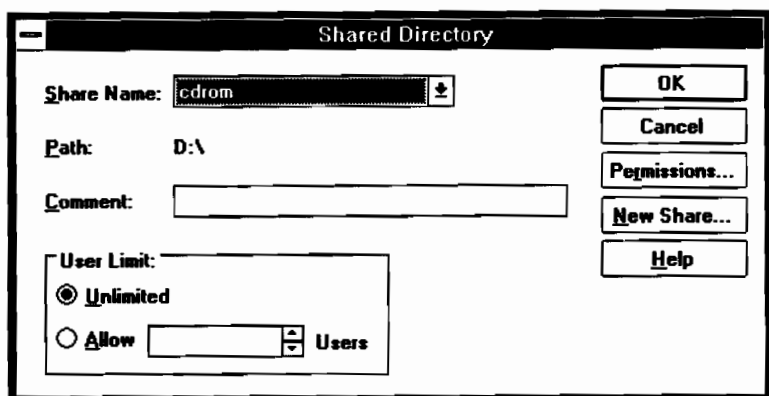
Windows NT comes with a FTP client and server built in, so file sharing between NT and Unix is fairly easy. During the evaluation, I exchanged data between NT and HP3000s, and NT and HP9000s, via FTP. Configuration of the FTP server within Windows NT is quite simple, as can be seen by the bitmap below.



CD-ROM Server

One very nice feature of the Windows NT server capability is the ability to hang a CD-ROM drive off of a Windows NT system, and share it on the network.

During the evaluation, I made CD-ROM based Windows NT patches available to other evaluators within HP by simply loading the CD-ROM on my personal CD-ROM drive, and sharing the drive on the network through Windows NT. This proved to be an efficient and easy way to provide updates to fellow-NT evaluators.

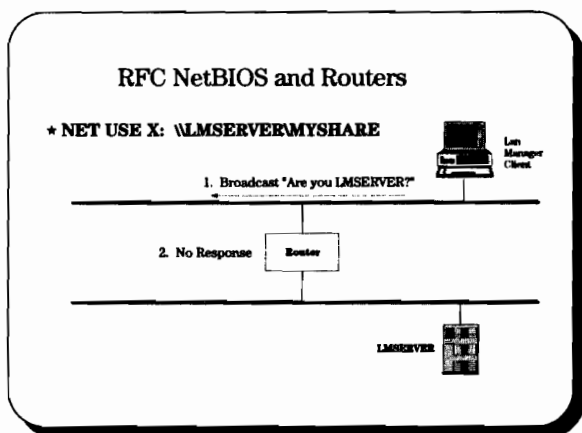


Once I had created the share (as shown above), other Lan Manager or NT users on the network could access the CD-ROM drive the same way they would access any other share:

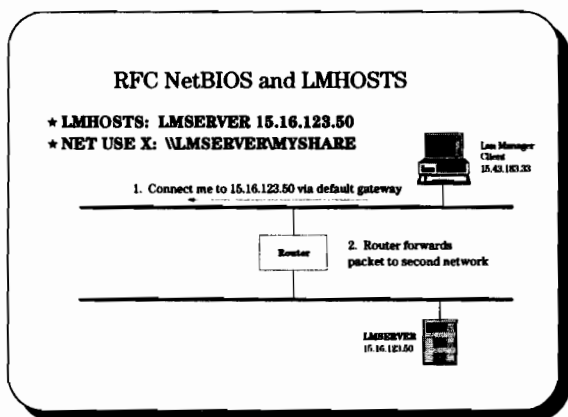
```
NET USE R: \\NTDESKTOP\CDROM
```

Internetworked File Sharing

Of course, sharing files on a single subnet is fairly uncomplicated. Lan Manager works on a single subnet by broadcasting the name of the server that it wants to connect to, and receives the IP address from the target system in return. However, since routers are generally configured not to forward such broadcasts, this mechanism does not work in an internetworked environment. In the example below, the user is trying to connect to a server named LMSERVER. The NET USE fails, because LMSERVER is on the other side of a router.



To solve this problem, the original Lan Manager provided a host name to IP address mapping mechanism known as RFC for NetBIOS over TCP/IP. This mechanism required the user to provide the IP address and name of the server they were trying to connect to in a file named LMHOSTS. Once that mapping had been made, the NET USE command could execute.



While this methodology worked, it was clumsy and hard to manage. Users were required to provide name to IP address mappings personally. There was no mechanism to update local LMHOSTS files (what if an IP address for a server changed?) This mechanism was particularly distressing in HP's internal environment, as DNS is the preferred name to IP address mechanism. (The

original Lan Manager did not support DNS for NET USE lookups). Windows NT, however, has gone a long way towards improving this situation. Several enhancements have been made, including:

- In the original Lan Manager, changes in the LMHOSTS file required a reboot to take effect. In Windows NT, the LMHOSTS file is read dynamically.
- Windows NT allows LMHOSTS files located on other machines in the network to be accessed from a NT desktop, during name - IP address resolution.
- Windows NT 3.5 will have additional name-to-IP address resolution enhancements.

NOTE: Windows NT fully supports use of DNS with Internet Protocol Suite applications such as FTP and TELNET.

Using Windows Sockets Applications with Windows NT

The rapid acceptance of Windows Sockets 1.1 as the emerging standard for Sockets programming on the PC has been a great boon to application developers trying to write client/server applications for Microsoft operating systems. Prior to this standard specification, application developers were forced to provide slightly different versions of their product for the several different sockets implementations that existed in the PC world.

Windows Sockets Defined

"The Windows Sockets specification defines a network programming interface for Microsoft Windows which is based on the "socket" paradigm popularized in the Berkeley Software Distribution (BSD) from the University of California at Berkeley. It encompasses both familiar Berkeley socket style routines and a set of Windows-specific extensions designed to allow the programmer to take advantage of the message-driven nature of Windows.

The Windows Sockets Specification is intended to provide a single API to which application developers can program and multiple network software vendors can conform. Furthermore, in the context of a particular version of Microsoft Windows, it defines a binary interface (ABI) such that an application written to the Windows Sockets API can work with a conformant protocol implementation from any network software vendor. This specification thus defines the library calls and associated semantics to which an application developer can program and which a network software vendor can implement." ("A Guide to Windows Sockets", 1 June 1993, Martin Hall, JSB Corporation)

Windows NT has fully embraced the Windows Sockets paradigm, providing Windows Sockets over both TCP/IP and Novell IPX/SPX. I successfully tested a number of Windows Sockets apps within Windows NT, including electronic mail packages, FTP front-ends, terminal emulators, and Internet browsers.

(See Appendix A for a discussion of using the Microsoft RPC mechanism within Hewlett-Packard's environment.)

Remote Management of Windows NT Clients

One feature of special interest to HP IT regarding Windows NT is its remote management capabilities. These capabilities were evaluated by remotely accessing NT clients at 15 different HP sites in 4 countries.

Remote management functions evaluated included:

- Remote resource monitoring, including CPU, disk space, and memory. This includes the abilities to set up "alerts" that are mailed automatically to a central NT system, when configured thresholds are breached. (Example: Send an alert if system RCJONES3125 exceeds 85% disk space usage.)
- Remote log file monitoring
- Remote configuration changes
- Remote addition, deletion, and modification of users
- Remote addition, deletion, and modification of shares
- Remote execution of programs at a certain date and time
- Remotely starting, stopping and querying the status of services
- Remotely viewing the status of, and killing processes
- Remote backup of NT configurations
- Remote monitoring of access to system resources (via FTP, NET USE, sockets etc.)
- Remote hardware/system diagnostic ability
- Remote troubleshooting

As an overview, the following tables link up capabilities with tools:

GUI-based

Capability	Tool	Availability
Resource Monitoring	Performance Monitor	NT, NTAS
Log File Monitoring	Event Viewer	NT, NTAS
Printer Management	Print Manager	NT, NTAS
Registration Data Base Editing	REGEDT32.EXE	NT, NTAS
Timed program/command scheduling	WINAT.EXE	Resource Kit
Server Management	SRVMGR.EXE	NTAS, Resource Kit
User Management	USRMGR.EXE	NTAS, Resource Kit
Process Monitoring	PVIEWER.EXE	Resource Kit

Capability	Tool	Availability
Monitoring access to shared directories	NETWATCH.EXE	Resource Kit
Hardware/system diagnostic tool	WINMSD.EXE	NT, NTAS

Character-based

Capability	Tool	Availability
Timed program/command scheduling	AT.EXE	NT, NTAS
Log File Monitoring	DUMPEL.EXE	Resource Kit
Service Controller	NETSVC.EXE	Resource Kit
Registry Backup	REGBACK.EXE	Resource Kit
Registration Data Base Editing	REGINI.EXE	Resource Kit

Windows NT Remote Management Paradigm

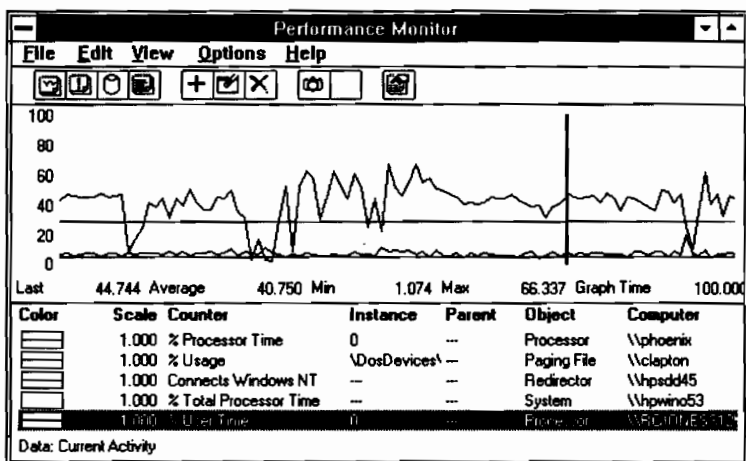
It is important to note that the Windows NT remote management paradigm is very much based on a client/server model. Most of the tools described here require administration capability (and the associated passwords) to either run, or make modifications.

Remote Resource Monitoring

Primary Tool: Performance Monitor. The Performance Monitor is a core part of Windows NT. It allows the monitoring of resource usage (such as memory, disk, network connections etc.) on local and remote Windows NT systems.

Evaluation

Certainly the Performance Monitor which ships as part of Windows NT and NTAS is one of the most comprehensive and easy-to-use resource monitoring tools I have ever seen. The tool allows both graphical and report modes, and allows monitoring of resources on multiple machines at the same time. The bitmap below shows resources being monitored on 5 different machines in four different geographic areas:



The Performance Monitor also contains an "Alert" capability. The following Alert was generated in response to the alert condition stating, "Let me know if the free disk space on machine \\SAMICHA3125 goes under 20%". The Alert information can appear in a window or as a pop-up message (shown below).



Remote Log File Monitoring

Primary Tool: Event Viewer. The Event Viewer is a GUI-based tool that allows access to Windows NT log files (system, security, and application). It can be used locally or remotely.

Evaluation

The Event Log is easy to use in a remote environment. Once the initial connection is made, performance in reading an event log remotely is comparable to reading it locally.

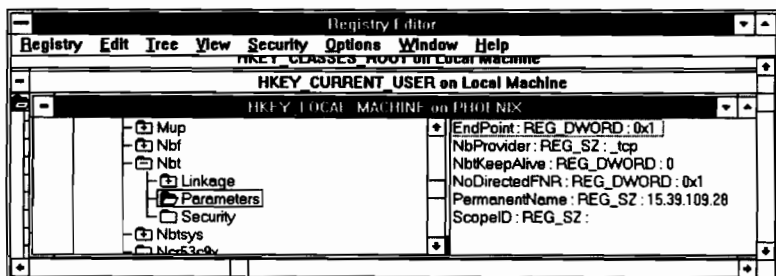
There is also a character-based log file reader. The command sequence below says "Write to the file MYFILE, everything in the SYSTEM log on the machine PHOENIX which has to do with the Lan Manager REDIRECTOR".

```
dumpel -s phoenix -f myfile -l system -m Rdr
```

12/1/93	6:12:00 PM	2	0	3012	N/A	Rdr
PHOENIX	\Device\LanmanRedirector	hpbs4164				
1/25/94	1:41:03 AM	2	0	3013	N/A	Rdr
PHOENIX	\Device\LanmanRedirector	hpbs4164				
2/5/94	11:06:11 AM	2	0	3012	N/A	Rdr

Remote Configuration Changes

Primary Tool: REGEDT32.EXE. All configuration information in Windows NT is stored in one central location, named the Registration Database. The Registry Editor allows direct changes to be made to the Windows NT Configuration Registry. It can be used on local or remote systems. The Registry Editor is easy to use, and allows access to the registries on multiple machines simultaneously. The bitmap below shows the Registry Editor accessing the local system, as well as one remote system.



There is also a command line method of making remote registry entries. The command sequence below says "At the specified time, execute the REGINI command on machine \\SAMICH3125, using as input the REG.INI file located on machine \\RCJONES3125, and redirect the output back to RCJONES3125". The

command listed below adds a logon message window to the NT system \\SAMICH3125, which says (cleverly) "Message from Afar".

```
C:\>at \\samicha3125 16:13 "regini \\rcjones3125\rcj\hp\reg.ini > \\rcjones3125\rcj\hp\sam1613"
```

REG.INI file:

```
\registry\machine\SOFTWARE\Microsoft\Windows  
NT\CurrentVersion\Winlogon  
LegalNoticeCaption = REG_SZ Message from Afar
```

REGINI could be useful in situations where it is desirable to make batch, rather than interactive, updates to configurations.

Remote Addition, Deletion, and Modification of Users

Primary Tool: USRMGR.EXE. (User Manager for Domains)

Evaluation

User Manager for Domains provides remotely the same capabilities (and uses essentially the same interface) that User Manager provides locally. The User Manager is actually part of Windows NT Advanced Server, not Windows NT, and is targeted toward user management in a NTAS Domain environment. However, it can also be used to make configuration changes on Windows NT clients that are not members of NTAS Domains. Functions include:

- add and delete users
- modify user passwords, group memberships, and profile information
- modify system password and user audit policies
- modify user access rights
- add, modify, and delete groups

Other than the fact that some functions are not applicable in a non-Domain environment (and are therefore a bit confusing), User Manager for Domains is very easy to use.

Remote Addition, Deletion, and Modification of Shares

Primary Tool: SRVMGR.EXE. (Server Manager)

Evaluation

The Server Manager, part of NTAS, is a powerful and easy to use tool which, (among many other functions), allows management of shares on remote systems. With the Server Manager, shares can be:

- created
- modified (permissions, descriptions)
- deleted

Users can also be disconnected from active shares to the system.

Remote share management in Windows NT is as easy to do as local share management. See the earlier section "*Windows NT as a Simultaneous Client and Server*" for a bitmap which shows the sample "Shared Directory" screen.

Remote Execution of Programs at a Certain Date and Time

Primary Programs: AT.EXE, WINAT.EXE. (WINAT is a GUI front-end to AT.EXE).

Evaluation

AT (and WINAT) work in a manner similar to CRON on Unix systems. Programs can be executed remotely at a certain time and date, or on a regular basis ("every Thursday"). Character output can be redirected to another computer. (Windows NT today does not have the capability to redirect graphical output). In the example below, the DIRUSE command is executed on the computer \\UXRMG3125 at 4:00 PM, and the output (a listing of directories with greater than 20MB) is redirected to computer \\RCJONES3125.

```
at \\uxrmg3125 16:00 "diruse c:\ /s /q:20 /m /d >
\\rcjones3125\rcj\test"
```

	Size (mb)	Files	Directory
!	73.88	16	C:\
!	38.48	546	C:\WINDOWS\SYSTEM32
!	225.23	3516	TOTAL: C:\

Remotely Starting, Stopping and Querying the Status of Services

Primary Tool: SRVMGR.EXE. (Server Manager)

Evaluation

The Server Manager allows services to be started and stopped manually, and allows service startup parameters to be established. It allows the same capabilities remotely that the "Services" tool in the Control Panel provides locally.

An example of where it is very useful to be able to turn on a service remotely is when one is setting up remote Alerts in the Resource Manager. The MESSENGER service must be started on the remote machine before an Alert message can be sent. Using the Server Manager allows this to be done with ease.

A command line equivalent is NETSVC.EXE which can be used to start, stop or query the status of a remote service. In the example below, the machine PHOENIX is queried to ascertain whether the MESSENGER service is running.

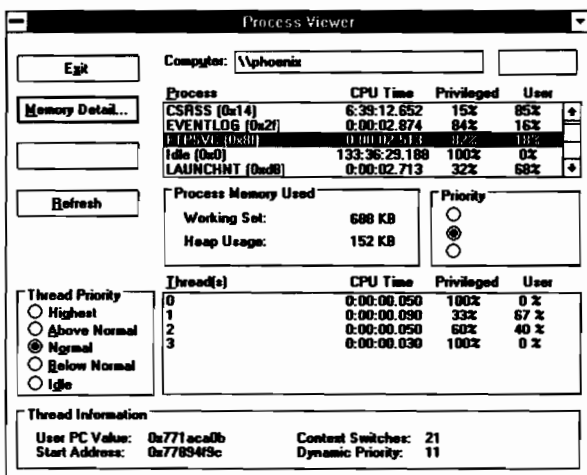
```
C:\>netsh messenger \\phoenix /query  
Service is running on \\phoenix
```

Remotely Viewing the Status of, and Killing Processes

Primary Tool PVIEWER.EXE. (Process Viewer)

Evaluation

The Process Viewer, available in NTAS and the Resource Kit, is a very powerful tool which allows viewing of information about running processes/threads on a NT system. When used for a local system, it also has a "Kill Process" function. For some inexplicable reason, the "Kill Process" function in Windows NT 3.1 is disabled when viewing the processes of a remote machine. This is unfortunate, as there are situations where this would be a useful feature. (i.e. A badly-behaved program hangs the 16-bit Windows VDM in Windows 3.1. Killing the NTVDM allows new 16-bit Windows applications to be run.)



Remote Backup of NT Configurations

Primary Tool: REGBACK.EXE. (Registry Backup)

Evaluation

REGBACK is a command line program which allows the remote backing up of the Configuration Registry. This could be done on a timed basis by using AT or WINAT. The backed-up Registry components ("Hives") can be redirected to another computer. In the example below, REGBACK is executed at the specified time on machine SAMICHA3125. The backed up hives are stored on machine RCJONES3125 (thus allowing centralized storage of backed-up Configuration Registries), and the output is also redirected to RCJONES3125.

```
C:\purgeme>at \\samicha3125 13:23 "regback
\\rcjones3125\rcj\hp\samicha > \\rcjones3125\rcj\hp\mm3476"

saving SECURITY to \\rcjones3125\rcj\hp\samicha\SECURITY
saving SOFTWARE to \\rcjones3125\rcj\hp\samicha\SOFTWARE
saving SYSTEM to \\rcjones3125\rcj\hp\samicha\SYSTEM
saving .DEFAULT to \\rcjones3125\rcj\hp\samicha\DEFAULT
saving SAM to \\rcjones3125\rcj\hp\samicha\SAM
saving S-1-5-21-14295905-356678474-608655777-1004 to
\\rcjones3125\rcj\hp\samicha\LMXAD000
```

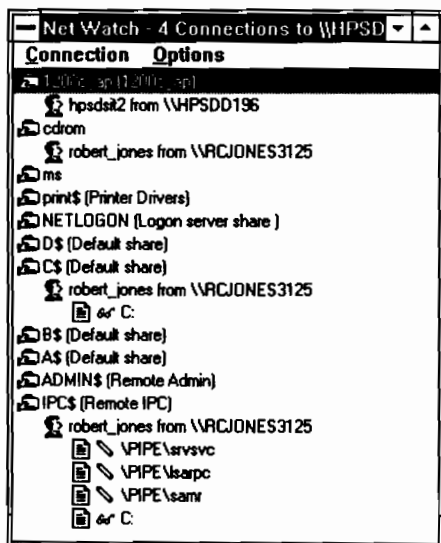
To test that this REALLY works, I used REGBACK to backup the Registry on my own machine, rebooted in DOS, and deleted my NT Registry files(!). A simple XCOPY restored the hives to the proper place, and I was able to reboot NT with no problem.

Remote Monitoring of Access to System Resources

Primary Programs: NETWATCH.EXE (Net Watcher), SRVMGR.EXE (Server Manager)

Evaluation

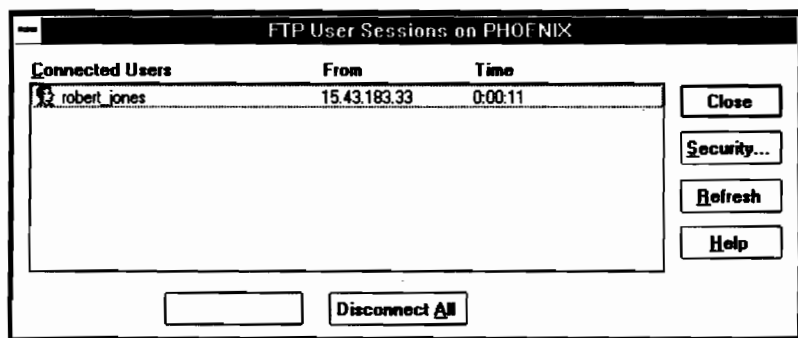
Windows NT allows easy monitoring and management of user sessions on remote systems. Net Watcher is a simple program which, at a glance, can show who is accessing what resources on a remote system. An example follows:



The more important tool in terms of monitoring and managing user access to a remote system is the Server Manager. The Server Manager allows monitoring (and disconnection) of:

- User Sessions
- Shared Resources (who is accessing what shares)
- Open Resources
- FTP sessions

As with all other GUI-based system management tools in Windows NT, Server Manager is easy to use. An example of a Server Manager screen is shown below. It depicts the screen which shows active FTP sessions on a remote system, and includes the capability for the remote system administrator to disconnect any/all of the active sessions.

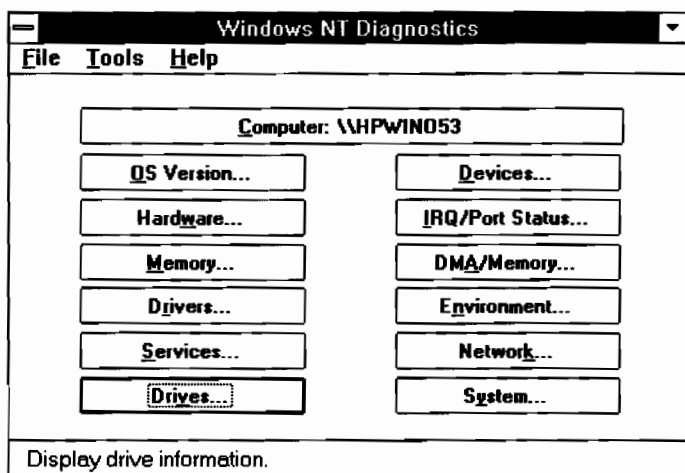


Remote Hardware/System Diagnostic Ability

Primary Tool: WINMSD.EXE. (Microsoft Windows NT Diagnostics)

Evaluation

Windows NT 3.5 has added remote support to the powerful WINMSD utility. As the following bit map shows, WINMSD can return a veritable plethora of information concerning the hardware/system status of a machine. Choices include OS Version, Hardware, Memory, Drivers, Services, Drives, Devices, IRQ/Port Status, DMA/Memory, Environment, Network, and System.



WINMSD also has a feature to print a report, or save the report to a file. The output below shows information about the OS version and memory of a remote NT system.

Microsoft Diagnostics Report For \\HPWINxxx

 OS Version Report

Install Date: Thu Feb 17 09:01:34 1994
 Registered Owner: UK Response Centre
 Registered Organization: Hewlett-Packard Ltd
 Version Number: 3.1
 System Root: D:\winnt
 Build Type: Uniprocessor Free
 System Start Options:
 Product Type: LanmanNT
 Build Number: 511
 CSD Number: 0

Memory Report

Available Physical Memory: 4,784 KB (4,898,816)
 Total Physical Memory: 20,032 KB (20,512,768)
 Available Paging-File Space: 20,952 KB (21,454,848)
 Total Paging-File Space: 44,280 KB (45,342,720)
 Memory Load Index: 0 %
 Paging Files: D:\pagefile.sys 27

Remote Troubleshooting

There are many ways that Windows NT can aid in remote troubleshooting. All of the tools below can assist in diagnosing problems remotely.

- Performance Monitor
- Event Viewer
- Registry Editor
- Server Manager
- User Manager
- Process Monitor
- Windows NT Diagnostic Tool

In addition, it should be noted that access to all files on a system is available through the C\$, D\$, etc. "special" shares. Of course, one must have admin capability (and the correct password) to use these. A statement such as NET USE X: \\PHOENIX\C\$ gives the remote administrator access to all files on the "C:" drive of system PHOENIX.

Remote Management Summary

The remote system management features in Windows NT go a long way towards making the PC remotely supportable (as Unix workstations are today). There are a few things missing from the Windows NT remote support/management inventory. Two prominent gaps that come to mind are the following:

- Redirecting graphical output to a remote NT workstation. In troubleshooting, it is often useful to be able to "see what the end user is seeing". Vanilla Windows NT does not yet have this capability.
- The ability to easily make configuration changes on multiple machines is a bit rudimentary in the vanilla Windows NT. However, the Domain Administration feature in Windows NT Advanced Server has a number of features which allow this sort of functionality.

Appendix A

In addition to Windows Sockets, Windows NT also includes the Microsoft RPC (Remote Procedure Call), a DCE-compatible, not compliant client/server communication mechanism. Hewlett-Packard has experimented with using the Microsoft RPC in our HP 9000, DCE server environment. The following observations were provided by Paul Lloyd of Hewlett-Packard CNS. It should be noted that most of the testing was done with the DOS version of Microsoft RPC, as opposed to the Windows NT version. However, the issues are very similar.

In general, our experience using Microsoft RPC has been good. Some specifics:

1. Initially, the fact that Microsoft RPC is not DCE-compliant caused concern, but once we had established to our satisfaction that interoperability did exist, it lessened as a concern. The fact that they chose a Microsoft style naming and calling convention rather than the OSF conventions is still a matter of concern. It means more training for development teams.
2. With respect to RPC interoperability, all supported data types work fine. There has only been one instance where I have found an incompatibility in the way MS-RPC and OSF wanted to marshal data; an investigation of the OSF spec indicates that the problem could be on either side.
3. The missing data types have caused problems. The absence of pipes and full pointers has made it difficult for application teams to use MS-RPC. Although the desired operation is usually possible with some other parameters, it can become much more convoluted than it should be. Furthermore, most application teams that I work with want to produce clients on both Windows and HP-UX; the fact that MS-RPC doesn't support these types means that both clients must suffer and take the least common denominator approach.
4. The lack of naming is a problem. Microsoft supplies a template for constructing a naming gateway, but it will not work with HP-DCE because HP-DCE does not support unique pointers. Our solution was to simply create our own naming gateway based on types that HP-DCE supports. This was not a huge undertaking, and it works fine; nonetheless, it would be ideal if MS-RPC included access to CDS.
5. The lack of security is a showstopper for many applications. Until MS-RPC includes obtaining credentials from DCE and using them in authenticated RPCs, it will not be a fully-functional RPC. One of the major appeals of DCE to application development teams is the powerful, robust security of DCE.

Sources

Title	Author	Date
A Guide to Windows Sockets	Martin Hall, JSB Corporation	6/93
Advanced Internetworking with TCP/IP on Windows NT	J. Allard, Microsoft	8/93
HP IT Experiences with Microsoft RPC	Paul Lloyd, Hewlett Packard	6/94
Integrating Windows NT into Hewlett-Packard's Present and Future Technological Infrastructure	Robert Jones, Hewlett Packard	2/94
Remote Management of Windows NT Clients	Robert Jones, Hewlett Packard	5/94

Paper Number: 4011
Title: Client / Server: Plug & Play or Crash & Burn...?

Ralph Bagen
Aircast Incorporated
92 River Road
Summit, NJ 07901
(800) 526-8785 x227



Perspective:

It is the purpose of this paper to provide a guideline on how to implement client / server; we will not be focusing on definitions of various terminologies but rather we shall examine approaches to justify, select and implement a client / server application, drawing primarily from a case study that I was responsible for throughout the past year to serve as an example (of both good and bad strategies).

One of the most significant points that I would like to illustrate is that when I decided to give this talk (around October of 1993 - after last year's Interex), I was convinced that by Denver, we would have our client / server application up & running - and that I would be able to stand here and tell you how smooth it all went. I had not appreciated some of the planning issues and various delays associated with implementing such an immature technology.

Perhaps the events of the past year make me less qualified to extol the virtues of a client / server migration - I personally think that my experiences make my implementation even more useful to this target audience.

Why did we attempt to implement a client / server solution?

Based on many of the reasons cited to implement client / server, we probably never would qualified if it was on an a "need to apply basis". We have a small (3 person) MIS Staff whose duties run the gamut, supporting a classic HP3000 "lights out" commercial transaction processing environment. No cry for open systems, no need to downsize, and no tired legacy system showing its age. Our single biggest problem was that our HP3000 ran our order processing and accounting functions, and that everyone else was working tirelessly at using stand - alone PC applications to run the rest of the business (from Sales Forecasting to MRP). The addition of an HP3000 Manufacturing package did little to alleviate the fact that all real analysis was down on the desktop, and the proliferation of PCs within the company and the support headaches of the "spreadsheet du jour" phenomenon was growing out of control. We needed to find a way to integrate our desktop applications with our HP3000 corporate databases seemlessly without taking away the users' desktop interface...

All but one of the company's (TurboImage) databases were integrated to our manufacturing module, but this was our "mission critical", home grown order entry package. Surely maintaining a bridge was the prudent, cost-effective way of protecting our investment in a highly-customized front-end of a product that everyone liked. Over

time, however, we realized that our business functions were changing; order processing was no longer "heads down data entry". Our systems had to be sophisticated and flexible enough to accommodate rapid change in the medical and health profession. The need for computer integrated telephony, fax on demand, image enabling of applications, EDI, warranty and lot control and above all, stellar customer service driven by business rules that will "empower" agents in the call center helped us reach the inevitable conclusion that our order entry application had to be re-written (or more correctly, the entire process re-engineered).

The obvious choice was to develop a "next generation" version that would be flexible enough to adapt to the many changes lurking in our industry. The superior application development tools offered by client / server, plus the integration of spreadsheet tools to our corporate data was everything we could have hoped for. The fact that we had the hardware infrastructure in place was a bonus - because of our over-reliance on PC tools, everyone was already beefed up on the desktop (486/33 or better!).

The first and most significant step we performed was to implement a Novell Network; if you gain no other insight from this paper than this, then it will still have been worth your time. The speed of communication on a network is night and day over RS232s over a DTC, and the Network Administration is a snap - we have almost replaced all of our dumb terminals and would completely if they didn't still work or had any re-sell value. Since everyone already had PCs on their desk, they were already using WRQ's Reflections (albeit serially) and accordingly costs were minimal to have them run over the network instead. Users were also ecstatic over printer sharing, file sharing, automated back-up and E-Mail.

The infrastructure was in place, the application chosen - next came the hard part!!!

Vendor, Database and Application Package Selection:

A lot of people are claiming to have client / server applications shipping today. Make sure that you get references and maintain an edge of skepticism on everything that you hear. If it sounds to good to be true, then it almost certainly is.

Our vendor selection process was relatively straight forward. Since one objective was to bring our orphan application under our vendor's integrated suite of manufacturing modules, and that vendor had a client / server offering, the project was theirs to lose. Do not, however, impose a client / server requirement on a vendor that has no strategic plan of their own to head in that direction - at best this will lead to exorbitant project overruns. Our particular vendor was offering a program called Headstart, wherein prospective accounts could become involved and have input in the actual development and shaping of a product at a significantly discounted rate before it becomes available to the general public. Their initial offering was a Customer Service / Order Entry module with built in hooks to the very same manufacturing databases that we toiled with maintaining our bridge to. A proverbial "no-brainer"!

Gotchas:

So we ventured into the co-development effort. The recommended Database Server of choice was an HP9000, running ORACLE 7.0. Having been strictly an HP3000 shop, part of the sign-off was to pursue HP-UX system administration training and as much

ORACLE as was to be required for daily database administration. If you have never used UNIX or ORACLE and you are strictly from a classic HP3000 environment, you just landed on Pluto. I quickly, realized that such a migration to this particular platform would require significant training and on-going database administration functions. Database packages in the client / server arena require constant massaging, as does disc space (de-fragmentation, data striping, extent size optimization etc.) - all of the issues that our small MIS staff had learned to take for granted on our HP3000. Staffing issues became a potential hurdle and an immediate cause for concern.

One of most eye-opening revelations occurred while attempting to optimize our ORACLE databases. During the database installation, primarily due to a self-confessed lack of ORACLE expertise, the rule seemed to be "when in doubt, take the default". This turned out to be one of the most important mistakes made during the optimization process. There is much within ORACLE (and you may substitute any comparable database of this genre) that requires customizing to your particular site specifics. This was a major contention with our provider who assured us that the "canned" implementation would be just fine. Obviously, a vendor who is not selling RDBMS consulting services is going to attempt to force a "one size fits all" approach for ease of implementation and site tracking. **DO NOT SETTLE FOR THIS.** In our instance, performance was so bad that we couldn't, but I hazard to guess that had performance been border line acceptable, then we would have never identified the factory settings within ORACLE that were hurting us.

Fortunately, HP and ORACLE committed to getting ORACLE 7.0 onto the HP3000 with MPE/iX 5.0 which we now have, and in the process this has enabled us to eliminate an entire operating system (HP-UX) from the equation. I also understand that ORACLE 7.0 was written directly for MPE/iX and, as such, takes advantage of many of the proprietary calls - initial performance benchmarks on the HP3000 platform have been favorable. The other significant by-product of this development was the elimination of an entire corporate business server (the HP9000). Remember that even although one of the goals with client / server computing is to establish the correct blend of processors and servers, the cost of maintaining a minicomputer such as an HP9000 is still significant. Furthermore, we have ample room to grow on our existing HP3000 configuration, and the introduction of an additional minicomputer may have required us to evaluate the need to downsize our existing production machine, particularly once the order entry application logic has been successfully transferred to the clients.

Application Development Packages:

Once again, this selection had been pre-ordained by our vendor. Their selection was the GUPTA suite of tools, with the application development done in SQL/Windows. After attending intensive training, it became obvious that there was a very steep learning curve with a huge payback at the end. One challenge here for the smaller shop is to determine whether or not you want to entertain in-house application development - if you choose to, managing upgrades becomes significantly more challenging. In our case, we are leaning towards performing cosmetic changes in-house, with all changes to the application logic being done by our vendor (this was one reason why we wanted to get away from COBOL code). In this way, we will minimize our upgrade pain by remaining mainstream in application code, but still have the ability to add, change or remove functionality in our graphical user interface in-house. The single most important key is to over-configure the database model on the back-end to anticipate every possible area

for growth or expansion. Flexibility is a major reason for the push to client / server. The ability to add and join tables on the fly, and the speed with which one can develop application code (or better yet, borrow from existing code) is truly a competitive advantage over environments using classic computing methods.

Establish Required Performance Benchmarks and Acceptance Criteria

From day one, the Headstart project was up against established performance benchmarks from our existing system (remember that we were not moving in this direction for any gain in performance). We had a very good understanding of exactly what was required to ensure the application be fast enough for our OLTP environment. At first, we were an order of magnitude away from our minimal acceptance criteria. Upon examination, we observed that our information flow would be improved by adding indexes to selected elements in our databases. The tradeoff here is that these keys will speed up information retrieval but slow down maintenance functions like customer modifications. Also, "striping" the data across multiple drives helped performance. The more drive mechanisms we could get to move concurrently to retrieve strategically spread out (striped) data, the faster the retrievals became. Another significant observation that became evident almost immediately was to be generous with memory on the database server. ORACLE ships with some reasonably useful monitoring tools. However, even after contracting third-party ORACLE consulting, the performance was still much too slow to be used in our OLTP production environment.

The Next Step

The entire client / server project was now at a cross-roads; we were about 30% invested in it, not counting the cost of internal resource drain and training expenses incurred thus far along the way.

We had plenty of horsepower, both on the clients and the server (in fact, we exceeded the recommended specifications across the board), unused bandwidth on our network - and yet we had an unusable prototype application. Some bottleneck had yet to be identified.

We focused our search energies to the two areas that had been left unexamined - the SQL / Windows code and the data base model itself.

The Application Code

Our first task had been to tailor the navigational aspects of the code to more closely resemble the flow of our customer service department. This essentially amounted to consolidation of many screens down to two, with the greater majority of prompts defaulted or assigned based of the value entered in one key field. It was at this point that we began to realize the true performance advantage that putting all of this application logic on the server would buy us. We really only have to interact with the database server twice on the average order - once to retrieve the customer record (and associated data) and again at the end when we have to write the order record (and update related tables). Everything else in between would take full advantage of the client's processor - the business rules, such as credit holds, or our elaborate discounting

logic, would no longer be a burden to the central processor. Each client locally would not be affected by the elaborate order that their neighbor was processing. Both the client and the server would function faster by having the graphical user interface and application logic on the client, and the server managing all data routing requests - a true two tier client / server environment, where each piece does the piece that it can handle most effectively! (see Figure 1)

All that remained was to find a way to expedite the data requests from the server. Since all requests from the clients were essentially the same (fetch a customer record), and that is all that is basically requested over and over again, the answer became obvious. Create STORED PROCEDURES anywhere that there are repetitive requests from the server to the client. This will greatly increase the speed at which these operations are performed, since they are running in a native mode on the database server. A further advantage with ORACLE 7.0 is that these stored procedures no longer have to be written in C (although they still can be), but that they can be written intrinsic to ORACLE, and in our environment, they will take full advantage of MPE/iX and Image intrinsics to accelerate the process. Additionally, ORACLE 7.0 takes full advantage of the ability to store the SQL / Windows code in memory cache on the client. Apart from the first time through, when you are processing orders on the client, you should never have to swap SQL statements in and out of the cache.

In light of the above, performance improved dramatically. At first, our vendors were a little reluctant to marry the application code to a database - that is, they wanted to implement stored procedures in C type languages for portability reasons, rather than write anything intrinsic to the back-end. However, after much persistence, they conceded, and now the product is truly tied to the site specifics in terms of work flow of the application logic, as well as being optimized to our particular hardware platforms (both the network and database server).

The Structuring of the Database.

A good database is meant to serve the applications that run on it. This rather simple, obvious statement should not be understated. There is a fundamental difference in database design between an OLTP / data entry environment, and a decision support or analytically intensive application. If you have to attempt to do both with one data model, understand going in that you will not be able to do either particularly well. Ideally, an OLTP environment's data model will be simplistic, with as few joins as possible, and indexed only where necessary for data retrievals - in other words, it should be as de-normalized as possible. The exact opposite is true in a decision support model, where there are great benefits to the use of indices and multi-joins - the more normalized the better. In general, if the records are read more than written, lean towards normalization, and conversely, if frequent updates are required de-normalize as much as possible. We had decided to keep our OLTP database model as de-normalized as possible, and map this data into a specifically designed database for decision support type analysis. It is my belief that the overhead of carrying (redundant) data in two forms is minute compared to the advantage gained in having each application optimized for performance.

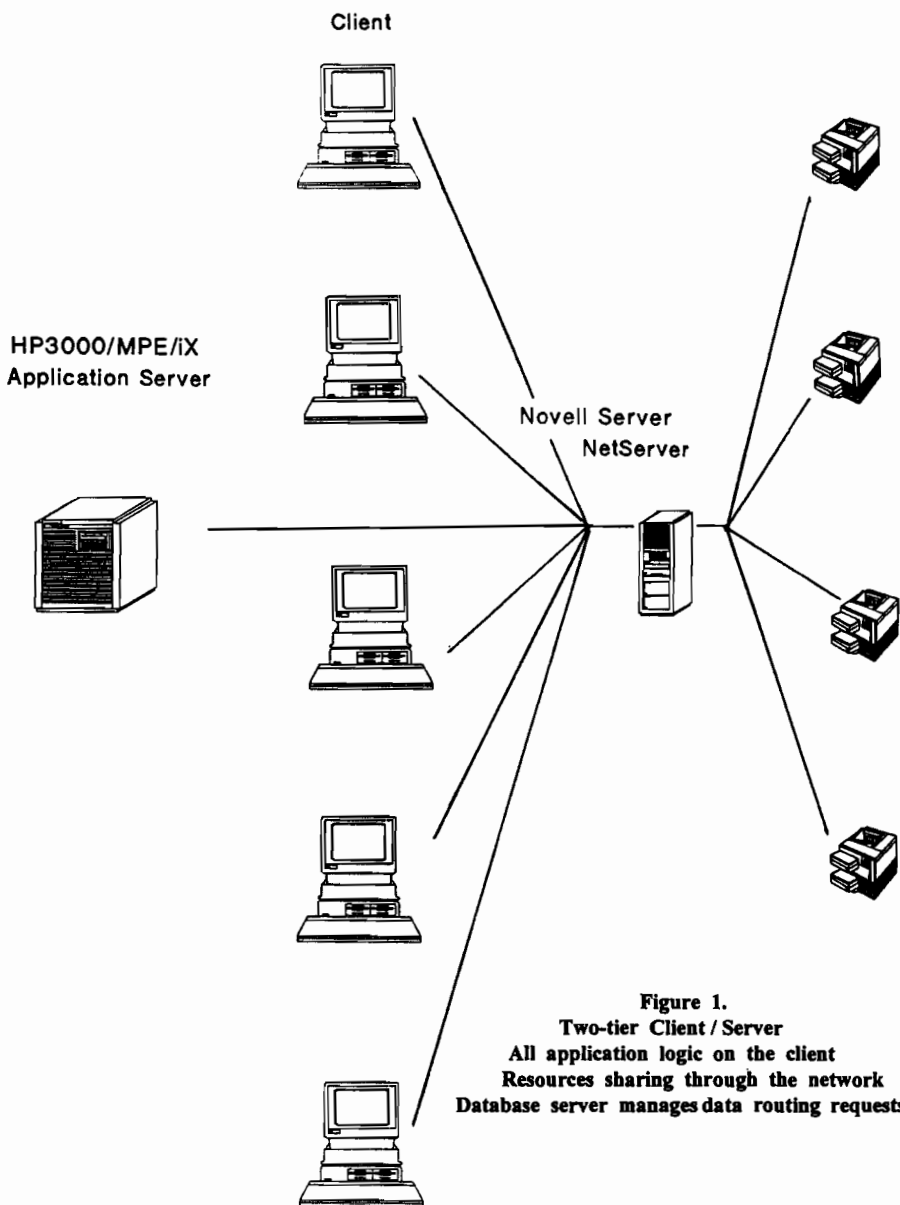


Figure 1.
Two-tier Client / Server
 All application logic on the client
 Resources sharing through the network
 Database server manages data routing requests

Client Concurrency

It is imperative that you factor in network traffic when measuring performance benchmarks. Establish test similar to that which you would experience during a regular (or irregular, for that matter) production day. Test applications that are developed against as many clients as you anticipate to one day be using it. Experiment at various times of the day, particularly if you are aware of known bandwidth contentions at particular junctures (i.e. large downloads, payroll, shop floor data collection transmits). It was very easy for us to recognize that our busiest times were after 3:00 o'clock, especially on Tuesdays. Also, experiment on the client. Try to simulate as closely as possible what will be executing on the client during a regular business day. Involve the users. Test your application concurrently whilst running a large spreadsheet or elaborate word processing document. Interact with E-mail. Maintain connections with other applications running on the host. Switch back and forth between applications. This is the time to identify problems.

In our testing, we discovered that several combinations of applications running concurrently caused Microsoft Windows to lock up without warning! Some solutions we are currently entertaining are:

- evaluate Windows NT, particularly if the only fast applications that you need have been written native for NT
- eliminate MS-DOS applications as soon as possible
- impose restrictions on concurrent applications on a client; this requires discipline to work
- there are several low-priced offerings available by utility vendors like Norton that monitors Windows resources and notifies when shortages are imminent
- look at products like Window Watcher for Novell
- keep current on new product release information on Windows Chicago, Daytona and Cairo.

Client / Server Myths

- Application develop is cheaper; possibly - if compared to a mainframe environment. The advantage that client / server developers have are superior tools and the re-usability of code, both of which result in greatly improved ease of modification.
- Client / server raises demand on the network - when implemented correctly just the opposite is true. Without any tools, users are moving vastly more data across the network than necessary. Even although bandwidth continues to rise and its price drop, by seizing control over who can access what, when and how, demand for bandwidth will go down dramatically.
- Client / server will proliferate applications on the desktop. There is a fear that if corporate data can now be accessed via ODBC drivers within a spreadsheet, that MIS has lost control. In fact, just the opposite can be true - MIS can regain a measure of the control that it lost on the desktop since the advent of PCs. The goal should be to eliminate "data outposts"; all strategically important data will reside on a database server for resource sharing considerations, and accordingly, will be managed and backed up by MIS.

Next Directions

We are now at the point where the front end performance and the back end performance are both adequate to implement. However, we still believe that the flexibility provided by our software vendor with their product yields a data model that is over normalized, and, consequently, less than optimal for OLTP. An other consideration, since we have eliminated the need for HP-UX - why have ORACLE as the database of choice? It brings with it the need for training, administration, and considerable costs - why not use the database package that we presently own - IMAGE / SQL? This makes particular sense if we have to de-normalize the data model anyway. In order words, it would be senseless to map the design inadequacies of the model when used in our environment from one RDBMS to another. If we are re-designing the model, tailor it to an OLTP form. Apart from the cost of ownership, additional factors are obtaining technical support from ORACLE, and, perhaps most importantly, that IMAGE / SQL will outperform ORACLE by 20 - 33% on an HP3000. If you use TurboImage as a 1.0 in relative performance scale, then IMAGE / SQL is about 0.8 and ORACLE and SYBASE are about 0.6 on an HP3000.

RDBMS	RELATIVE PERFORMANCE
TurboImage	1.0
IMAGE / SQL layer	0.8
ORACLE 6.0	0.6

Table 1

[Note: comparisons made on an HP3000]

With all of the above in mind, we may be much better served taking our SQL / Windows front end and implementing it on a de-normalized IMAGE / SQL back end on the HP3000 running MPE/iX 5.0. Note that the original design specifications called for a SQL / Windows front end to a highly normalized ORACLE 7.0 back end on an HP9000 running HP-UX 9.0. The only component that remained unscathed was the SQL / Windows front end! The major point here is that the application development tool (in this case, GUPTA's SQL / Windows product) is the critical choice - the operating system, network environment, hardware platform and database selection should be the secondary choice. It is imperative that you select a tool that will permit future growth. Remember that the client processor executes the graphical user interface and all program logic; the server merely manages database request routings and disk I/O. Understand that although these development tools are easier to use, most that are used for client / server development have to be database dependent.

The latter fact is something that we are presently wrestling with. Are we better off with stored procedures in native ORACLE optimized for the HP3000 running against an over-normalized database or should we utilize a faster RDBMS in IMAGE / SQL against a more suited data model using stored procedure calls in non-native C language? There exists legitimate concern regarding subsequent version upgrades in the IMAGE / SQL scenario. Essentially, it would necessitate a custom back end interface - one of the very reasons cited for switching away from our current application. Both propositions are eminently possible. Money will be the tie-breaker...

Miscellaneous Tips:

Convincing management is going to be the most difficult part of any client / server justification. In no way will you be able to claim do this to save money in the short term. It is very much a long range strategic decision. One statement that you may hear is "why should we switch when we have been so successful with our host to terminal methods?" The rebuttal to this is "everyone else has too!" If any competitor successfully implements client / server computing before you have considered it, they will have a significant jump start on you, and accordingly will have a significant strategic advantage. Change is difficult, but this is no harder than it was to implement "classic" host to terminal applications when it was first done. A common error that is made (and we fell prey to this) is to underestimate the network management, training and support factors. A good rule of thumb here would be to triple your estimate. There will be hidden costs that even the most savvy could miss.

View the implementation as a migration, not as "out with old and in with the new"; a successful venture should be built around your legacy investment. Think of it as a face lift. It is also very important to limit the initial scope of the project - make sure you select a controlled environment. Plan a pilot. The pilot should be viewed as a license to experiment. Do not try to implement your pilot - it should have changed beyond recognition and design scope before then! Most of the changes are made during the pilot. You must understand all the necessary data flows to perform within the application. Its importance is magnified in a client / server environment, since any redundancies will merely provide network traffic and processing overhead to your application. Involve your key users - this is an enterprise wide operation.

Remember that your company has committed to this for the long haul - avoid the temptation of taking short cuts, particularly when it comes to taking any deviations from established standards. They will only come back to haunt you - or your successor!

Finally, I want to say a word on troubleshooting. Never before has it been so important to have a rigorous approach - because the nature of the beast is linking together more independent components than in the history of computing. It is important to use a process of elimination to isolate problems. Identify when something stopped working - more things change and they change more frequently in a client / server environment. Do not assume that every client is affected. Do not assume that every client is affected. If it is a block of clients, that is a significant clue. Try to establish standards for client setups. Version control of memory managers, routers and software in general will be of huge assistance if your environment is standardized. Use your network to upgrade all software and try to standardize your hardware configurations - this is a golden opportunity to regain control over what is out there. Flowchart types of problems and form solution teams. In the beginning, document everything - you are like a newborn assimilating information in an exciting, but strange, new world.

Paper Number: 4012

Integrating Windows NT Clients into an Hewlett-Packard Server Environment (Case Study)

Robert C. Jones
Hewlett-Packard
2015 South Park Place
Atlanta, Georgia 30339
(404) 850-2887

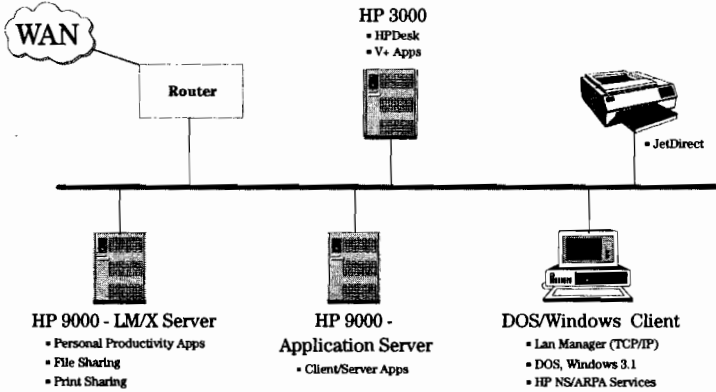
Hewlett-Packard Corporate IT, in conjunction with the HP United States Field Operations, began an evaluation of Windows NT as a potential PC client OS/NOS for HP's internal PC environment in 1993. This case study will examine a) what integration/migration issues were identified and b) how they are being addressed.

In early-1993, HP Corporate IT was in the process of architecting and rolling out a common PC operating environment (hereafter referred to as PC COE), which was comprised of the following components (hardware and software):

- DOS 5.0 (since upgraded to DOS 6.x)
- Windows 3.1
- Lan Manager 2.x
- HP NS/ARPA Services
 - FTP
 - TELNET
 - Sockets (WSOCKETS.DLL)
 - Domain Name Resolution
 - DSCOPY
 - VT (3000 terminal connectivity)
 - NetIPC (NS-based proprietary sockets interface)
- Server-based Personal Productivity Applications (word processing, commercial presentation graphics, spread sheet, etc.)
- HP9000/8xx file/print servers running LM/X (share level security)
- HP3000 servers running HPDeskManager and V+ applications
- 30,000+ 386/486 PCs
- Worldwide "HP Internet" TCP/IP network

The primary charter of the Windows NT evaluation was to answer the following question: "Can Windows NT integrate into the current/future HP-internal IT environment?". This case study will examine the integration barriers that were encountered in the evaluation, and discuss how those barriers are being addressed.

Today's HP IT PC Environment



Integrating Windows NT into HP's Internal PC Environment

HP IT is still in the process of evaluating whether Windows NT should become part of the internal-HP PC COE environment. As part of that evaluation (and as an aid to early adopters who have decided to migrate to Windows NT on their own), the key integration and migration issues were identified, and, where possible, addressed. They are listed below. A more detailed explanation of each follows.

- HP 3000 terminal connectivity
- HP 9000 terminal connectivity
- Sockets connectivity
- LM/X integration
- ADDNAME support
- Hardware requirements
- Performance
- Driver availability
- Installation
- Configuration
- Architectural issues
- Printing
- Training

HP 3000 Terminal Connectivity

Within the internal HP IT environment, a number of terminal-based HP3000 applications are in daily use, including HPDeskManager, and a number of other internally written applications. Thus, 3000 terminal connectivity is a must in the HP environment. PCs using PC COE achieve HP 3000 terminal connectivity through the TSR-based HP NS Services virtual terminal capability (VT). HP NS Services is NOT compatible with Windows NT. (Note: Serial connectivity to the HP 3000 from Windows NT is possible through terminal emulators such as AdvanceLink.)

Resolution: This problem was one of the toughest to resolve. The solution was provided by several vendors offering a 3000-VT capability based on the industry standard Windows Sockets 1.1 interface, instead of through HP NS Services. As Windows Sockets 1.1 (WINSOCK.DLL) is a core part of Windows NT, this completely resolved the problem.

HP 9000 Terminal Connectivity

Windows NT, of course, ships with a basic Telnet program. However, it does not support the termtype of "HP". PC COE today achieves HP 9000 TELNET connectivity through a terminal emulator (that supports the termtype of "HP") running over the HP ARPA Services BAPI and TN TSRs. As is the case with HP NS services, HP ARPA Services will not run under Windows NT.

Resolution: Once again, Windows Sockets 1.1 came to the rescue. A third-party vendor provided a TELNET capability over Windows Sockets, that also supported the "Termtype=HP".

Sockets Connectivity

In 1993, the majority of applications using sockets on the PC within HP were using HP ARPA Services sockets (WSOCKETS.DLL). A smaller number of applications were using HP NS Services NetIPC. Neither of these TSR-based solutions will run under Windows NT. NetIPC applications were already in the process of being phased-out within Hewlett-Packard, so this was a corner-case issue. However, not only were there a large amount of WSOCKETS applications in use, but a number of new ones were under development. Long term, we realized that migration to the Windows Sockets 1.1 standard was desirable, but such changes (especially to existing applications) are not made overnight!

Windows NT supports Windows Sockets 1.1 (WINSOCK.DLL), but does NOT support WSOCKETS. This remains a formidable migration issue for use of Windows NT clients within Hewlett-Packard. One factor that has ameliorated this problem somewhat is the amazingly rapid pace which Windows Sockets 1.1 has been accepted as a standard in the industry. Many internal and external HP application developers have added Windows Sockets 1.1 as a connectivity option to their client/server applications, and most of those that haven't already have plans to do so. However, the fact remains that client/server applications using WSOCKETS will not run under Windows NT, and this will be a definite inhibitor for some users within HP.

LM/X Integration

The PC file and print servers used within Hewlett-Packard are based on LM/X 1.4 ("share" level security). Windows NT clients have not been certified by HP to be

used with a LM/X server. A general principle used within HP IT is to not roll out production solutions which do not have vendor support. Ergo, this remains a migration inhibitor for using Windows NT clients within HP.

Of course, "Not Supported" does not necessarily mean "Doesn't Work". Early in the Windows NT evaluation, three main problems with Windows NT - LM/X 1.4 ("share" level security) connectivity were identified, and workarounds were provided by either Microsoft, or HP. (This allowed the Windows NT evaluation to proceed, but did NOT provide support for NT-LM/X connectivity.)

1. The first symptom was easy to identify - Windows NT clients could not connect to LM/X servers at all! The solution turned out to be the addition of the parameter #NOFNR to the LMHOSTS entry for a remote LM/X server. This flag disables Name Queries from Windows NT, and allows LM/X connectivity to take place.
2. Once that barrier was passed, the second symptom was also identified quickly - Windows NT sessions to LM/X servers timed out after about 5 minutes. Once again, a workaround was quickly made available - edit the Windows NT Registry to disable NetBIOS over TCP/IP "keepalives".
3. The third problem was a bit more subtle - A bug existed which caused Windows NT users to be unable to see any files on a LM/X server share, if any files exist in the LM/X share directory which contain a naming convention comprised of a "." followed by more than 7 characters. ("sh_history" is a good example). This problem was fixed in the "*Microsoft Windows NT and Windows NT Advanced Server 3.1 U.S. Service Pack 2*".

Once these three issues were dealt with, Windows NT clients were able to connect to LM/X 1.4 "share" level security servers reasonably well. (Note that additional problems exist for "user" level security, and the Windows NT clients remain unsupported with LM/X).

ADDNAME Support

ADDNAME, a Lan Manager 1.x and 2.x convention, is used to dynamically add host name to IP address mappings to the RFC NetBIOS cache on the PC. (When the PC boots, the LMHOSTS file is read, and the host name-IP address mappings found within are added to the RFC Cache. ADDNAME allows entries to be made to the Cache without having to reboot the PC). ADDNAME has not been ported to Windows NT by Microsoft.

A number of applications in use within Hewlett-Packard today rely on the ADDNAME program. The sequence listed below is typical of application use of ADDNAME:

1. User clicks on icon
2. Application executes an ADDNAME command (ADDNAME APPSERVER 15.xx.xx.xx)
3. Application executes a NET USE to APPSERVER.
4. The server portion of the application is accessed.
5. After the user presses exit, the application does a NET USE /D.

Windows NT works a bit differently. The LMHOSTS file is read dynamically, not just at bootup, so new entries to the LMHOSTS file do not require a reboot to be read. While in the long term, this is probably a preferable methodology, in the short term, it presents a migration issue for applications that have hardcoded ADDNAME use. One possible way to solve this migration issue is to write a simple new ADDNAME for NT/Chicago, which puts the host name - IP address mapping into the LMHOSTS file, and removes it when the connection is severed.

It should be noted that Microsoft is actively working on ways to remove dependence on the LMHOSTS file in Windows NT 3.5.

Hardware Requirements

Windows NT realistically requires a 486 with 16MB of memory to be an effective client. HP still has a large installed base of 386 PCs, and 486s with 8-12MB of memory. As a result, Windows NT can not be positioned as an operating system for all PC users within HP.

Performance

Many 16-bit Windows apps do not run as fast under Windows NT as under Windows 3.1 (although some functions are much faster, such as printing large, complex word-processing documents). Users wishing to make the jump to Windows NT need to realize this limitation "up front".

To get full use out of Windows NT as a desktop will require ready availability of 32-bit, native Windows NT Personal Productivity applications. At this point, it appears that the availability of such applications will coincide with the release of

"Chicago". As Chicago and Windows NT share the same API (Win32), most apps written for one platform should run on the other.

Driver Availability

A problem of any new operating system is the lack of ready availability of drivers for LAN cards, video adapters, printers etc. (16-bit real mode drivers do NOT work under Windows NT). Windows NT, though, had a remarkable amount of drivers available from initial release, and many more have been added since then. Today, most HP printers and LAN cards have Windows NT-compatible drivers. However, there will no doubt be some devices in use within HP that work under DOS/Windows 3.1, which will not work under Windows NT. Users will have to be aware of this before making the decision to use Windows NT as their desktop OS/NOS.

Installation

Few installation issues specific to the HP PC COE environment were uncovered during the evaluation. The issues that were uncovered include:

Dual-Boot or not Dual-boot? An important question that arises during the Windows NT installation is whether or not one wishes to have a dual-boot DOS/NT machine, and install Windows NT under the Windows 3.1 directory structure (\WINDOWS\SYSTEM32 instead of \WINNT\SYSTEM32). For the purpose of the evaluation, the decision was made to have a dual-boot environment, and install Windows NT under the Windows 3.1 directory structure. This allowed easy migration of the Windows 3.1 environment (Program Groups, desktop configurations, .INI files, etc.) into the Windows NT environment. This allowed the highest degree of integration with the existing PC COE environment (at the expense of 35-40MB of additional disk space used for DOS, Lan Manager, and Windows 3.1).

- **Custom vs. Express Installation** - The default protocol installed during the Windows NT 3.1 "Express" installation is NetBEUI. Therefore, the "Custom" installation option was desirable for the HP environment, as this allows TCP/IP to be installed and configured during the basic installation process. (The FTP Server, and the DLC protocol for printers might be useful to install during this sequence, also.)

Once TCP/IP has been installed, configuration information such as IP

address, the Gateway (Router) IP address, the DNS server IP address, etc. is entered. Windows NT gives the option of "importing" the existing Lan Manager 2.x LMHOSTS file to ..\SYSTEM32\DRIVERS\ETC\LMHOSTS. As many internetworked servers are used within HP, this option was chosen.

During the installation, the installer is given the option of joining a DOMAIN. As PC COE uses LM/X, the DOMAIN concept is not in use. The default WORKGROUP option was chosen instead.

- **FAT vs. NTFS** - During the evaluation, FAT was retained as the PC file system, rather than migration to the new NTFS file system. Again, this was to maintain the highest degree of integration with the existing PC COE environment.

Configuration

Several of the DOS applications used within Hewlett-Packard require DOS environment variables to be set. In PC COE, these environment variables are set in batch files CALLED from AUTOEXEC.BAT at bootup. Windows NT automatically sets any environment variables contained within the DOS AUTOEXEC.BAT file (if one exists), but it does not look for batch files that are CALLED from AUTOEXEC.BAT. The solution was very easy - global environment variables can be configured within NT from the "User Environment Variables" section of the SYSTEM icon in the Control Panel. The bitmap below shows an example.

System	
Computer Name: RCJONES3125	<input type="text"/>
Operating System	
Startup: MS-DOS	<input type="text"/>
Show list for 30 <input type="text"/> seconds	<input type="text"/>
System Environment Variables:	
ComSpec = C:\WINDOWS\system32\cmd.exe Os2LibPath = C:\WINDOWS\system32\os2\dll; Path = C:\WINDOWS\system32 windir = C:\WINDOWS	
User Environment Variables for robert_jones	
infonet = i:\hp\i path = i:\hp\bin temp = C:\temp tmp = C:\temp	
Variable: <input type="text"/>	<input type="text"/>
Value: <input type="text"/>	<input type="text"/>

Architectural Issues

The PC COE environment CALLs some network-based batch files at boot-up from AUTOEXEC.BAT. These batch files "set the stage" for the PC before Windows 3.1 is executed. For example, the batch files download Windows Group files to the client before executing Windows. This way, icons/groups can be added/deleted in a user's environment every time at bootup.

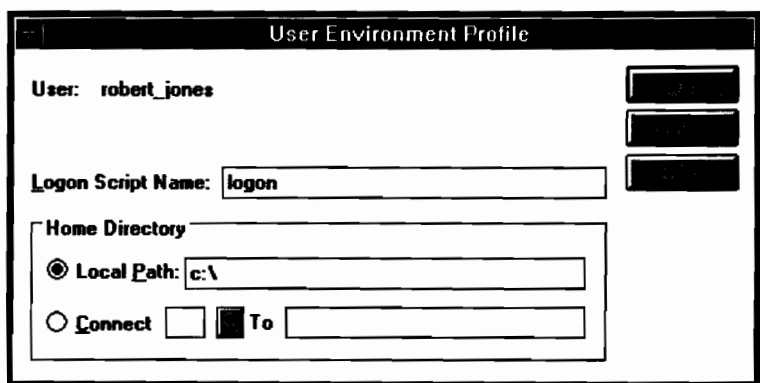
In Windows NT, of course, the networking loads "simultaneously" with the GUI. Thus, some architectural changes will be needed if the decision is made to make Windows NT part of PC COE. Windows NT *does* make it very easy to execute batch files or executables at startup, though. One method is to place whatever needs to be executed into the "StartUp" program group. Perhaps a more elegant

methodology is to place the executables/batch files that one wishes to have run at startup into the logon script.

Logon scripts in Windows NT 3.1 can be put under the following path:

..\SYSTEM32\REPL\IMPORT\SCRIPTS\LOGON.BAT

The next step is to assign this logon script to the desired user. This is done by accessing the user's profile via the "User Properties" screen of the User Manager. Type in the name of the script file that you have created for that user (in the example shown below, the script name is LOGON.BAT):



Note that logon scripts are assigned in the User Environment Profile on a user-by-user basis.

Printing

Windows NT is fairly flexible when it comes to setting up printers. In the PC COE environment, LM/X is used as the print server for DOS clients. DOS clients execute NET USE commands to logical print devices, such as:

```
NET USE LPT2 \\MYSERVER\PRINTER1
```

Windows NT allows this methodology to continue, for backwards compatibility. However, this method is rather limited, as only a few peripheral devices can be

configured at a time on a DOS client. To get around this limitation, Windows NT also has a "Connect to Printer" option, which allows one to set up a connection directly to a printer share. One could connect to \\MYSERVER\PRINTER1, for example, and this printer would be available every time NT is booted, without having to do a NET USE. If that printer is served off of a NT server, one can actually use the printer driver on the server, rather than on your client. In a LM/X server environment, though, the client must provide a printer driver (as is required under the existing Windows 3.1 environment).

Windows NT also has the capability to act as a print server itself. NT can act as the server for printers physically attached to the NT box, or for JetDirect networked printers. In the case of JetDirect printers, the DLC protocol is used to communicate between the NT server and the JetDirect printer. As LM/X servers were already in use as Print Servers in the PC COE environment, it was not necessary to use Windows NT clients as print servers.

Training Issues

From an end user perspective, Windows NT *looks* very similar to Windows 3.1 - the user interface is essentially the same. Thus, users should be able to find their way around Windows NT fairly easily. The necessity of a logon with a password will be new for some people, and will require some familiarization, as will other NT-specific functions.

A much larger training program would be needed for support people, as Windows NT is a completely new operating system. Luckily, many of the skills that support people already have in the areas of Microsoft Windows, Lan Manager, TCP/IP etc. can be directly leveraged to support Windows NT. Also, familiarizing PC support people with the remote management features of Windows NT will require training, as well as a change in thinking - in the past, most troubleshooting, administration, and management of PCs within HP had to be done "hands-on". The ability to do many functions remotely should simplify the life of the PC support person, but it will require a new support process.

Summary

As would be expected with any new operating system/network operating system environment, Windows NT presents a number of migration and integration issues for HP IT. However, as has been outlined in these pages, none of them seem insurmountable. Windows NT presents some interesting possibilities for the internal HP PC environment, and it will continue to be evaluated.

Integrating HP OpenMail and HP 9000 Fax

Tony Jones
Hewlett-Packard Company
20 New England Avenue
Piscataway, NJ 08854-4107
(201) 562-6248

HP OpenMail provides enterprise wide, client/server based electronic messaging. HP 9000 Fax provides a client/server based outbound and inbound fax solution. This paper describes how to integrate the two products to provide a more effective solution than what is currently available. The features of this solution include:

- Fax modems directly attached to HP 9000 RS/232 ports instead of an external fax server. This can lead to a lower cost solution.
- Fax capability available for any program that can send its output to HP 9000 based printers.

The topics discussed in this paper include:

- Introduction to HP OpenMail architecture
- Introduction to HP 9000 Fax
- Techniques used to integrate the two products

Upon completion of the session, the participants will have a better understanding of how to:

- 1] Interface existing applications to HP 9000 Fax
- 2] Electronically mail application program output
- 3] Integrate existing applications to derive new solutions

The presentation will include a live demonstration of faxing an electronic message and receiving a reply fax.

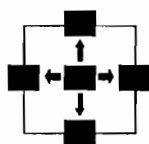
Introduction to HP OpenMail architecture

OpenMail is a software application that provides electronic mail and other messaging services. It is compliant with the X.400 Recommendations and provides facilities based on international standards to exchange and manage information.

OpenMail consists of a user agent Application Programming Interfaces (API) for communication with the client applications, a message store that holds messages, messaging-optimized Directories and connection to X.500 Directory systems, interfaces that connect to transport systems, and gateways that link to other systems.

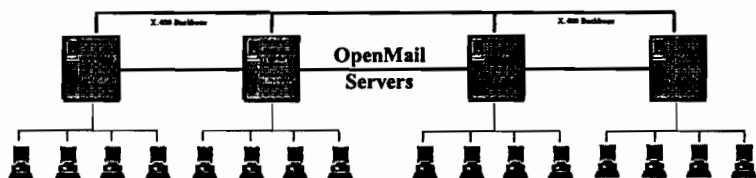
OpenMail

Typical OpenMail Architecture



- Variable Lan Topologies
- Integrated Message Store, Directories, APIs and MTAs
- Non-Hierarchical Physical Structure
- Standards Based

- ★ Scalable
- ★ Manageable
- ★ Secure
- ★ Reliable
- ★ Standards Based
- ★ Flexible



Cooperative Computing Systems Division

Figure 1. HP OpenMail

OpenMail uses a client-server architecture to provide electronic mail facilities. The OpenMail architecture consists of three main parts:

- 1] Interfaces to the clients, the X.400 Message Transfer agent, and to Sendmail
- 2] Gateways to other messaging systems
- 3] Remaining services, such as local delivery and message routing.



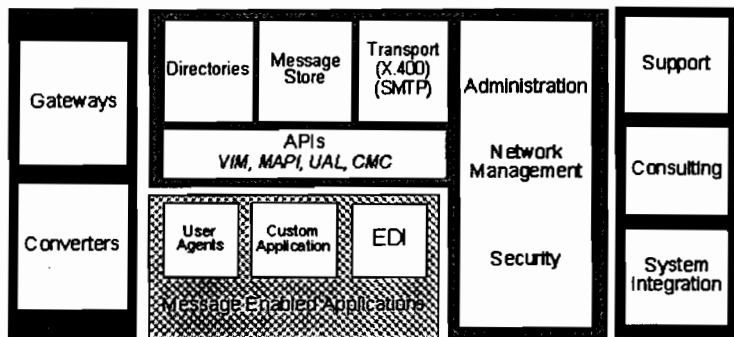
Integrated Messaging Services

HP OpenMail

Accessories

Messaging Services

Services



Corporate Computing Services Division



Figure 2. HP OpenMail Messaging Services

Every user in OpenMail is addressed in the following manner:

GivenName SurName/OrgUnit1,OrgUnit2,OrgUnit3,OrgUnit4

The four organizational units make up the mailnode. A user can have from one to four organizational units.

Example:

Tony Jones/hp,piscataway

Users who are on external mail systems such as the Internet can be reached via the Unix gateway. To address users on external systems, use the Foreign Address field area which consists of the "name@domain.organization" convention surrounded by parentheses after the mailnode.

Example:

Jim Smith/hp,unix(jsmith@compusa.com)

OpenMail supports a command line interface to access various functions including:

<u>Command</u>	<u>Description</u>
omsend	Send an electronic message
omread	Read an electronic message
omnew	List new messages
omdelete	Delete a message
omsearch	Search a directory for user(s)

Programs can use the command line interface to electronically send program output. For example, to send a report file "report.out" to a user called Tony Jones:

```
omsend -t "Tony Jones" -s "FY'94 report" -r "report.out" -u $USER -p $PASS -q
```

The -s option is the subject, the -u and -p options are for the OpenMail user and password.

Request Server

The OpenMail Request server allows a user to mail a request to execute a script and get the results in an electronic mail message. These scripts can provide functions such as return the amount of free disc space on the OpenMail server. Access control lists are also available to control who can access the scripts. To access the request server, replace the GivenName with the request (name of the script), and use "+req" as the SurName:

Request +req/ou1, ou2,ou3,ou4(Foreign Address Field)

Introduction to HP 9000 Fax

The HP9000 Fax product is a client/server based solution that allows users to send and receive faxes. In addition, inbound fax routing is available through the use of a barcode pattern located on the fax cover sheet. The fax server resides on one server and is available to multiple HP9000 clients over the network. It is also incorporated into the HP MPower product. Fax modems are directly attached to the HP9000 instead of using a PC based fax gateway. This technique requires less equipment, and also allows the modem to be occasionally used for inbound or outbound connections when fax functionality is not required. Incoming faxes can also be sent to a printer or a fax machine.

There are three system processes associated with HP9000 fax:

<u>Process</u>	<u>Description</u>
faxemd	Electronic mail process used to send faxes
faxsched	Schedules outbound faxes and returns status
faxlisten	Listens to the modem port waiting for incoming faxes

The client interfaces include:

- Motif based application
- Electronic mail interface
- "lp" printer interface

Note: the mail and lp interface currently only support text messages.

To send a fax via the electronic mail interface:

Mail message to faxemd@hostname
Read in cover sheet and fill in fields
Add message text
Exit editor with save option
Send the message

The faxemd process will send a mail message reply containing the results of the fax request.

To send a fax using electronic mail, route the mail containing your fax through a system that is running the HP9000 fax server. Include the sender and recipient information and the message all together in the mail text.

Example: Send an electronic fax using the "elm" mailer. The host with the fax server is called "faxsys".

1] Start the mailer

```
elm faxemd@faxsys
```

2] When prompted for a subject, press [RETURN]. You can choose to use the "subject:" keyword in the cover sheet section to describe the subject of the fax. Press "y" when prompted to continue with message.

3] When prompted for "Copies to:", press [RETURN].

4] Elm starts an editor (default is 'vi') session to allow you to enter text. Enter the following fields with the associated information.

```
from:  
style:  
class:  
account:  
subject:  
to:  
fax#:  
message: First line of text  
Second line of text....
```

5] Exit from the editor using a command that saves the file (:wq!).

6] When prompted for what to do with the file, type "s" to send the file.

To reduce typing the cover letter, save the cover sheet portion in a file and read it into the message, then replace the information in the "to:", "fax:" and "message:" fields.

The faxemd process will pick up the fax message and fax the contents of the message.

Printer interface

To use the "lp" printer interface, you need to specify the fields as options to the command.

The minimum number of fields required to send a fax are:

from:
style:
class:
account:
to:
fax#:

Example: To fax a message contained in "/tmp/message" to Dee Wallace at (908) 562-6246, use the following command:

```
lp -dfax01 -o'from: Tony Jones\  
-o'style: Basic (High Res)\  
-o'class: Standard\  
-o'account: Sample\  
-o'to: Dee Wallace\  
-o'fax#: 9,19085626246' /tmp/message
```

Most of the cover sheet information can be stored in a defaults file to reduce typing. The "return_envelope" keyword can be used to facilitate incoming fax routing back to the original sender. This option tells the fax server to include the sender information and a barcode return address with the outgoing fax. The recipient can use the return envelope to reply to the send. When the return address is detected by the fax server, the incoming fax will be routed to the original sender's mailbox. The fax can then be viewed (by the Motif client) or printed.

Note: You will notice that the "lp" model script for the fax invokes the faxemd process. This shows how a process normally associated with mail can be used for printer interface functions.

Techniques used to integrate the two products

Use the OpenMail request server to communicate with the HP9000 Fax server faxemd. Read the OpenMail message, re-format the environment variables, place the last body part of the message in a file, then call the faxemd process.

The Request server provides five items to a script:

- Four environment variables:

<u>Variable</u>	<u>Description</u>
LANG	Language used by the sender (english, etc)
OMSUBJECT	Subject field of the message
OMSENDER	Sender's name (Given & Surname) and mailnode
OMRECIPFA	Foreign address field of recipient's information

- Last body part of the message

To access the Request server to send a fax using our fax script, use the following format:

```
fax +req/ou1,ou2,ou3,ou4(Name@fax_phone_#)
```

```
Subject: _____
```

The Foreign address field typically looks like: name@host.domain.organization. In our case, we will provide the recipient's name and fax telephone number separated by an "@" sign.

Example:

```
From: Dee Wallace/hp,piscataway
To: fax +req/hp,fax(Tony Jones@9,19085626246)
Subject: Directions to stadium
```

Here are the directions to the stadium:

- 1] ...
- 2] ...
- 3] ...

Good luck,
Dee

Refer to the "fax" script in appendix A to see a sample of how to integrate OpenMail and HP9000 fax. Note: the script is a prototype to show proof of concept. Additional error checking and enhancements need to be added before this should be employed in a production environment.

How to:

Interface existing applications to HP 9000 Fax

Use the "lp" printer interface
Use the electronic mail interface

The application can mail output to the fax server or print output to a destination printer which is being simulated by the fax server. If the application uses the mail method, the cover sheet parameters should be listed in the beginning of the message. If the application is using the "lp" printer interface, the cover sheet parameters must be specified in the "lp" command line. Refer to the fax printer model script for more information on how to invoke the faxemd process.

Electronically mail application program output

Programs can use the command line interface to electronically send program output. For example, to send a report file "report.out" to a user called Tony Jones:

```
omsend -t "Tony Jones" -s "FY'94 report" -r "report.out" -u $USER -p $PASS -q
```

The -s option is the subject, the -u and -p options are for the OpenMail user and password.

Integrate existing applications to derive new solutions

Summary of OpenMail to Fax integration

Find common interfaces:
 Electronic mail interface
 "lp" printer command
 Command line interface

Select an interface that is reasonable to work with given constraints such as available environment variables provided OpenMail test server.

Find documentation covering interfaces

Format the information into a form that is recognizable by the appropriate process such as faxemd.

Use the OpenMail Request server to allow user to send a fax.
Format information into required files for faxemd
Pass the files to faxemd
User should expect to receive two mail messages after sending a fax

Acknowledgments

I would like to thank Stefan Buerger and Kumar Rangan of the HP Professional Services Organization for providing the foundation for the fax Request server script.

Appendix A.

Fax Request Server Script

```
#!/bin/sh #####  
#  
# OpenMail to Fax/9000 using the request server  
#  
# Original Script by Stefan Buerger, PSO - HP Germany  
# Modified by Kumar Rangan, PSO - HP New Zealand  
#  
# If this script is called /usr/openmail/req/fax  
# Address a message to fax +req/ou1,ou2,ou3,ou4 (foreign address)  
# we expect the contents of the foreign address  
# to be in the format receiver@faxnumber where  
# receiver is in the format firstname lastname  
# example : Kumar Rangan@3843380  
# No spaces in the fax number, please ....  
#  
# variables available from OpenMail : OMSENDER, OMRECIPIFA, OMSUBJECT  
# LIMITATIONS :  
# - No spaces in the fax # part of the foreign address  
# - Conversion of the fax message to PCL format (from PC  
# based packages) is to be done *before* submitting the  
# message to OpenMail. No worries if it is only text ...  
# - Only 1 body part can be sent as the fax message. The  
# request server will accept only the *last* body part in  
# the message, the other parts are discarded by the request  
# server.  
# - To make life easier for you, use only Fax/9000 supported  
# modems. I used the Multitech MT1432 modem successfully.  
#  
#####  
  
PATH="/bin:/usr/bin:/usr/lib:/usr/fax/lib"  
export PATH  
TZ=NZST-12NZDT  
export TZ  
  
FAX_SENDER=`echo $OMSENDER | awk '{print $1, $2}'`  
  
FAX_SUBJECT=$OMSUBJECT  
  
# extract the receiver's name from the foreign address
```

```

FAX_RECV=`echo $OMRECIPFA | tr "@" " "|awk '{print $1,$2}'`
# extract the receiver's fax number from the foreign address
FAX_NUMBER=`echo $OMRECIPFA |tr "@" " "|awk '{print $3}'`

# If you manage to find a generic PCL converter, use it here.
# Currently we assume that the document will be in text format
# or PCL conversion is done before the document is passed on
# to the OpenMail Client
PCL_CONV="/bin/cat -"

WORKDIR=/tmp                # faxemd works only with /tmp
FAX_PROG=/usr/fax/lib/faxemd

FAX_TMP=$WORKDIR/faxout.tmp # The last body part of the fax
HDR_TMP=$WORKDIR/header.tmp # The header info for the fax

rm -f $FAX_TMP
# cat - > $FAX_TMP
$PCL_CONV > $FAX_TMP

# convert senders OpenMail address to unix mail format, so fax/ux can send
# an acknowledgement email message to the senders OpenMail mailbox

rm -f $HDR_TMP
M_SENDER=`echo $OMSENDER | awk '{print $2, $1 $3}'| tr " " "_" | tr ", " "_`

# build the fax/ux header file
(
echo "From $M_SENDER "`date`
echo
echo "FROM: $FAX_SENDER"
echo "STYLE: Basic (High Res)"
echo "CLASS: Standard"
echo "ACCOUNT: Sample"
echo "SUBJECT: $OMSUBJECT"
echo "TO: $FAX_RECV"
echo "FAX#: $FAX_NUMBER"
echo "ATTACH_FILE: $FAX_TMP"
echo "END:"
) > $HDR_TMP

## Example Header #####
#
#From Rangan_Kumar/hpnz_pso Wed Jan 5 16:00:52 NZDT 1994
#
#FROM: Kumar Rangan
#STYLE: Basic (High Res)
#CLASS: Standard
#ACCOUNT: Sample
#SUBJECT: Test Message
#TO: John Doe
#FAX#: 3843380
#ATTACH_FILE: /tmp/faxout.tmp
#END:
#
## Example Header End #####

# now hand it to the fax-email daemon ...
rm -f /tmp/faxlp.lock

### original statement from "faxlp" #####
# faxemd 1 0 ' ' /tmp/faxlp.in.q' /tmp/faxlp.err' "$file" /tmp/faxlp.lock'

```

#####

\$FAX_PROG 1 0 ' ' /tmp/faxlp.in.q' /tmp/faxlp.err' "\$HDR_TMP" '/tmp/faxlp.lock'

R_CODE=\$?

acknowledge processing of request ...

echo "Your fax message : "

echo "subject: \$OMSUBJECT , for: \$FAX_RECV , on Fax#: \$FAX_NUMBER,"

echo "was received and processed at the "`hostname`" Fax Gateway,"

echo "at `date`"

echo "Return Code from the Fax Server was \$R_CODE"



Paper Number: 4014

**Evaluating and Implementing
Office Automation Solutions**

Eric Kowalski & Ken Smith

Higley & Company, Inc.
1920 West Eighth Street
Erie, Pennsylvania 16505

(814) 454-5215

Office automation is a critical success factor in today's business environment. Automation can improve communication, shorten decision-making time, improve focus on business goals and flatten the organizational hierarchy. Automation can also reduce training, support, and labor costs and provide tighter security within the organization.

Companies must consider their strategic goals before deciding to automate. Automation can increase the efficiency and effectiveness of employees, improve access to data, provide the ability to communicate anywhere, anytime, with anyone, supply the proper tools to employees and ensure they are being used properly. Automation increases employee efficiency and effectiveness by having the data available right at the employee's fingertips. This lessens the time it takes to gather the information and process it.

In order to reap the benefits of office automation there are factors that must be taken into consideration. A company that wants to automate needs to establish their company's strategic direction, determine a suitable office environment, decide among the numerous software packages and suites available, and keep the issue of security always in their minds while making decisions.

OFFICE ENVIRONMENT

Two important issues of office automation deal with hardware and software decisions. There are really two options for hardware configuration. The first is PC-based configuration. A PC-based configuration does not have a dedicated server, which means the server may also be used as a workstation. In a host-based configuration there is a designated processor, a dedicated server. This machine's only function is as a server. It can not perform any workstation functions. The host-based configuration is sometimes more expensive, but is more efficient than the PC-based configuration.

CROSS-PLATFORM CAPABILITY

Cross-platform capability suggests that a document created on one system (i.e., Windows) can be imported and used on another system (i.e., Macintosh) without any significant changes or loss of data. This provides an immediate productivity advantage because no changes means that it is easier and faster to share and recreate information. In 1992, UNIX was the only operating system that provided cross-platform compatibility, graphical user interface (GUI), DBMS, and multi-user, multi-tasking capabilities all bundled together in a single package.

Cross-platform compatibility becomes a big issue if your company supports more than one operating system. In order to import a document from one application to the same application on a different operating system, the application software must be cross-platform compatible. Therefore, look carefully into the software applications that you purchase.

INTEGRATED SOFTWARE - "SOFTWARE SUITES"

Software suites are groups of freestanding applications sold as sets, that share the same basic user interface. The applications in the suite are fully integrated so users can move data easily, from one application to another. The software suites are normally discounted from the individually purchased program prices.

The price makes the software suites attractive, however, there are disadvantages so don't shop on price alone. One disadvantage is that user's may not be willing to accept the new application over their old one. Users may not be willing to give up their favorite application to try and learn a new one.

The main disadvantage is that users may not be getting the best product for each application. A rival company may have a better spreadsheet, however, you would be stuck with the one purchased in your suite. Be sure when purchasing a suite that you are satisfied with the application packages it contains.

Microsoft Corporation and Lotus are the market leaders in sales for PC-based office suites. Microsoft Office combines Word, Excel, PowerPoint, and Mail. The applications are fully integrated. It is supported on the Windows platform and costs \$750.00. Microsoft Office is supported by third party software such as AutoMac III, Macro Recorder, SmartScrap Desk Accessory, QuickBasic, Docucomp, and AutoSave II. Microsoft was the first company to use Object Linking and Embedding (OLE). OLE is a new communication technology. It brings us one step closer to object oriented programming, building packages from small modules.

Lotus' Smart-Suite includes Ami-pro 1-2-3, Approach, Freelance Graphics, and Organizer. There is no mail function available with Smart-Suite. However, Smart-Suite's third party software includes Lotus Notes and cc:Mail so either package could be incorporated for mail features. Smart-Suite sells for \$795.00.

Borland International is teaming up with WordPerfect to introduce a new suite. It includes WordPerfect, Paradox, and Quattro Pro. It is run on Windows and costs \$595.00.

WordPerfect also has a product called WordPerfect Office. This package includes WordPerfect, a calendar/scheduler, mail functions, WPEditor, a file manager, a notebook, a calculator, and WPTalk. There are versions available for MS-DOS, MS-WINDOWS, Macintosh, UNIX, and Dec VAX/VMS. A five user license is only \$495.00 which makes the per user price \$99.00. However, there are no established third party software developers.

In the host-based environment, Uniplex has developed Uniplex II+. It includes a word processor, a database, a spreadsheet, and screen and menu builders. All of these applications are fully integrated. Uniplex II+ is supported in the UNIX and DOS environments. It costs approximately \$475.00 per user. Uniplex II+ can import WordPerfect and Lotus documents and has over 200 third party software vendors, as well as add-on modules for calendaring, business graphics, faxing and more. It also operates in many user interfaces including text, PC, X-station, and DOS.

IMPLEMENTATION AND TRAINING

Training is a very important factor in office automation. If users have proper preparation and training they are more apt to accept the new system. Before training individuals managers need to identify who will be using the new technology; what they will be using it for, how it will effect their productivity, and what they hope to accomplish with it.

Good training will provide:

- higher productivity
- reduction in labor costs (due to need for less staff)
- improved accuracy
- improved quality of output
- increased timeliness
- faster decision-making
- increased creativity
- better communication
- easier sharing of information

In the implementation and training phase there are rules of thumb that will aide in making a smooth transition. Involve the users in as much of the process as possible. Have manuals available for users to reference. If training takes place in the classroom, limit the class sizes.

Schedule the training so as not to interfere with department procedures. Train the users for skills AND for attitude. Finally, present the system to the users in a manner that increases their perceived value of the system.

Once the system is in place, motivate employees to use the e-mail and scheduler features. This will aide in their satisfaction with the new system. Also, make sure that users' responsibilities are clearly outlined and that users are working in a controlled environment.

CUSTOMIZATION

Some packages provide the ability to tune their menus and extract data into an executive management system designed to reflect the needs of management. This can be important when looking at the differences in user requirements. Many PC-based solutions are solution - WYSIWYG (what you see is what you get).

SECURITY

Computer crimes continue to grow and plague U.S. companies. Therefore, security is a major issue when making the decision to automate. Everything from the data communications link to your disks to your personnel are potential security risks.

Due to threats of virus and worm infections that can spread through floppy disks, it is recommended that CD-ROM be used to install programs and applications. CD-ROM are safer because they are read-only.

It is suggested by some that dial-up access is the best type of installation to have because it can potentially stop hacking. This is done through passwords. The passwords are generated every 30 seconds and not all passwords are actual words. However, dial-up products are expensive.

Others say that batch processing is the most secure installation because the data is processed according to a schedule. Companies must consider security issues when determining the type of processing to be used when installing a computer system.

Some ways to reduce risks are to practice risk management, be aware of security issues, plan for business resumption, limit computer access, rotate user identifications and passwords, and screen new employees carefully.

MAINTENANCE

Another important consideration in office automation is maintenance. Many companies charge on-going fees or fees for upgrades. Be sure to determine the amount of excess fees your company will be undertaking when installing a new system.

SUMMARY

In choosing a system for office automation, remember to first define your company's strategic goals. This will aide you in choosing between the host or PC-based environments. In selecting a software package to run on your system, remember not to choose on price alone. Look at the individual applications in the software package to determine if each will meet specifications. Also, when selecting software it is important to consider security issues and maintenance fees.

Most importantly, take your users into consideration in your implementation plan and train them well! The better trained your users are, the more benefits you will obtain.

ACKNOWLEDGEMENTS

Many thanks to Chris Higley for the research and help to get this paper together.

Individual trademark symbols have not been placed on every occurrence of trademarked names. We state here that we are using the names in an editorial fashion and to the benefit of the trademark owner with no intention of infringement of the trademark.

WORKS CITED

- Bielawski, Steve. "Conquering the Cross-Platform Issue." *Computing Canada*. Vol: 18 Iss: 8, April 13, 1992, p: 46.
- Brandt, Richard. "Breaking Down the Walls Between Programs." *Business Week*. Iss: 3342, October 25, 1993, p: 126.
- Brown, Carolyn M. "All Together Now!" *Black Enterprise*. Vol: 24 Iss: 8, March 1991, p: 36.
- Finlay, Doug. "Management Has Key to Data Security." *Office*. Vol: 114 Iss: 5, November 1991, p: 76, 85.
- Frollick, Mark N. "Management Support Systems and Their Evolution From Executive Information Systems." *Information Strategy: The Executive's Journal*. Vol: 10 Iss: 3, Spring 1994, p: 31-38.
- Radding, Alan. "Software Suites' Ups, Downs." *Computerworld*. Vol: 26 Iss: 11, March 16, 1992, p: 101.
- Remenyi, Dan. "What Is End-User Computing and Why Is It Important?" *Management Accounting*. Vol: 68 Iss: 11, December 1990, p: 31.
- Robson, Jane. "It's the Human Element That's Important In PC Training." *Computing Canada*. Vol: 16 Iss: 18, September 4, 1990, p: 37.
- The, Lee. "Downsize Your Database with UNIX." *Datamation*. Vol: 38 Iss: 21, October 15, 1992, p: 65-68.

Environmental Systems Research Institute, Inc.
380 New York Street
Redlands, California 92373
(909) 793-2853
Tony Burns

USING GIS FOR BUSINESS APPLICATIONS

BACKGROUND

As far back as twenty years ago there emerged a new technology that revolutionized the ability to analyze written tabularized data. Data, for the first time, could be attached to a map and visualized, queried, analyzed, and modeled for the purposes of visualizing patterns, trends, and ideas. The term coined to describe this powerful system was geographic information system, or GIS. While this new technology was first used primarily in the environmental and resource management area, it is now used in over 1,000 different applications in every sector of federal, state, and local government; education; and industry. GIS has become the standard tool for the management of very large databases and spatial analysis and modeling. Federal, state, and local government and industry had, for the first time, the ability to electronically automate certain complex and critical decision processes. Today these systems are responsible for assisting organizations worldwide.

The next chapter in GIS technology has been evolving with the increasing power of desktop computers; more powerful, lower cost, UNIX workstations; and widespread computer industry developments in open, distributed systems supporting high-speed networking technology. Better software technology combined with better hardware technology has enabled GIS to spread to many more users in both the public and private sectors. According to Daratech and Dataquest the largest growing market segment for GIS is the business community. In 1993 this market grew 53 percent, and it is expected to continue this type of growth for the foreseeable future, which brings us to the focus of this paper: the use of GIS in business.

WHY BUSINESS GIS

The question is, "why?" How can this technology benefit business? What can they do with it and why all the fuss? I believe the issue has surfaced because of a change in the business climate. Anticipating global changes in the 1980s, businesses invested over one trillion dollars on information technology but saw only minimal payoff. What happened to that technology and why was there so little return for the money? As a result of the pressure to change, businesses in the 1990s have now been forced to reexamine how they spend their dollars. Businesses' call to action today is to maximize their significant infrastructure investments, increase profits in a very competitive marketplace, and reduce costs and waste. Business is now having to reexamine how to adapt existing technology to be more responsive to changing needs. Business Process Re-engineering (BPR) redefines how work gets done and breaks down old methods. BPR, through a multidepartmental approach, redirects information to those who need it the most in a way that gives them the most intelligence to accomplish their jobs. What has BPR meant to business?

- Optimize resources, which means downsizing and outsourcing.
- Focus more on the customer and less on hierarchy.
- Require departments, divisions, and units to work more as partners and teams.
- Leverage existing assets to higher performance.

The object of re-engineering is to reduce inefficiency, to save money, and find new opportunities to make money. The linking component of all business processes is data. The trillion dollar

investment of the 1980s in information technology gave organizations lots of data but few ways to link business processes and share that information. Data were frequently fragmented and located in places that make access by decision makers and analysts difficult. Providing a means allowing end users, managers, and technical staff a common ground to re-examine processes requires bringing data together in a form that a multidisciplinary team can use to make informed decisions.

Results of Change

- Mass Marketing-----> Micro Marketing
- New Customers-----> Long-Term Relationships
- Customer Lists----->Customer Databases-----> Customer Communication/Knowledge
- Data----->Information---->Linkages---->Leverage
- Aging of America
- Conspicuous Consumption-----> Price/Value

NEW TECHNOLOGY TO MEET THE NEEDS OF BUSINESS

To maintain your business position in today's marketplace, you need to continuously make careful, innovative decisions about new methods of business operations and consider new ways of thinking about how you do business.

Periodically, events or breakthroughs occur that revolutionize business operations. Many of these are technological in nature. These significant advancements in technology create turning points that change the way business is conducted. Examples from relatively recent history include the professional computer, word processing and spreadsheet software, electronic mail, and the fax machine. Successful, innovative businesses quickly and carefully evaluate the potential of these new technologies within their own business context and adopt those that have genuine utility and promise. Through successful use of such innovations, businesses experience improved efficiency, competitive advantage, and increased profits.

A technological innovation that is experiencing rapid acceptance in the general business community is geographic information systems (GISs) "on the desktop." Significant contributing factors to this include

- Computer telecommunications networks that enable desktop technologies to be linked to each other and to corporate information systems making them more integrated in the overall corporate technology infrastructure. This creates the *opportunity* for better data integration.
- Recognition that at least 80 percent of the data the business community manages has a geographic or locational component (e.g., loan records, customer addresses, warranty cards, sales figures by territory, company locations, and sales histories). However, much of these data exist in separate data management technologies and cannot easily be combined.
- Computer mapping or GIS can manage, *integrate*, and analyze that information much more effectively and efficiently through a geographic framework *on the desktop*.

These advances can provide today's business with a construct for a new business environment. That environment is one where the corporate information resource is available to anyone on the communications network using geography (in the form of an "electronic map") as the data organizing and integrating medium. This very simple, yet powerful, concept provides the opportunity for you to adapt your operations to new challenges in the market and within your organizations and to compete more effectively. Wherever there is competition for a limited prize,

GIS can create a competitive edge. Wherever assets or investments are geographically dispersed, GIS offers significant management capabilities.

GIS technology can range from simple map creation tools that allow you to visually compare and evaluate characteristics of a location (your customers, the distribution of sales, and consumer profiles by demographic characteristics, etc.) to complex analytical models that combine volumes of statistical information about an area.

GIS technology brings a comprehensive set of capabilities based on state-of-the-art computer technology to the locational, marketing, sales, and planning problems of business. These tools let you more effectively manage your assets. These tools can be the foundation for visualizing customer location, product, and service types by geographic areas and demographic characteristics, market penetration, and even assist with planning how to retain customers. They can be the tools for integrating information from multiple operating divisions to allow management new insight into an operation. The tools can provide a systematic means for managing the geography that affects the corporation's various assets; and these tools can be the analytic foundation for applying and evaluating new models for store operation or store location.

The vision is one where geography becomes the organizing framework for business operations. This is a significant statement that is not necessarily easy to completely understand or accept. It is, however, an ultimate goal to which business can subscribe. GIS then becomes part of the corporate infrastructure of the company. It is the technology that integrates disparate, related, relevant information, allows for the visualization of that information, and provides the "workbench" and basic tool set for complex scientific analysis in relation to location.

Geographic information systems let users organize and access virtually all digital information in their computing network using a map as a kind of visual *organizer*. Desktop mapping will change the way you think about and use geographic information. It will provide technical staff with analytical tools, but will also provide the nontechnical end users in other business units an advanced geographic visualization and query system for the desktop. Historically, data such as customer profiles and demographic data have been used only in tabular or spreadsheet formats or presented as business graphics. Linking these tabular databases with maps so that spatial analysis functions can be performed will result in a much clearer visualization of your business requirements such as customer profiling, marketing research, competitive analysis, target marketing, demographic analysis, and trade area analysis.

GISs are simply systems for creating linkages between your corporate and regional office databases and features displayed on maps. What this means is your company has the ability to link vast corporate databases to maps. The geographic references in corporate databases typically include customer addresses, sales by region or office, credit card information, sales patterns, and history. Using maps to link this information (along with available off-the-shelf census and business facts data) with other demographic and business information gives you a powerful new way to utilize that information, and to bring it within the purview of the visual mind's capacity for finding, analyzing, and interpreting patterns.

Business wants answers to locational questions on customer distribution, buying power, buying history, advertising efficiency, competitor distribution, payment history, and how to best target new customers.

To answer these and other kinds of questions requires information about the geographies involved. Geographies are natural building blocks for relating spatial data with many other types of business information—information that is a corporate asset that must be used, analyzed, and manipulated for the purpose of increasing market share in a very competitive marketplace. Using these data, along with new computing technologies such as GIS, new worlds of analysis are made possible relating

diverse business data from a geographic vantage. The information assets of any business can now be brought into a common framework to discover new patterns and relationships. The fundamental capability to integrate disparate information sources paves the way for many mapping applications in your business environment.

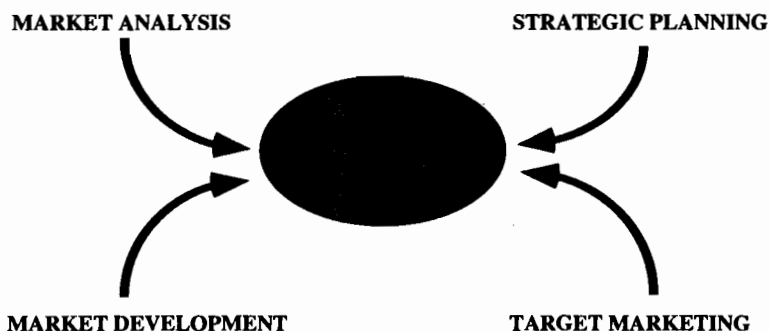
Let's examine more closely what this technology is and what it can do for you in your business.

WHAT IS GIS TECHNOLOGY?

Computer mapping software allows you to link features on a map (see next section of map feature types) with tabular databases and spreadsheets. GIS is formally defined as the retrieval, display, analysis, and management of geographic and geographically referenced data. The desired purpose and outcome of computer mapping is better and more effective decision making. People who use these systems feel that GIS can reduce obstacles to understanding large and complex data sets and thus facilitate better management of projects and ongoing operations. Users of GIS for business are finding they are gaining a competitive edge over others because they are analyzing geography smarter.

WHAT ARE THE BUSINESS ISSUES?

TODAY'S FOCUS



The primary issue facing business today is how to improve profits, remain competitive, and reduce risk. This can be accomplished by getting to know more about your customers, their buying habits, lifestyles, and disposable income. Related to this are the following issues:

- Market and Locational Analysis Issues
 - What markets and locations offer the best opportunity?
 - Where are my most profitable customers?
 - Which markets offer the greatest opportunity for my products and services?
 - What impact will new locations have on my existing locations?
 - Who are my best customers?
 - Where are they located?
 - How can I better understand them?

A GEOGRAPHICAL WAY OF LOOKING AT YOUR BUSINESS: MAPPING CONCEPTS

Desktop mapping and GISs are simply systems for creating linkages between databases and locational features displayed on maps.

What this means to you as a business is the ability to link the vast amounts of data in your corporate database to maps, which are available either in the public domain or commercially from a variety of vendors. The majority of corporate databases have geographically referenced information in them such as addresses, sales by regions, inventory counts at various stores, or records of other transactions that are referenced by location (such as ZIP Codes or even telephone numbers). About 80 to 85 percent of corporate business databases contain some sort of geographically referenced information. Linking this information with maps gives you a powerful new way to utilize this information, to bring it within the purview of the visual mind's capacity for finding, analyzing, and interpreting patterns. Data visualization is the fastest growth area in computing today, and desktop mapping is an important visualization technology and the one with the widest range of applications.

Mapping is the key. Using maps and your tabular data together is a powerful way of understanding and communicating your message and presentation. Desktop mapping enables you to geographically manage, analyze, and present your data, and view the geographic results of your analysis.

For example, by combining street addresses of your customers with socioeconomic and census data, you can create a profile of your customers by per capita income, age, lifestyle, and buying preferences to target neighborhoods meeting your criteria.

In order to understand how this is accomplished, we need to examine some of the basic concepts of desktop mapping.

BASIC MAP CONCEPTS

The following are the two basic types of map information:

- Spatial information—describes the location and shape of geographic features and their spatial relationships to other features.
- Descriptive information—information about the features.

A locational or spatial database contains several types of spatial features. These are points, lines, and polygons.

Points are exemplified by features such as the address of a customer, the location of a store or factory, an ATM machine, and so forth.

Lines may be individual streets, a highway network, a truck route that includes many streets, utility lines, or any other linear feature.

Polygons are enclosed areas that can represent sales regions, sales territories, cities, counties, ZIP Codes, or census tracts.

FEATURE ATTRIBUTES

Descriptive information in a desktop mapping system or GIS is called feature attributes. This is the tabular information contained in your company's computers, or tabular data available from commercial vendors as described later in this paper.

Point attributes—Point-related attributes are those that describe locational information such as addresses, competitor locations, sales by location, or locations of businesses by SIC code and addresses.

Line attributes—Linear features, such as street centerlines, have attributes such as left and right (side of the street) address ranges, street names, left and right ZIP Codes, and census tracts, as well as information about number of lanes and street types. This type of information allows you to address match your customer files to a street, to know the distance from a distribution center to a single store or a number of stores, or to optimally route delivery trucks to perform analysis such as accessibility modeling.

Polygon attributes—A census tract or sales territory is a good example. The data from commercial vendors or from your corporate database may contain information about market share by sales territory or sales volume by region or may include demographic information and household counts by a census tract or ZIP Code. There are no limits to the potential for storing data that can be linked to polygon map features.

In summary, a desktop mapping system does not store a map in any conventional sense, nor does it store a particular image or view of a geographic area. Instead, these systems store data from which you can create the desired view, drawn to suit a particular purpose.

TYPES OF QUESTIONS THAT MIGHT BE ASKED OF A GIS

Using the above types of analyses, a number of questions can be asked. These include the following:

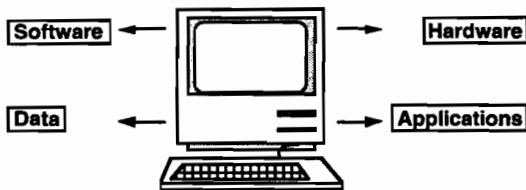
- **Locations**—What is at...?
- **Conditions**—Where is it...?
- **Trends**—What has changed since...?
- **Patterns**—Which data are related...?
- **Models**—What if...?
- **Why?**—Why do conditions/patterns exist the way they do?

DATA

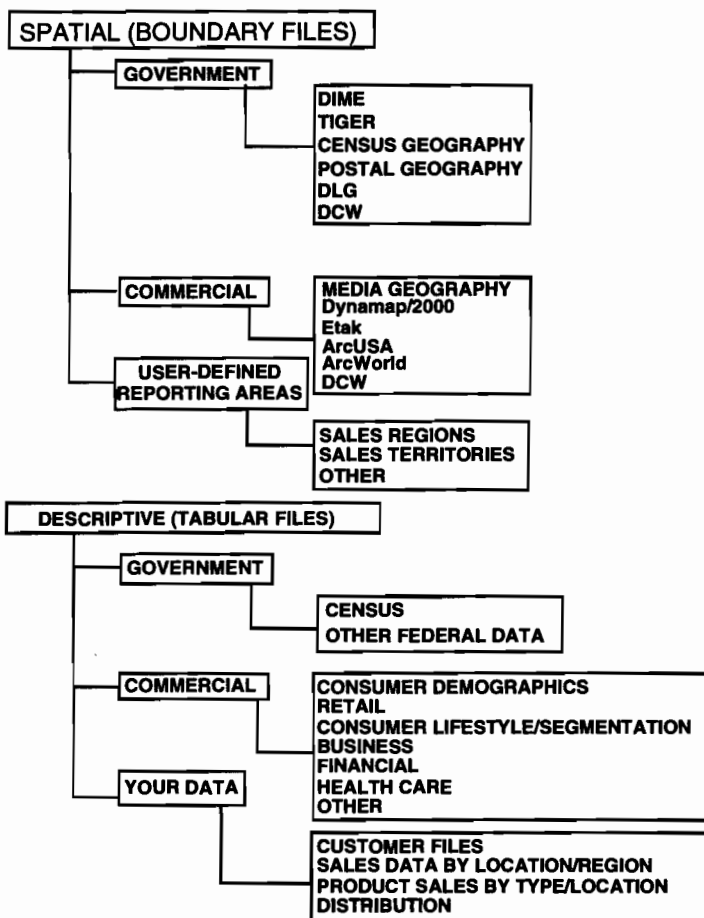
Applications cannot exist without data

Data are useless without applications

Construct for Desktop Mapping



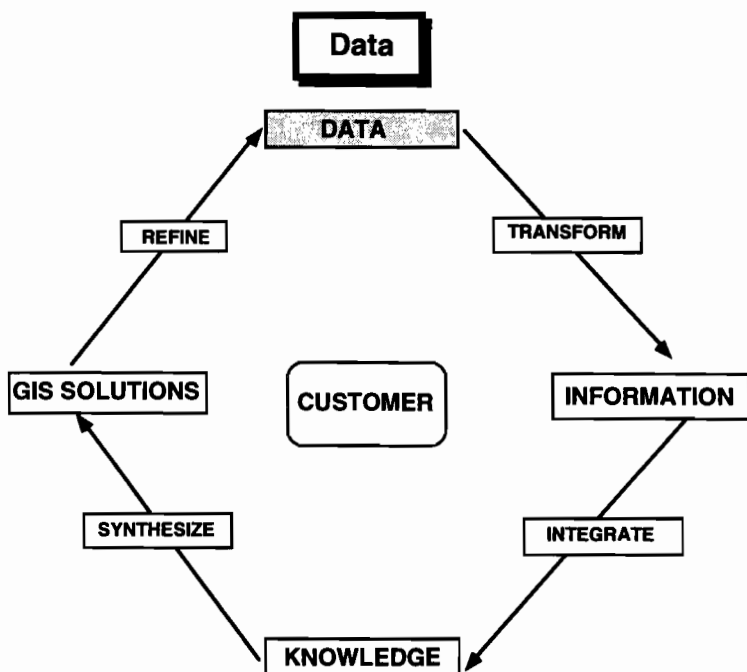
General Data Types and Classes



WHY SO MUCH EMPHASIS ON DATA?

- Data represent 75–90 percent of nonlabor costs of implementing GIS.
- The only limits to successful GIS use are creativity of user and available data.
- Gaining competitive advantage.

Construct for Attribute Data



Data are the building blocks for GIS applications. Data are defined as the raw facts or factual information, as measurements or statistics, that must be processed to be meaningful. Examples include such things as population, households by income, bank deposits by type, number of businesses by type, income, or addresses. Information is defined as knowledge obtained from investigation, or study, a quantitative measure of the content of data. Examples include creating business and residential lifestyle segmentation classifications. Knowledge is what you bring to the table. You know your business better than anyone else. This can include such things as what criteria you apply to the siting of a new store or knowing how many customers you need to make a store successful or knowing what the sales potential should be within a sales territory. Applying your knowledge to both data and information will let you proceed in developing needed GIS applications and solutions to business problems.

The level of detail you want (information at the census tract level or individual address at the street level) will determine the type of geographic boundary files you will need. The figure below identifies the number of typical households found in each level of geography.

Levels of Geographies

Geography	Total	Typical # of Households
States	51	1,862,745
MSAs	303	250,000
Counties	3,141	30,245
ZIP Codes	41,940	3,167
Census Tracts	61,257	1,551
Block Groups	226,399	420
Carrier Routes	556,821	171
Blocks	6,500,000	22
ZIP+4s	28,000,000	10

THE CONCEPT OF CLUSTERING

Clusters are the result of the classification of objects into similar groups, similar to the concept of segmentation models. The United States population is clustered into numerous distinct groups of homogeneous geographies sharing a common demographic characteristic.

Clusters provide a composite lifestyle interpretation of a wide variety of demographic information, and offer a standardized framework for evaluating consumer behavior for various products and services. Knowing in which cluster a consumer lives provides a reasonable means of understanding how that consumer will behave in the marketplace.

Lifestyle clusters are developed through the characteristics of the neighborhoods in which people live. The neighborhoods are small—block groups, census tracts, ZIP Codes, and even ZIP+4s—and homogeneous enough to enable marketers to draw reliable conclusions about the people residing there. The concept of neighborhood clustering is based on the theory that families living in neighborhoods of similar demographic characteristics tend to exhibit similar lifestyle and purchasing patterns.

With linkages to syndicated surveys and other marketing and media databases, such as Simmons Market Research Bureau, Nielsen Marketing Research, Nielsen Media Research, and others, marketers can define precisely which segments really comprise their best prospects and target their heaviest efforts at these segments.

GEOCODING

Geocoding is the process of matching a client's customer records to current census and postal geography. Geocoding assigns the appropriate Block Group, Census Tract, ZIP Code, County, and MSA to each record in the file. Any customer file containing a street address, city, ZIP Code and state can be geocoded. This is your link to geography. Examples of customer files include warranty cards and registrations, bank customers, sweepstakes and promotion respondents, survey respondents, new car buyers, magazine and newspaper subscribers, credit card holders, customer billing statements, or collecting a customer's ZIP Code at the point-of-sale, and so forth.

Two types of geocoding are commonly used. The first method involves using your customer address database, and the ZIP Code of each address in the file is matched against current ZIP Code boundary files. Addresses are assigned as clusters to each ZIP Code.

The second method is point geocoding, which is the assignment of coordinates (latitude and longitude) based on the street address, city, and state. Point geocoding is very useful for displaying customer and competitive locations within trading and service areas, conducting spatial analyses, determining primary and secondary trade areas, calculating distances from one location to another, planning transportation routes, displaying billboard locations, and more. The trend today is a move to point geocoding and the use of street centerline files (TIGER or commercial files such as Dynamap/2000 or Etak).

Geocoding, also known as address matching, is the process by which an address is linked to a physical location on earth. It is this essential function that allows businesses to translate tabular data (data found in spreadsheets and databases, such as customer addresses and demographic data) into spatial data (data that relate in space, that is, geographically) for any number of different analytical, marketing, sales, service, presentation, and planning applications.

THE UNIVERSAL TRANSLATOR

If you think of geocoding as a "language translator," the concept of what geocoding is might be more easily understood. Let's take an example. Suppose that, while on a business trip to Japan, you receive an invitation to a party in a small town located on the back roads somewhere between Yokahama and Tokyo. Deciding that it would be both enjoyable and profitable to attend, you consult a local street map to determine the best route. You speak little of the language and your knowledge of Japanese stops a little short of understanding kanji characters. You need a way to convert Japanese to English, and in a hurry. Without it, you will be unable to get where you want to go.

This scenario is analogous to the problem we confront when asking computers to locate an address on a computer's digital map. Electronic maps are built using a system that assigns latitude and longitude coordinates to every feature on the ground. A computer reads these "maps" by referencing the coordinate positions of each feature, such as a road or a ZIP Code.

Since the locations that businesses most often want to find have an address, and since electronic maps contain that address as a coordinate, geocoding becomes the logical way for an address in "postal" language to communicate with the coordinates in "computer" language. In other words, geocoding is simply the necessary—and only—method for identifying where on earth, in physical terms (such as Main Street, Elm Street, a particular bank, or a competitor), just about any physical feature with an address can be found.

THE BENEFITS OF GEOCODING

U.S. businesses currently spend over \$60 billion a year delivering people and products from one location to another, but logistics is not the only area where geocoding is important. This is because virtually every business data record contains a street address; in fact, address information makes up the majority of the data found in most business databases. And the ability to relate this address information to computerized maps is of vital importance.

GISs and desktop mapping systems allow users to associate attribute information (such as age, income, or buying patterns) to addresses, thereby gaining the unprecedented power to inspect a variety of different databases for spatial relationships—relationships that might go unnoticed and unexploited were it not for such computerized maps.

THE BASIC ELEMENTS OF GEOCODING

The components of the geocoding process can be reduced to three basic elements, each of equal importance.

1. A database of addresses or locations: Supplied by the users, the addresses or locations list contains customer, prospect, site, and other address information in a consistent and well-organized format with few, if any, spelling errors and little missing information.
2. An accurate computer map: This map can be one of a variety of commercially available digital maps. However, the quality of the map always affects the success of the geocoding process. The better the map, the better the match rate. Computer maps with incomplete or out-of-date information do not provide the opportunity to match many records.
3. Geocoding software: This is the functional portion of the process, the part doing the actual "translation."

COMMERCIAL DATABASES

There are numerous companies that provide databases that can be used in conjunction with geographic boundary files and demographic data available from the Census Bureau. In general, the following categories of databases are available. It is important to note that these databases can be incorporated and used with your own corporate databases.

CONSUMER DEMOGRAPHICS

- 1980/1970 Census
- 1990 Census
- Population Estimates and Projections
- Extended Income Ranges—Income by Age of Head of Household

Consumer Lifestyle/Segmentation

- Affluence/Net Worth Categories
- Lifestyle Clusters
- Hispanic Segmentation Clusters
- Product Potential
- Financial Clusters

RETAIL

- Convenience Stores
- Drug Stores
- Grocery Stores
- Market Potential Retail Expenditures
- Mass Merchandisers
- Shopping Center Locations

Business

- Business Locations
- Business Statistics
- Mailing Labels
- Prospect Lists

Financial

- Financial Institution Deposit Data
- Financial Institution Locations
- Financial Institution Competitor Data
- Product Potential

Other

- Automobile Registration Counts
- Consumer Lists and Mailing Labels
- Crime Vulnerability
- Residential Household Counts
- Residential Household Locations
- NRB Shopping Center Database

Health Care

- Ambulatory Demand Forecasts
- Diagnosis-Related Groups Forecasts
- ICD-9 Surgical Procedures
- Health Care Facilities
- Health Care Usage Profiles
- Hospital Utilization File
- Major Diagnostic Categories Forecasts
- Physician Office Locations
- Physician Summary Counts
- Physician Lists and Mailing Labels

APPLICATIONS

Now that you understand the basic concepts of GIS and what data are available, what types of things can you do to solve your business problems? The following discussion presents a brief summary of the types of applications you can use for your organization.

Within the business community, the proliferation of commercial, off-the-shelf data continues to develop at a rapid pace. The ability to analyze, visualize, and query these data and transform them into usable information for decision making is at the top of every corporation's agenda.

Environmental Systems Research Institute, Inc. (ESRI), has built and developed software that significantly helps people make better decisions. Our company philosophy has been, and continues to be, to provide the best possible software products to help people make better decisions and to make them successful in that effort. Other vendors also provide GIS software, ranging from just desktop versions to high-end systems.

ESRI brings this powerful technology of GIS and desktop mapping—the ability to link maps with both commercial data and your data on workstations, personal computers, or mainframes—into the mainstream of corporate America.

Most corporate databases contain critical location-based information such as customer addresses, sales regions, store locations, and profiles of demographic populations, and yet most businesses do not view their information geographically. GIS software allows all types of businesses to view their corporate databases, as well as off-the-shelf commercial data, quickly, cost-effectively, and easily.

- GIS for desktop mapping is a tool that enables dynamic interpretation of geographic data.
- GIS for desktop mapping is a management and decision support tool.
- Desktop mapping has become a key component for decision support systems in many organizations.
- Innovative and advanced software design combined with desktop technology has made it possible to create reports, analyze data, query it, display it, and publish it in ways that are impossible with business graphics or spreadsheets.

SOME EXAMPLES OF WHAT YOU CAN DO WITH DIFFERENT DATABASES

CENSUS, DEMOGRAPHIC, AND SOCIOECONOMIC DATA

- Identify demographic and economic trends and base your business decisions on future potential.
- Define market areas based on their population, race, age, and income characteristics.
- Track rapidly growing markets.
- Compare the demographic makeup of one market to another and select the best site for your business.
- Analyze the most current demographic variables and make the best possible business decisions.
- Evaluate potential sites for your business based on the occupations of the people who reside near the locations.
- Determine the property values of the homes in your market area.
- Identify the percent of households with children and tailor your marketing program to this segment of the population.
- Evaluate population by race.
- Tailor your marketing programs to the people who live in your trade area.
- Analyze single versus married population within your trade area.

BUSINESS DATABASES

- Select and rank geographic locations based on the number of competitors in your trade area.
- Analyze your business-to-business sales potential.
- Determine daytime market potential by examining the number of employees, by industry, who work in the area.

FINANCIAL AND BANKING DATABASES

- Analyze the consumer demand for specific financial products, such as annuities, auto loans, and home equity lines of credit.
- Determine whether or not the population in your trade area meets your specific criteria.
- Identify the exact locations and influence of the competitive financial institutions near your branch locations.
- Evaluate the bank and savings and loan deposits in your trade area to determine your market share.

INSURANCE DATABASES

- Analyze the consumer demand for a variety of insurance and investment products, such as universal life, stock funds, liability insurance.
- Identify geographic areas that meet or exceed your population and median income criteria.
- Identify the number of bank and savings and loan branches in your trade area.
- Evaluate your potential based on the number of investment services and security brokerage and trust management businesses in your area.
- Meet the Federal CRA regulations.
- Evaluate your market potential based on the consumer demand for specific products and services.
- Plan the most profitable acquisition strategy possible.
- Determine the number of customer households for financial products, such as demand accounts, first mortgages, bank credit cards, and certificates of deposit.

- Analyze your institution's market penetration versus potential.
- Identify new opportunities and allocate your marketing resources accordingly.

RETAIL DATABASES

- Estimate the consumer retail expenditure for major retail categories.
- Compare the consumer demand for specific retail categories to the business supply of these same categories.
- Present objective information to potential commercial property buyers, developers, or retail tenants.
- Identify retail concepts offering the greatest potential for a specific trade area.
- Determine the best tenant mix for a strip center and shopping mall.
- Analyze the number of competitors in a market by business type.
- Identify and rank the potential of shopping centers, over 100,000 square feet, in your trade area.

REAL ESTATE DATABASES

- Analyze and compare the 1990, current-year, and five-year projected population figures to identify areas with potential growth.
- Identify areas where property values meet the needs of potential home buyers.
- Focus your marketing efforts on areas that meet your median income criteria.
- Help sell your residential properties by providing prospective buyers with information such as the number of elementary and secondary schools, churches, hospitals, and so on within a specific radius of a home.

UTILITY DATABASES

- Determine the ownership and distribution of a wide array of energy-consuming electronics and appliances, such as dryers, electric ovens, or personal computers.
- Anticipate future energy demands using projected population and household growth information.
- Tailor your marketing programs to the utility needs of the consumers in your service area.

TELECOMMUNICATIONS DATABASES

- Determine national standards on different customer satisfaction measures such as quality, service, and value.
- Compare your company's customer satisfaction performance to that of other telecommunication companies.
- Identify the types of telephone services utilized by consumers in your market area.
- Analyze population and household trends to help improve your business decisions.

CABLE TV DATABASES

- Measure the viewership and subscribership of various types of cable services matched against demographic information.
- Target your cable programs in areas that meet your income and age criteria.
- Identify areas with high concentrations of children so you can promote your Disney Channel.
- Select the areas in your service territory with the greatest cross-selling potential.

AUTOMOTIVE DATABASES

- Measure the ownership and usage of various types of automobiles.
- Identify areas with high concentrations of domestic versus imported cars.
- Determine the percent of households in your market that have two or more vehicles.
- Compare the number of households that tend to purchase used versus new cars.
- Analyze your market potential based on the types of professionals who work in an area (i.e., sales and management, construction, clerical).

Retail/Packaged Goods

- Promotion planning
- Store formatting
- Merchandise mixing
- Space management
- Competitive analysis
- Customer acquisition/retention
- Advertising
- Siting new stores
- Target marketing

Financial Service Applications

- Product cross selling
- Private banking
- Branch analysis/planning
- Regulatory compliance
- Property appraisal
- Cash management
- Portfolio acquisition
- ATM placement
- Bank/ATM security
- Competitive analysis
- Market share analysis
- Demographic analysis

Real Estate Applications

- Acquisition/Disposition
- Site analysis
- Appraisal/Risk assessment
- Tenant mix analysis
- Demographic analysis
- Shopping/Traffic pattern analysis
- Market analysis/Feasibility
- Community/Neighborhood profiles
- Competitive analysis
- Demand potential

Health Care Applications

- Physician referral
- Senior services
- Emergency response
- Caseload analysis
- Service area expansion
- Site location of new clinics
- Demographic analysis
- Competitive analysis

Manufacturing Applications

- Sales territory alignment
- Dealer development programs
- Product development
- Sales prospecting
- Lead generation
- Media allocations
- Dealer referral system
- Measure sales performance
- Demographic analysis
- Competitive analysis
- Market share analysis
- Demand potential

Utilities/Telco's Applications

- Cable programming
- Subscriber analysis
- Market share analysis
- Demand potential
- Demographic analysis
- Load forecasting
- Disaster planning/recovery
- Conservation programs
- Economic development
- PUC compliance
- Yellow Page ad sales
- Forecasting future growth

BRINGING IT ALL TOGETHER: HOW GIS CAN HELP YOU

SELLING

In selling, you go to your customers, most often with a sales force.

GIS can improve the productivity of sales forces and service operations by reducing windshield time, balancing workloads, and providing better information.

SUCCESSFUL SELLING REQUIRES

- **Improved Customer Knowledge**—GIS can help reinforce your customer knowledge, helps you to calibrate competition, develop sales strategies and forecasts, and strengthen niche marketing opportunities.
- **Effective Sales Territory Planning**—GIS can help evaluate sales force effectiveness, balance workloads, and redistrict territories to reduce costs and increase productivity.
- **Accurate Brand Management**—GIS can relate product mix to demand to build market share and customer loyalty.
- **Cost-Effective Communication Focus**—GIS can be used to focus promotions for increased sales lift, measurable results, and reduced advertising waste.

RETAILING

In retailing, your customers come to you.

- **Distance, competition, price, quality, comfort, convenience, and range**—as well as demographics, expenditure, and clusters—all come together to determine how profitable your business will be.
- GIS can help increase the value of your company by commanding all aspects of the retail marketing mix.

SUCCESSFUL MARKETING REQUIRES

- **Better Retail Intelligence**—GIS can help you understand your brands and merchandise purchasing drivers, as well as your competition at the trade area level.
- **Better Location Decisions**—GIS is the marketing tool for evaluating performance, estimating demand by merchandise line, developing strategy, and reducing risk.
- **Merchandising Decisions**—GIS is used to develop store promotions and build brand equity according to consumer demand. It can help establish accurate measures for merchandise demand, pricing, quality, comfort, convenience, and range.
- **Improved Use of Advertising and Promotion Resources**—GIS's functions can help to focus advertising and promotions to build better trade partnering alliances.

THE VISION OF AN ENTERPRISE BUSINESS GIS

THE DISTRIBUTED COMPUTING MODEL

Today, a clear alternative to mainframe computing has emerged. Across the computer industry desktop workstations, maturing network technologies and standards-based, nonproprietary server computers offer another way to meet the needs of enterprise computing. These open systems are defined by the shared nonproprietary interfaces that allow computing products from competing vendors to work together. Open systems tend to be distributed solutions because interoperability among computers makes using multiple computers together possible. Cost-effective systems interoperate on special-purpose, less expensive computers across a shared standards-based interface. Open system architectures provide attractive price/performance. More and more frequently, MIS executives select a distributed processing solution for new, existing, and mission-critical applications. The costs of acquiring, upgrading, and maintaining mainframe computers is often measured in the millions of dollars. Today, new technology is available that can dramatically reduce these costs and provide faster and more efficient computing.

GIS TECHNOLOGY AND DISTRIBUTED SYSTEMS

GIS technology is now being used in enterprise data processing infrastructures. GIS is a broadly applicable technology found in many computing architectures. ESRI has been challenged to provide a GIS solution that works in these varied environments. In response, ESRI's enterprise computing strategy exploits the capabilities of open, distributed systems while preserving user investment in data center technology.

GIS technology has special characteristics that favor implementation in a distributed, rather than mainframe single processor, computing environment. These include:

- The need to support both graphic and tabular processing. While mainframes are very good at handling the relatively small data packets associated with block mode terminals, they are less well prepared to handle the greater demand for terminal I/O required by graphics applications, particularly imaging applications. And since GIS does both graphics and tabular processing at the same time, it is impossible to tune a single machine to optimally handle both applications.
- The need to support bitmapped window-oriented user interface software. The specialized terminals and software required to support this style of user interaction require dedicated processing power. A mainframe that must support several other applications and possibly hundreds of users usually does not have the resources available.

- The need to support a wide variety of graphic input and output devices. GIS applications need access to devices that may be difficult to integrate in a mainframe configuration. These devices can include digitizers, electrostatic plotters, laser printers, and scanners. The computer industry is producing a remarkable variety of choices in this area. You may want to pick a particular device or have the freedom to choose among a selection of like devices based on factors such as price or service.
- GIS technology is in the mainstream of workstation development. GIS software makes extensive use of workstation processing power to drive graphic display and window-oriented user interfaces. Distributed GIS makes full use of the network capability built into workstations. GIS benefits from the developments that appear first in the workstation environment.

The open systems approach means the integration of industry-leading hardware and software products into a customized solution that is ideally suited to GIS applications. Today, single vendor solutions based on proprietary technology and not industry standards are a gamble—technology trends are based on industry standards, not proprietary solutions. In the GIS world, industry-standard equipment is critical. This is because each user site has tailored its computer system to its particular needs, combining personal computers, workstations, minicomputers, mainframes, and a wide variety of peripherals. Without industry standards, it is often difficult and sometimes impossible to tie these diverse elements into a cohesive, cost-effective computer network. Proprietary systems, which do not adhere to industry standards, are problematic. For example, it may not be possible to run a certain kind of software or link a particular type of computer, plotter, or printer to such systems. An open system adhering to standards has far fewer limitations; users maintain maximum flexibility in configuring their computer environment.

Industry standards move the power of choice from the computer manufacturer to the end user. Technology users, choosing among vendor offerings, can direct the evolution of the industry by promoting open competition. To compete, hardware and software vendors must supply equipment with excellent price/performance and interoperability.

Open systems based on shared standards result in lower costs than proprietary systems. And an open system can include the hardware, networks, and databases that you already use. The open systems philosophy can integrate the systems you already have by implementing products on them that conform to standards. For example, distributed applications can use network gateway products to translate from one protocol to another (e.g., between TCP/IP and APPC). Adding new equipment to a proprietary system can also be expensive. Even though a prospective new system may be more powerful than the existing system, the cost to implement it may not be worthwhile if the older, but still useful, equipment is made obsolete. When that happens, the original investment in the equipment is lost and users must be retrained on the new system. The overall costs of such a move makes many users think twice before moving to a proprietary system.

WHY ENTERPRISE GIS

- **Centralized Vision, Decentralized Work**
 - Corporations have decentralized the decision process but still need vision.
 - Focus more on the customer less on hierarchy.
- **Fast Response to Change**
 - Legislative change worldwide complicates decisions and requires ways to assure and assess compliance.
 - Finding new opportunities by spotting trends and patterns is time critical.
 - Global markets present an enormous information management problem.

■ **Optimizing Investments**

- Closer hard and soft asset management has become more important.
- Optimizing resources means downsizing and outsourcing.
- Need to leverage existing information assets to higher performance.

BENEFITS OF ENTERPRISE GIS

REALIZE A RETURN ON THE 1980S INFORMATION TECHNOLOGY INVESTMENT.

■ **Make Business Resources More Productive**

Waste is due to inefficient processes. Systems and processes waste money when they are out of control or based upon incorrect conclusions. Enterprise Business GIS enables you to analyze and visually display enterprisewide data sources within a single framework.

■ **Develop Consistent Standards and Eliminate Redundant Data**

Inconsistent standards and redundant information within an organization are the norm. Business GIS acts as a filter to visually weed out repeats and to quickly expose inconsistencies and encourage discipline.

■ **Aggregate Large Data Sets to See Trends and Patterns**

More accurate decisions require a large sampling of data. Enterprise Business GIS can aggregate data in a variety of ways: by stringing together points statistically to create meaningful networks or encompassing statistically related information in meaningful regions.

PROMOTE THE DEVELOPMENT OF ORGANIZATIONAL INTELLIGENCE.

- Integrates the division, unit, or department to construct a new responsive structure.
- Leverage information by connecting employees and functions.
- Displays analysis results to optimize knowledge.
- Requires departments, divisions, and units to share information as partners.

ORGANIZATIONAL INTELLIGENCE WILL PRODUCE RESULTS

- Continuing *improvement*—of a process, product or service
- *Exploitation*—exploiting existing knowledge into new and different products, processes, and services.
- Genuine *innovation*—finding new opportunity from trends and patterns.

REALITY OF ENTERPRISE GIS

Even though GIS technology has been available, it has not been an obvious choice in business system integration. It has been a case of "almost there" technology. To be able to fully exploit the power of Business GIS there are fundamental technological thresholds to make corporatwide integration possible. Primarily there are four reasons why Business GIS is now becoming a real option:

- Computer communications network technology is enabling better links between corporate information systems.
- Technological advances in desktop computer power have made them better able to accomplish the tasks necessary for Business GIS data analysis.
- ESRI's hardware-independent suite of software technology has allowed desktop GIS systems to seamlessly share the resources between departments and divisions.
- The wider availability and lowering cost of geographically based data is now an option available to most business applications.

HOW TO GET STARTED

IMPLEMENTATION PLANNING

Based on twenty-five years of customer experience, ESRI has developed a formal methodology for the planning, design, and implementation of Business GIS. It is presented here for you to use in planning, and to assist you in identifying a GIS vendor who can help you through all the necessary steps of implementation.

- Initial strategic system planning and design
- Orientation seminars
- Needs assessment
- Requirements and cost/benefit analyses
- Database design
- Application definition
- Hardware configuration and integration
- Implementation and cost-recovery strategies

DATABASE DESIGN

GIS is all about converting data into useful information and making your application work.

- Preautomation services
- Assembly and integration of existing data
- Scanning and digitizing automation
- Geocoding
- Database access design
- Reformatting

APPLICATION DEVELOPMENT

- Application assessment
- Application process design
- Consultant/User design assistance
- Development code design
- Cross-platform application design and implementation

OPERATION SUPPORT

- Standard training courses
- Customized training
- On-site training
- On-site technical services

NEXT STEPS

COMMUNICATING RESULTS

The biggest single outcome of Business GIS functionality is the ability to produce a map that communicates important information in a way not possible by looking at tables and columns. The benefit is to enable you to *communicate* this visual information to achieve changes and improve organizational strategy. A system that does not enable communication is failed technology.

Learning organizations that successfully implement GIS initially begin to see future functionality soon after they have mastered the initial application. As an organization grows, competitive pressure increases and opportunities arise, the planned system with the greatest growth potential and flexibility will painlessly accommodate these changes. Technologically limited systems will hit many obstacles, some of them insurmountable.

FUNDAMENTAL ASSUMPTIONS

- Obtain the understanding of top management. Business GIS is a long-term venture and needs vision.
- There should be a team/management structure in place to plan the system.
- To make Business GIS productive requires a clear focus for the long term. Full benefit will evolve over time.
- Realizing long-term productivity requires a constant stream of short-term results. Plan for intermediate applications.
- Successful conversion and implementation requires participation of many players. Establish communication links. Business GIS usually requires partnerships with data providers, other departments, and consultants.

Computer technology is making it easier to compile and store information. Databases and spreadsheets abound—everyone tries to organize their data in some way and analyze them and present them in an informative and impressive fashion. How can you make your own data easy to understand, quick to comprehend, and visually understandable? **A Picture is Worth a Thousand Words.** Data visualization is one of the hottest growth areas in computing, and desktop mapping is an important visualization technology.

Mapping is the key—using maps and your data is a powerful way of understanding and communicating your message. Desktop mapping enables you to graphically manage, analyze, and present your data, and view the geographic results of your analysis.

HP-UX to PC Porting Tools and Techniques

Arthur F. Walasek

Palo Alto Information Technology Center

Hewlett-Packard Company

3000 Hanover Street Palo Alto, CA 94304

Phone: (415) 857-3693 Fax: (415) 857-5518

Introduction

The HP9000 and PC are two types of machines commonly used in many organizations today to solve complicated business problems. Because of the diversity of business needs, users may not have access to both an HP9000 and a PC. For this reason, it is very desirable to have the same program working on both the HP9000 and PC platforms so that users may have a common interface to solve their business problems. To make application development and maintenance feasible for these multi-platform programs, it is necessary to have one set of source code work for both of these environments. A portable programming language such as C is supposed to provide the necessary portability between these platforms, however, due to environment differences, so called "portable" programs are not that portable between HP-UX machines and PCs. This paper will point out major differences between the HP-UX, PC DOS, and PC Windows C programming environments; problem areas when creating portable applications; techniques that can be used to port applications between these environments; and suggestions that development teams can use to make C programs seamlessly compatible between HP-UX, PC DOS, and PC Windows programming environments.

The Project

This paper is based primarily on a one and a half year project to port a UNIX client module to the PC Windows 3.1 environment. About 200 C source files, 70 C header files, and an additional 100 C source files used for regression testing were ported from the HP-UX platform to the PC Windows 3.1 platform. The HP-UX client version of this program was written to run in character mode on HP-UX workstations and servers, and the version on the PC was to be run in the Windows 3.1 environment. Even though this project did not consist of an enormous number of files, the project did hit a lot of major areas: C programming, network programming, database programming, using Lex and Yacc, and Windows programming on the PC. The project was broken up into 2 stages that seemed logical at the beginning of the project. The first stage was to port from the HP-UX platform to the PC DOS environment, and then from the PC DOS environment to the PC Windows 3.1 environment.

What tools were used to accomplish the task

Due to the great number of different areas that this project touched, a lot of tools on the PC had to be created to help accomplish this task. As well as creating some specialized tools, C compilers, Lex and Yacc parsers and compilers, editors, and other tools had to be used. The following is a partial list of the tools used to help accomplish this task:

- ◆ Microsoft Visual C/C++ compiler version 1.0
- ◆ MKS LEX and Yacc Toolkit
- ◆ MKS Toolkit (for VI, Strings, Grep, Awk, Sed)

File naming conventions

After preparing myself mentally for the task at hand, the first order of business was to move all the files from the HP-UX machine to the PC. Unfortunately, the original developers of the product never had any

idea that this code might be moved to another platform. The source code filenames were case sensitive, greater than 8 characters long, and sometimes ending with a suffix greater than 3 characters long - all of which the PC could not handle very well. A script was created on HP-UX that would convert the names from the old HP-UX format to the PC format and vice versa. Files that had *#includes* with long header file names had to be changed as well.

Recommendations

When creating new applications follow these simple rules for file naming so as to help avoid problems when porting between platforms:

- ♦ Use up to 8 characters for the filename (without extension)
- ♦ Use up to 3 characters for the filename extension
- ♦ Try to use case insensitive filenames (you may still need to rename files when going from the PC to HP-UX)

C datatypes

After correcting the filename problems, I now could turn my attention to the really fun part - trying to figure out how this code works, and what problems I might encounter when trying to compile this on the PC. The first thing that I found was that the *int* C datatype was used in a lot of places. This turned out to be a bit of a nightmare.

Using the int datatype

All C programmers are used to implementing the *int* datatype in a lot of places in their code. It is often used for counters, loops, sizes, array elements, etc. The whole problem with *ints* is that they are machine dependent in size. It just so happens that on HP-UX systems *ints* are 4 bytes long (the same as the *long* C datatype), but on the PC (in the DOS or Windows 3.1 programming environments) *ints* are 2 bytes long (the same as the *short* C datatype). The main problem, in this

project's source code, was that the *int* datatype was assumed to be 4 bytes long by definition and that it was interchangeable with *long* C datatypes. In the code a pointer to an *int* would be passed to a function that required a pointer to a *long*. On an HP-UX machine this would not cause any problems because an *int* and *long* are defined for the same amount of storage, but in the PC DOS and Windows 3.1 environments this caused major problems. I knew the *ints* had to go, at least as many of them as I could get rid of. At first I thought of using the *short* and *long* datatypes in C instead of *ints*. The *short* datatype is 2 bytes on both the HP-UX and PC platforms and the *long* datatype is 4 bytes on both the HP-UX and PC platforms - perfect, I thought. But then it hit me. Some poor soul down the road may have to port this to a machine where a *short* is not 2 bytes or a *long* is not 4 bytes. This is where user defined datatypes became very important.

User defined datatypes

From Kernighan&Ritchie, The C Programming Language, pg. 34,; "...each compiler is free to interpret *short* and *long* as appropriate for its own hardware. About all you should count on is that *short* is no longer than *long*." This mandated the need to make up some user defined datatypes. There were 4 major user defined types that came up and were used to replace all *shorts*, *longs*, and *ints*. These 4 types were:

- ♦ *int16* - a 16 bit signed integer
- ♦ *uint16* - a 16 bit unsigned integer
- ♦ *int32* - a 32 bit signed integer
- ♦ *uint32* - a 32 bit unsigned integer

By using these 4 user defined datatypes instead of *ints*, *longs*, and *shorts*, a programmer can be fairly confident that the program can be easily ported to different hardware platforms where *shorts*, *longs*, or *ints* are of different sizes.

Recommendations

It has become painfully obvious to me that implementing user-defined datatypes can make a program easily portable to various hardware

platforms. I am not saying to completely quit using *ints*, *shorts*, and *longs*. Perhaps the best places for user-defined datatypes are arguments to user functions and procedures and for calling user functions and procedures. An example might be something like:

```
int32 myfunc(int16 arg1, int32 arg2, uint16 arg3, ...)
```

Memory Issues

After changing most of the code to reflect the new user defined datatypes, I thought I was getting pretty close to finishing this project. After clearing up all the little bugs that I left behind and creating a few functions for NLS message catalogs, I tried to compile the code. I got a message that a HP-UX user would probably never see: *Segment size exceeds 64K - error*. It turned out that the data structures and the static data in the code were enormous.

Available memory

There is a lot less available memory on the PC than there is on an HP-UX workstation. Large static arrays and data structures use a lot of segment memory storage - memory that is extremely valuable in the PC environment, but not so important in the HP-UX environment. There is also the notion of Memory Models in the PC environment that does not exist in the HP-UX environment. Along with this come the ideas of far and near pointers to data. Far pointers are basically pointers that contain a segment and an offset into that segment for data access, while a near pointer only contains an offset into the current segment for data access. To get around the far/near pointers in the PC environment I decided to compile in the Large Memory Model where all pointers to data would be considered as far pointers to data (containing both a segment and an offset).

Allocating memory

Another memory problem surfaced at runtime. The *malloc()* call is defined as: *void *malloc(size_t size)* where *size* is the number of bytes that you want to allocate. The problem is that *size_t* is defined both on HP-UX and the PC as *unsigned int*. Our *int* problem reared its ugly head again. On HP-UX a user can *malloc()* up to $2^{32} - 1$ bytes, but on a PC all that a user can *malloc()* would be $2^{16} - 1$ bytes. Since parts of the program were trying to allocate more than the 65535 bytes of data that the PC can handle, I had to create a new *malloc()* call that could take a 32 bit size, and create my own memory managing routines.

Recommendations

The following lists recommendations for dealing with memory for cross platform development:

- ♦ Use the Large Memory Model on the PC for far pointers to data (this can cause problems when coding for DLL's).
- ♦ If possible, do not declare large static arrays or data structures - allocate the data dynamically. If you must declare large static arrays or data structures, do not waste memory by making the bounds of the arrays or data structures unnecessarily large.
- ♦ Create your own memory allocation routine that can accept the largest size based on your requirements. If you do not want to code your own memory management routines, you can return an error if you attempt to allocate more memory than the software allows.

What about networking

At this point the code could be compiled and linked, and a client program was created in the PC DOS environment (a major step in my opinion). At last, I could try to run the client program. I ran the client, but could not connect successfully (remember that this was a client/server application and that I was porting the client piece of this

software to the PC DOS environment). This led in to the next series of problems - networking.

Sending and receiving data between different platforms

The first issue in the networking problem arena took us back to our *int* problem of long ago. This application used Berkeley Socket calls to communicate between client and server. Luckily, Berkeley Socket calls are supported on both the PC and HP-UX environments. The problem is that the *send()* and *recv()* Berkeley Socket calls use an *int* parameter for the length of the data to be sent or received. Therefore, on the HP-UX platform, $2^{32} - 1$ bytes of data could be sent or received with a single *send()* or *recv()* call, while only $2^{16} - 1$ bytes of data could be sent or received on the PC. This application had no protocol set up to say that multiple chunks of data had to be sent (i.e. sending two 64K chunks of data for one request). A new protocol had to be developed to accomplish this. The number of chunks of data and the total length of the data had to be sent first, and then each chunk would contain its length and its data. Placing the chunk order in the data buffer was not important since we were using TCP/IP (in other network protocols it might be relevant).

Different byte ordering schemes

The next problem encountered was that numbers (*integers, floats, shorts, longs, doubles, etc.*) were not being correctly passed back and forth between the PC and HP-UX environments. The application was directly copying from the memory of the computer to a data buffer that was to be used to send data back and forth between the PC and HP-UX platforms. The PC and HP-UX systems have a completely different byte ordering scheme for all number datatypes. For example, the PC would take the *short* number 7 (which internally on the PC is 0x0700), stick it directly in the buffer, transfer it to the HP-UX machine, and the HP-UX machine would stick it directly in a *short* number. Since the byte order is completely opposite, the number 7 on the PC got transferred to the HP-UX machine as the number 112. Obviously, this is not what we wanted to have happen. I had to decide

on a standard format to transfer all numeric data between multiple platforms. The decision was to use the Network Byte Order developed by Berkeley. The Berkeley Sockets interface came with calls to convert *shorts*, and *longs* into and from the Network Byte Order, but not with calls to convert floating point or double precision numbers to and from Network Byte Order. I created these additional calls, and any time a number was to be put or taken from a data buffer used for network communication, these calls would be used to ensure data uniformity.

ASCII vs EBCDIC

Even though we are talking about 2 ASCII environments, it is also worthy to mention that if this code was to run on an IBM mainframe or any other platform that uses EBCDIC for character representation, a standard format for character data would be required. An example would be to use ASCII data, and anytime an EBCDIC machine would be involved, it would be required to convert to and from EBCDIC any time character data was taken or put into a data buffer used for network communication.

Recommendations

The following are recommendations for dealing with client/server communication with multiple platforms:

- ♦ Use the concept of sending the number of chunks and the total data length for transferring large blocks of data
- ♦ Decide on a Network Data standard for numeric data and always convert data to and from that format when moving data to and from a network buffer
- ♦ Decide on a Network Data standard for character data (I would always use ASCII unless dealing with a lot of EBCDIC based machines) and always convert data to and from that format when moving data to and from a network buffer

Compiler and Runtime Issues

At last, the client program seemed to work and I could start my huge suite of regression tests. After doing some testing, I noticed some interesting behavior differences between the PC environment and the HP-UX environment. These differences were not necessarily errors, but did play a major role in the redesign of certain key modules.

Maximum values for long/short may be different

To my surprise, there are different values between HP-UX and the PC for the maximum positive numbers for the different number of bytes that can be in a number (1,2,4,8). For example, on HP-UX the maximum positive short is 32768 where on the PC it is 32767 (this is for the Microsoft Visual C++ version 1.0 compiler). I decided to do a little investigating, and I found the ANSI standard pertaining to maximum positive and maximum negative numbers. To say the least, the standard is a little vague. From my understanding, the standard states that the maximum positive number must be at least as great in absolute value as the maximum negative number for that number of bytes. That means, for a 2 byte number, where -32767 is the maximum negative number, the maximum positive number could be 32767 or 32768. Both HP-UX and the PC environments are following the standard. You may think that this is a mute point, but the Microsoft C compiler would refuse to read 32768 into a *short* value when using functions like *scanf()*, *sscanf()*, *atoi()*, etc.

Floating point numbers - rounding/displaying

Another interesting difference came up when trying to use floating point numbers. The way that floating point numbers are rounded is different between the HP-UX environment and the PC environment. The same floating point number might be displayed differently based on which platform you were on and what precision you want the number in. Another difference was the physical way a floating point number with an exponent was displayed. On the PC, the exponent always has 3 digits and a sign (i.e. 3.43e+012 9.45e-002). On the

HP-UX machine, however, the exponent may have 2 or 3 digits and a sign (i.e. 3.43e+012 9.45e-02).

Recommendations

The problems caused by these differences were not in the physical operation of the program, but in comparing known results against test results for regression testing purposes. I know that different PC compilers follow different rules for floating point rounding and display, and for the maximum values for numbers. Just be aware that these differences might exist, and are not necessarily a sign of a problem with your application.

Coding for Windows based applications

After finishing the regression testing, and making sure everything passed (this did take quite a lot of time), it was time to convert the code from the PC DOS environment to the PC Windows 3.1 environment. This involved creating a PC Windows 3.1 version of the client application and taking most of the code and placing it in a DLL for Windows to use. I will not discuss the actual conversion, although it was quite easy once the code was working in the PC DOS environment. The problems really arose when trying to run regression test programs.

Coding printf in a PC windows environment (standard functions)

There were about 100 separate regression test programs that had to be compiled and run under the PC Windows 3.1 environment. All of these program came from the HP-UX environment and were all character based in nature (they all used standard functions such as *printf()* to write to the screen). The *printf()* function call is not supported in the Windows 3.1 environment. Therefore, my choice was either to change 100 programs from using the *printf()* call or to make up my own *printf()* call. I decided on the latter. I also had to make up function calls for *gets()* and *getchar()*. The last page of this

paper contains the code that I used to imitate *printf()* in the Windows 3.1 environment. It uses the *debugwin.exe* program to display the data by using the *OutputDebugString()* Windows 3.1 function call. It is very basic and was only used to verify results from regression tests. It should not be used for commercial use nor is it in anyway robust. I am including it at the end of this paper in the event that someone might find it useful.

Recommendations

Be aware, before porting to the Windows 3.1 environment, of any calls that may not be supported in the Windows 3.1 environment. The majority of the functions that are not supported in the Windows 3.1 environment have to do with standard input and output functions. Remember to use the Windows 3.1 versions for all string functions that will be used in your application (*strlen*, *strcpy*, *strcmp* - these all become *lstrlen*, *lstrcpy*, *lstrcmp* in the Windows 3.1 environment). Also remember that porting from a character based environment to a Windows environment is a lot easier than porting from a Windows based environment to a character based environment.

Additional Observations

There are a couple other points that I would like to make to ensure that when creating a portable application, you do not make too much work for yourself or the people assigned to maintain the code.

Always think ahead about portability issues

Try to envision every program that you write as being a program that could be used on a variety of platforms. Always think ahead about machine specific routines that may cause problems when trying to port this application to other environments. Be aware of the memory issues that might be present for your application. Always try to create your own datatypes for functions to ensure standardization across all platforms.

Don't use a lot of #ifdefs

A lot of people, when writing portable code, overuse the `#ifdef` preprocessor directive to a point where there is actually two different sets of code within one source file. Try to avoid using `#ifdef` in source code if you can because:

- ◆ A lot of `#ifdefs` makes the code almost impossible to read
- ◆ You need to keep changing the `#ifdefs` every time you port the code to a new environment
- ◆ You basically end up with multiple copies of the same code - one for each environment in which the code runs

I believe, however, that `#ifdefs` are very useful in header files. It is in the header files where you can redefine machine specific functions, create new datatypes, and promote machine dependent code. By using the `#ifdefs` and `#defines` in the header files, you may be able to avoid using them in the source files.

Put all machine dependent code in single file

If you need to create code for machine specific functions, or Windows specific functions, you should put all of these calls into one source file. That way, when porting to a new environment, all machine specific functions will be in the same place and not scattered throughout the whole program source code modules. This makes porting to new environments easy, as well as removing the headaches for overused `#ifdefs` in the code.

My Windows 3.1 printf() function

```
int printf(const char *format_str, ... )
{
    va_list marker;
    int *value[21];
    int counter=0;
    char *big_str;
    char *big_str2;
    char *chptr1,*chptr2;

    big_str=malloc(5000);
    big_str2=malloc(5000);
    va_start(marker,format_str);
    while (counter < 20) {
        value[counter]=&va_arg(marker,int);
        counter++;
    }
    va_end(marker);
    sprintf(big_str,format_str,*value[0],*value[1],*value[2],*value[3],*value[4],
        *value[5],*value[6],*value[7],*value[8],*value[9],*value[10],
        *value[11],*value[12],*value[13],*value[14],*value[15],*value[16],
        *value[17],*value[18],*value[19]);
    OutputDebugString(big_str);
    chptr1=big_str;
    chptr2=big_str2;
    while (*chptr1!=0) {
        *chptr2=*chptr1;
        if (*chptr2=='%') {
            chptr2++;
            *chptr2='%';
        }
        chptr1++;
        chptr2++;
    }
    *chptr2=0;
    fprintf(fp_test_out,big_str2);
    free(big_str);
    free(big_str2);
    return(0);
}
```


Creating a Collaborative Enterprise with Multimedia

by
Charlie Fernandez
Hewlett-Packard Product Manager

Everyone's talking about multimedia and how much it can do for an organization. But we've all heard the wind blow before. Before investing in multimedia, you need to consider what multimedia is and what it isn't.

Definition of Multimedia

There are so many definitions of multimedia that for me to offer another one would be silly (but don't think I'm not tempted). Basically, all of them are trying to express the concept of using text, audio, image, and video data - multiple forms of data used simultaneously - to improve the richness of communication. Actually the term "multimedia," with its connotation of a single entity, might more properly be labeled multiplemedia to emphasize the distinctiveness of the medias included under the term. But then, nobody asked me.

Help or Hype

Multimedia "improves the richness of communication." Granted this sounds like one of those soft-headed, left-winged, liberal ideas that will cost a lot of money but won't amount to a hill of beans. Or, worse yet, it could be marketing fluff. But it happens to be true. Statistics from studies performed by the government (our tax dollars at work) show that, given a human being only has five senses, the more senses engaged to communicate the greater the understanding received. Actually, this should come as no surprise: think of describing to a co-worker a house you just bought; it's a whole lot easier if you have a photograph. By using sight as well as sound, your communication becomes more efficient and your co-worker's understanding becomes quicker and greater.

But let me draw your attention to a subtle point in your conversation with your co-worker: you had to share the picture with your co-worker before the communication became more

effective. That is, the use of multiple medias in and of itself does not guarantee improved communication.

Now you should be able to understand the design focus of MPower and HP's particular spin on multimedia: multimedia technologies by themselves buy you nothing, or rather, next to nothing. It is the collaborative use of these technologies that enriches communication. Me, sitting at my workstation, staring at an image of my new house doesn't help you to visualize the house. So what if I have imaging technology on my workstation, if I can share with you, and you can't see the house, we can't improve our communication.

What a Collaborative Enterprise Requires

Once you realize the potential of multimedia, the next question is which medias included under the term multimedia are of use to you? Not all of them may be useful in your work context. And certainly none of the medias if used by themselves may be useful if a large part of your work context is collaborative.

While you're examining your work context and asking questions, you should ask if the ability to convert from one media to another may be of value. Obviously, converting a video into a series of still images is probably a waste of time, although Walt Disney did quite well going in the other direction. But what about the ability to convert a fax image back into its original ASCII characters through optical character recognition? What about then turning those characters into spoken words using text-to-speech technology? You could then be on a business trip and listen to your fax messages in the hotel room the same way you can listen to your voice mail today. Just something to think about.

You can see that any collaborative enterprise built around multimedia, in order to be effective, requires three types of media technologies:

- multimedia technologies
- collaborative media technologies
- conversion technologies

Let's look briefly at each group of technologies in turn, but before we do let me state that the following groupings are not religious dogma. They're a concept thing. The point is not that a particular media is in a particular group or that a particular item isn't even generally considered a media let alone a multimedia. The point, rather, is that to create a collaborative enterprise with multimedia you need to consider the medias you need, the type of collaboration you need, and any need to convert data from one media type to another.

Multimedia Technology Types

The following list contains items that are generally considered to be the medias included in the umbrella term multimedia (no religion; concept thing):

- Analog video
- Audio
- Digital Video
- Image
- Text

Now the thing to note about each of these medias is that each one uses a particular data format standard. Some include several formats. See Table 1. Obviously, before investing in a particular media, you need to examine the data formats available in that media.

TABLE 1. Multimedia Media Formats

Media	Format Standards
Analog Video	NTSC, PAL, SECAM, S-Video, RGB
Audio	WAV, ALaw, MuLaw, Mac, Linear8, Linear16, Linear8Offset
Digital Video	MJPEG, MPEG
Image	JPEG, Tiff, Gif, Xwd
Text	ASCII,

The format could deter you from using the media. For example, the fact that MPEG, a digital video compression standard, is a lossy compression algorithm - it literally throws away data during the compression process - could deter you from using digital video. Your company lawyers, perhaps, may decide that analog video is more defensible in court because it doesn't lose any data and the heck with your storage problem.

Alternatively, you will need to examine the data formats of a media to discover which format would be most appropriate for your purposes. Industry standard formats are more portable than proprietary formats. What formats are required by the applications you want to run? Or, if you're choosing applications, which one accepts the most formats? What are the differences in multiple formats for the same media? For example, if your work context requires the synchronization of audio and video and you are examining digital video, you need to be aware that

MJPEG includes only video compression, not audio; but MPEG includes both video and audio compression in the same file for true synchronization.

Another point to belabor is that working with the different medias always requires the following three capabilities:

- creation (capture, record)
- modification (author, edit)
- presentation (playback, view)

While at some point in time most multimedia data passes through these three stages, your particular work context may only require one of them. In the case of remote monitoring, for example, you may capture audio or video and play it back, but you may never have the need to edit it. In the case of making product demos, you will probably create multimedia snippets of the product and modify them into the demo through a multimedia authoring program, but you're not the one who is going to watch the demo presentation.

Collaborative Media Types

The following list contains items that could generally be considered collaborative media types:

- Fax
- Shared Whiteboard
- Shared X (application sharing)
- Telephony
- Video Conferencing

Shared X and Shared Whiteboard, as implemented in MPower, don't have a data format as such (See Table 2), but they definitely enrich any collaborative enterprise effort. The other collaborative medias, though they have specific data formats like the multimedia medias listed previously, are more akin to Shared X and Shared whiteboard because they are collaborative by nature. With the exception of fax, all these medias provide real-time collaboration. Like Shared X and Whiteboard, fax, telephony, and video conferencing are not normally used in a single user, stand-alone fashion

I admit that on occasion I have found it convenient to fax myself something. But I have always faxed it to a place where I wasn't yet but planned to be, not a place where I currently was or had just been. Another example of the inherent collabora-

tive nature of the medias listed above is that, although people are frequently in the habit of talking to themselves, they seldom use the telephone to do it.

TABLE 2. Collaborative Media Formats

Media	Format Standards
Computer-based	Analog POTS, ISDN
Telephony	
Fax	Class II Group 3 Class II Group 4
Shared Whiteboard	
Shared X/Shared Apps	
Video Conferencing	H.320, MPEG, MJPEG

Again, as with the multimedias, formats could determine media usage and should also be examined for appropriateness to your work context. Computer-based ISDN telephony, for instance, although digital, won't do you any good if you're connecting to a proprietary digital switch. Another example: if you require caller ID, you may be able to get that functionality simply by upgrading your analog phone service rather than going to the expense of implementing a digital solution.

Conversion Media Types

The following list contains some of the media conversion types:

- Optical Character Recognition
- Scanning
- Speech/Voice Recognition
- Speech/Voice Synthesis
- Text-to-Speech

Conversion medias are the final media type you need to consider in your implementation of a collaborative enterprise. Of all the medias mentioned so far, this area is currently the least defined. Witness the lack of format standards available. See Table 3. With the possible exception of scanning, which relies on imaging formats, and optical character recognition, which relies on the ASCII text format, none of the other conversion medias has a single recognized format standard to follow.

Though somewhat lacking now the use of conversion medias will continue to grow as multimedia becomes more widely accepted in the commercial and technical

business segments. What conversion medias add to the communication picture is the ability to automate and facilitate communication.

TABLE 3. Media Conversion Formats

Media	Format Standards
Optical Character Recognition	
Scanning	Gif, Tiff
Speech/Voice Recognition	
Speech/Voice Synthesis	
Text-to-Speech	

As mentioned earlier, for example, optical character recognition coupled with text-to-speech would enable you to listen to your fax messages over the telephone, thus facilitating communication on a business trip. Speech recognition and synthesis, when used in conjunction with telephony, facilitate communication through automation. It is fairly commonplace today for bank and credit union customers to transfer funds and conduct other financial business over the phone following menus of prerecorded instructions with no intervention from bank personnel.

Summary

Turning your company into a collaborative enterprise through the use of multimedia can have a dramatic effect in improving the richness of communication and, as a result, your organization's bottom line. But multimedia, by itself, isn't the answer. To take advantage of new technologies to improve communication you need to look beyond multimedia technologies to collaborative and conversion technologies as well. These two types of technologies are really what will create a collaborative enterprise in your organization.

5001
Networking and ATM

Richard Nelson
Vanguard Technology
11211 East Arapahoe Road
Englewood, Colorado, USA 80112
(800) 840-6090 • (303) 799-9297 fax • (303) 790-6090
Email: rick@r2.com



A Brief History of Asynchronous Transfer Mode (ATM)

The original concepts for ATM and related techniques (Asynchronous Time Division) were published by AT&T Bell Labs and CNET in 1983. Another similar technique was Fast Packet Switching by Turner in 1983. The Research Center of Alcatel Bell in Antwerp, Switzerland worked with ATM concepts in 1984 and greatly contributed to the technical work on ATM. In the early eighties, field trials of Integrated Services Digital Network (ISDN) were also initiated, and the commercial introduction of ISDN was planned for the late eighties. ISDN would combine voice and data lines into a single transmission line. Alternative network systems and the lack of industry/commercial acceptance delayed the telephony industries plans for ISDN support on transport lines which caused a market stalemate. Telecommunications companies would not invest in ISDN without market demand, the market (users) turned away from ISDN partly due to the lack of transport support and implementation from telecommunications companies. Industry experts agreed that a broadband type of network would be able to support the future services that are needed. Services that were identified were telecommunication (voice), digital TV, digital HDTV, high quality videophony, high speed data transfer, video libraries, home education and video on demand. Ever since the beginning of the eighties, researchers have been experimenting with broadband techniques of various types.

The first standards in broadband networking were defined by the International Consultative Committee for Telecommunications and Telegraphy (CCITT) focusing on the transmission domain. These were based on the Synchronous Digital Hierarchy (SDH) concepts that directly complemented the telecommunication industry. The focus then turned to broadband transfer modes. In 1988 a minimal recommendation was made towards broadband ISDN and that ATM would be the transfer mode of the future. By 1990, CCITT had defined 13 standards recommendations that specified the basics of ATM networking. Today, the ATM Forum (referenced below in this document) is the front line industry spokes group for ATM standards. The ATM Forum is defining the proposed standards for ATM networks.

Existing networks focused on the specific needs for each industry or industry segment. Unique standards for telephony (PSTN), telex, packet switched data networks (PSDN), television using both radio waves, coaxial tree networks (CATV) or direct broadcast systems (satellite), data lines such as local area networks (LAN) - Ethernet, token bus and token ring, all have added to the confusion of trying to merge the transport of these technologies. Each one had services designed specifically for their industry which were not applicable to the other industries. Some examples are the original CATV did not allow voice transport, or that the transfer of voice over an X.25 network was problematic due to too large of an end to end delay and the jitters caused by this delay. Moderate mixing did occur, such as the transport of computer data over voice at minimal speeds, by using modems at both ends. Another issue was the original ISDN, now also known as narrowband ISDN (NISDN) was designed for 64 kbits per second (kbps) voice channels.

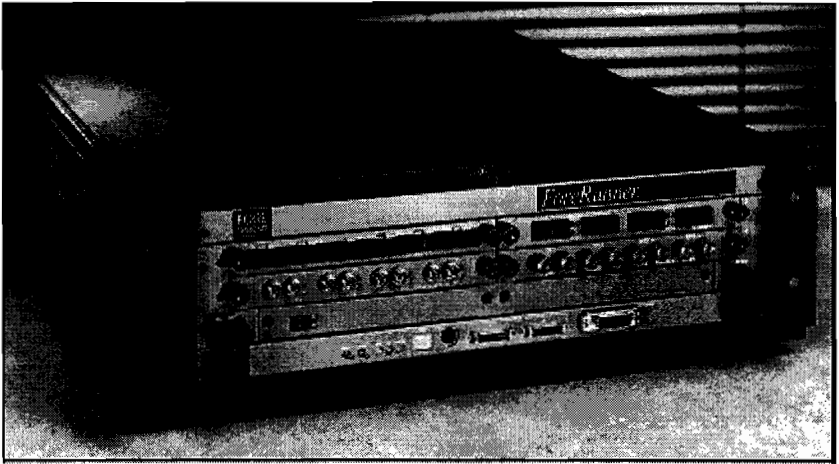
DS0	64kbps	1 voice channel
DS1	1.5mbps	24 voice channels (DS0)
DS1C	3.1mbps	2 DS1s
DS2	6.1mbps	4 DS1s
DS3	45mbps	28 DS1s
OC1	52mbps	
OC3	155mbps	byte interleaved
OC3c	155mbps	concatenated, indivisible payload
OC12	622mbps	
OC12c	622mbps	
OC48	2.5gbps	

ATM Speed Reference

Today, even voice can be transported at 32 kbps using Adaptive Differential PCM encoding techniques (13 kbps for mobile) and even lower rates are being planned. All of these items added to the combined need from all of these industries for a single type of network that could transport all types of signals. Add in the dramatic performance increases in chip technology, optical technologies and the evolution of network systems design where many of the transport functions were migrated to the edges of the network. This allowed more functions to be embedded into chips providing faster speeds, greater reliability, and more economical solutions as quantities increase.

ATM Benefits

Perhaps the single most important advantage offered by ATM, in addition to the speed at which data is transferred, is its open-ended growth path. ATM is not locked into a single physical medium or speed. The fixed size ATM cell allows traffic from multiple sources simultaneously (video, audio, voice, data) to be switched to multiple destinations by fast ATM switches. Unlike shared medium LAN technologies such as Ethernet and Token Ring, in which users must contend for bandwidth, ATM switching provides dedicated, deterministic, high speed connectivity.



High Speed ATM Switch

Overview of benefits;

- ATM will support voice, video and data simultaneously
- ATM uses a multi-gigabit switching fabric
- ATM is scaleable over the fiber, the interface or the entire network
- ATM will simplify internetworking
- ATM switching is very fast due to the system design and hardware
- ATM is suitable for both wide area and the desktop
- ATM will support "real" multimedia at the desktop
- ATM can provide full bandwidth multicast or broadcast
- ATM provides a transparent support of other protocols (i.e.; IP)
- ATM has connection oriented and connectionless services
- ATM provides resource reservation (ask for 50mbps, get 50mbps)

Networking and ATM

5001-3

- ATM has real billing capabilities, pay only for bandwidth you need
- ATM can provide automatic configuration and failure recovery
- ATM can provide dynamic address assignment and resolution
- ATM adheres to the established SNMP net management standards
- ATM inherently provides upgrade paths
- ATM's low latency supports real-time video and audio services
- ATM provides a software route setup and a hardware transport, this results in a very efficient transport of data
- ATM provides true aggregate bandwidth (2.5 + 2.5 does equal 5)
- ATM has contentionless traffic due to front end setup

ATM has the ability to transport any service regardless of characteristics like bit rate, quality requirements, burst rates or load control. As the overall load requirements of networks increase, so will the capacities of ATM. Today, current standards focus on OC3 at 155mbps, yet full trials have been successful at OC12 (622mbps) and OC48 (2.5gbps) rates. Even 5 gbps has been proven feasible. Network speed will now truly be scaleable. Fiber optic lines will not require replacement. In the very near future, we will see ATM gain an even larger acceptance than what Ethernet has seen in the last few years. And the FDDI networks (100mbps) days are numbered. There are some movements to improve FDDI speeds, but ATM components are priced the same as FDDI components and have more advantages. FDDI networks will no longer be chosen within 3 to 5 years.

ATM will provide the single universal network that has been defined as the ultimate solution for future networks. As multiple industries all move to ATM, costs will dramatically fall due to the economies of the vast increase in users from the combined industries. ATM can also provide control over the effective usage of the network. Entire ATM bandwidths can be used for voice, or it may be sliced up for voice, data, or video in any combination, and it can be done dynamically. Identical devices for different industries, and shared transports for different services.

The ATM Forum

Formed in October, 1991, the ATM Forum was established to provide a common industry sponsored direction for the standardization of the ATM networking specifications. In September of 1993, the Forum had over 350 member companies representing all sectors of the computer and communications industries as well as a number of government agencies,

research organizations, and large scale end users. Today there is 581 members. The primary goal of the ATM Forum is to provide multi vendor interoperability between vendors for the ATM industry. The initial UNI (User to Network Interface) specification was completed in June 1992.

An example of the "ATM standards problems" that you hear about is the following. Standardization of desktop ATM was defined as 51mbps over unshielded twisted pair (UTP) wiring. But, a 25 company splinter of the Forum, led by IBM favored the 25mbps over UTP standard stating "customers do not need a faster speed to the desktop, the 25mbps is based on widely implemented Token Ring technology, and 25mbps products will be less complex and therefore less expensive." A formal stance on the 25mbps rate has yet to be confirmed. What is really happening, the primary companies in this industry, worldwide are cooperating and setting standards faster than ever accomplished before. These proposed standards are being officially accepted faster than ever before. The majority of ATM standards are complete. There will be many more standards to follow, there always is. But ATM is functional today and it is possible to integrate systems today, and this integration is rapidly becoming more seamless.

Current ATM Forum User-Network Interface (UNI) Specifications

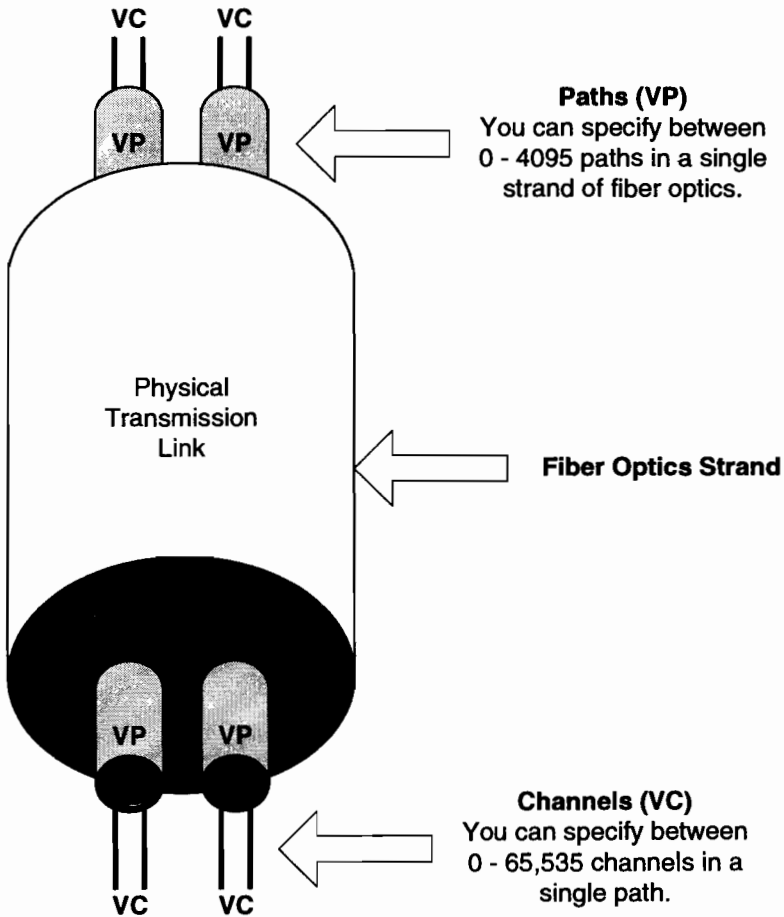
- Physical Layers - 100 mbps multimode fiber (TAXI)
- Physical Layers - 155 mbps multimode fiber
- Physical Layers - 45 mbps DS-3
- Physical Layers - 155 mbps SONET (OC-3c)
- Permanent Virtual Circuits (PVC) and Paths, (SVC's in progress)
- Portions of the signaling, addressing and data flow management
- SNMP MIB for UNI interfaces

ATM Definition and Structure

ATM is a communication architecture based on the switching of small fixed length packets of data called cells. In ATM, all data is transferred in 53-byte cells. Each cell has a 5-byte header that identifies the cell's route through the network and 48-bytes containing the user data. This user data in turn carries any headers or trailers required by higher level protocols. ATM is virtual connection oriented. There must always be a virtual connection established before cells can be sent. There are two types of connections that can be established, Permanent Virtual Circuits (PVC) and

Switched Virtual Circuits (SVC). Basically, PVC is a route connection that is manually pre-defined before use. A SVC is a dynamic route connection similar to the automatic phone connections that we have today.

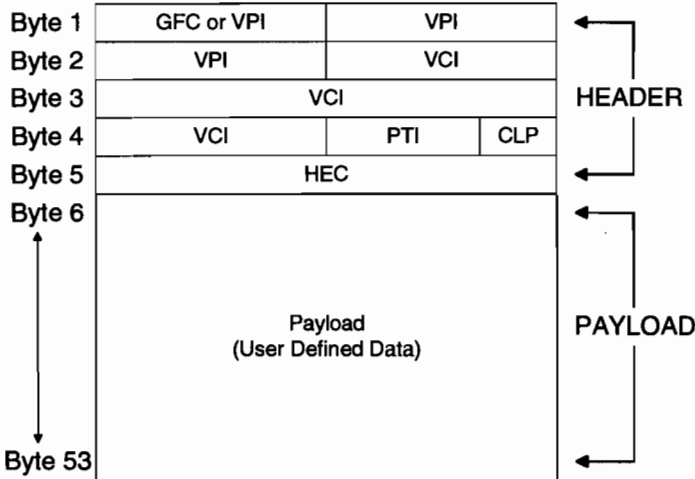
ATM is a sophisticated method of switching and transmitting mixed data that transports large quantities of information over a single strand of fiber optic cable to a specific location. Additionally, ATM may be transmitted over satellite links due to improvements in link enhancements techniques demonstrated in 1993.



Examination of an ATM fiber line

The operation of an ATM switch is conceptually quite simple. The header of each cell contains a virtual connection (VC) identifier, consisting of a virtual path identifier (VPI) and a virtual channel identifier (VCI). On each incoming link, an arriving cell's VC identifier uniquely determines a new VC identifier to be placed in the cell header, and the outgoing link over which to transmit the cell. In the case of a multicast connection, the VC identifier maps to a set of new VC identifiers and outgoing links.

ATM Cell

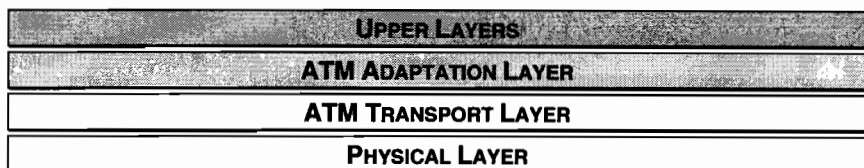


- GFC Generic Flow Control (to be formally defined)
- VPI Virtual Path Identifier
- VCI Virtual Channel Identifier
- PTI Payload Type Indicator
 - Bit 1 - user indication (set for last cell when using AAL5)
 - Bit 2 - indicates congestion upstream
 - Bit 3 - discriminates between data and operation cells
- CLP Cell Loss Priority (if set, discard cell preferentially)
- HEC Header Error Check

Anatomy of an ATM Cell

The user data (payload) is specified as an AAL (ATM Adaptation Layer) type. This defines how the AAL divides the user information into segments suitable for packaging into a series of ATM cells. There are 5 levels of AAL's;

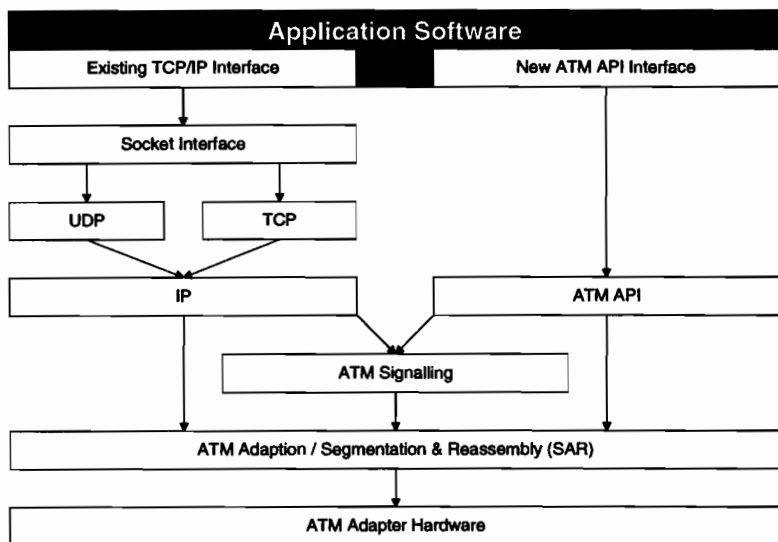
- AAL1 is primarily used for voice data (Constant Bit Rate (CBR))
- AAL2 is used for video (AAL5 is another alternative for video)
- AAL3 (AAL3/4) supports connection-oriented variable bit rate (VBR)
- AAL4 (AAL3/4) supports connectionless variable bit rate (VBR)
- AAL5 is defined as Simple and Efficient Adaptation Layer (SEAL)



ATM Protocol Stack

The addressing for ATM as defined by the ATM Forum and the IETF (Internet Engineering Task Force) (specified in RFC1483 • multiprotocol-over-ATM), is a combination of the E.164 standard and a 48-bit address. You can use one or the other, or both. The E.164 standard is used by the public switched networks, and uses a 4-bit type in a 60-bit address container (15 BCD digit phone numbers). There are also standards defined by OSI using NSAP for the encapsulation of any address. A single protocol is defined on a single virtual connection. The protocol type is defined in the VC identifier. This allows multiple protocol types to travel down the same single strand of fiber optics simultaneously.

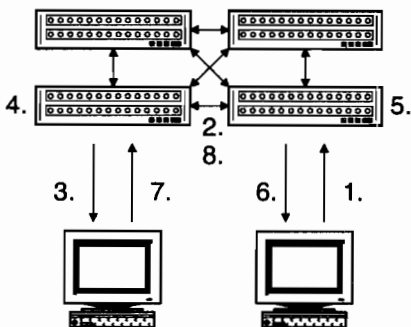
Running IP over ATM is done in the upper layer. The IP address is mapped to the ATM address dynamically, and both PVC and SVC connections are supported. One of the requirements is the use of only AAL3/4 or AAL5 for the adaptation layer. Another is that the IP packets are encapsulated into the ATM payload area. Note: AAL5 uses pure data in all 48 bytes, this provides a 10 to 30% increase in speed compared to AAL3/4. AAL3/4 has all other higher level protocol's control messaging embedded within. AAL5 is more streamlined but has less functionality.



Host Software Architecture for ATM

To make the IP connection, ATM will initiate an ARP over a connectionless channel for the IP destination. It will receive back the ATM address that the IP device maps to. It then will use signaling to establish a virtual connection by associating the IP address to a VPI/VCI stream. The data stream will be disassembled and packaged into ATM cells, transported to the destination and then re-assembled at the network edge as it leaves the ATM medium. A connection can be closed by an end of transmission, or an ARP time-out (20 minutes), or closed by a node failure. If a switch failure occurs, an SVC will automatically attempt to reroute and reconnect to the destination with no loss of data. Live field tests have shown full video tests with links being pulled out and reconnected in a different port or switch and the image will freeze while the cable is pulled, and immediately continue from the same frame immediately (<1 sec) upon reconnect.

SVC Establishment of TCP/IP over ATM



1. IP address registration
2. ARP Request
3. Address response received
4. Connection request
(to destination ATM address)
5. Connection indication
6. Confirmation
7. Confirmation
8. DATA Transfer

Steps involved to make an IP connection over ATM

ATM has the potential to move information at speeds over 1,000 times faster than is currently possible. ATM supports all types of traffic at extremely high rates on one network. It uses fixed size cells (packets) on pre-established channels to carry information between devices, thus providing a very low error rate. The bandwidth that you need is scalable and fixed load caps can be implemented. (Providing one customer 50mbps of a 155mbps line, and dividing the rest of the bandwidth between the other customers.) You will only pay for the amount of bandwidth you use, when traveling over wide area lines. ATM can provide consistent bit rate service with minimal time slice delays which are required for video traffic and medical imaging.

ATM Perceived Problems

The largest scare for everyone has been the question of standards and interoperability between vendors. True, the standards are not complete. But the major focuses are in place, and the long standing Optical Carrier (OC) standards of the telecommunications industry are being used by the ATM industry. You can make different vendors work together, but to do so the management of the multi-vendor network increases dramatically. Try to stick to one vendor as much as possible. The intelligent Switched Virtual Circuits (SVC) are not fully standardized, but they are close (ATM Forum's Q2931 standard). SVC's will make your life pleasant and serene, well maybe not quite that simple... but the benefits are great. SVC's will

set up connections automatically and route on demand. Permanent Virtual Circuits (PVC) are the current method to interconnect different vendors. It works and it's not that difficult (once you understand it), but it is tedious and management intensive.

Too many speeds to pick from... Yes, this can be a true problem with today's ATM. To effectively set-up an ATM network, decide which transport speed you want to focus on and stick with everywhere you can. You can mix speeds somewhat, if a switch has 155mbps boards between other switches, and 45mbps DS3 boards to workstations, it can work. But remember, you cannot fit 155mbps into a 45mbps pipe. You will need to determine your path/channel configuration to effectively utilize your resources. This will cause additional management requirements. Rule of thumb; you can mix speeds and make it work, but don't, it's not worth the headache.

There is many, many vendors jumping into the ATM game, mainly because they don't want to get behind. Find out if your potential vendor is a member of the ATM Forum and clarify that their products are following the Forum's proposed standards. Granted, the ATM Forum is not the standards body, but they are the only central driving force behind ATM and the standards bodies have accepted the ATM Forum as the provider of ATM proposals to them. Do not sign up with a vendor that has their own great new idea, or you may be replacing your network in a few years.

The industry has cited that ATM will be the network of the future, but it will take a few years because the hardware is too expensive. That was true six to twelve months ago. That is no longer true, ATM interface boards are as low as \$995 (Sparc S-BUS), and the switching devices are comparable to some of the current Ethernet routers and high speed switches. Long haul carriers are quickly announcing ATM plans with pricing ranging from 10 to 70% lower than T3 lease lines.

ATM is a new technology that can be confusing. If you are looking to redesign/design your network to eliminate bottlenecks or just increase throughput, consider hiring professional network consultants that have ATM experience. Do not try to implement ATM "without reading the manual". If you want your own administration staff to take care of your ATM network, send someone to class. Any network system is confusing, try setting up a Novell network without training. ATM can be set-up and



running "out of the box", but for on going management, your staff must be trained.

You can run pure ATM protocol, but do you really want too? There is API's established for writing ATM native transports/applications. But IP will run over ATM and has a very well established base of products and services. If you have plans for a specialized application that requires very high speed networking, you might consider using the API. But for the general populace, run IP. I know IP, you know IP, and it will be a protocol that will be around for a long time. Now Novell over ATM is in progress, and it will happen. Novell has insisted it will run on ATM, but it's not there yet, check back in three months (literally). Concerning other protocols, Banyan, Apple, etc. contact a leading ATM vendor and ask what their plans are. The protocols that are supported over ATM are driven by the number of requests by the user community, simple formula of supply and demand.

Case Studies and Implementation Examples

With ATM, an enterprise wide client/server system is truly possible. Existing LAN/WAN solutions make an almost impossible environment for national or global client/server applications. The speed of ATM and the ability to transport mixed data types makes it ideal for a wide area database or any other wide area application.

As application technology increases, so do the demands on the network. We now have graphical groupware, shared CAD drawings, video conferencing at your desk, and mixed media displays to reference data that was only looked at in text form just a few years ago. The user community has learned that graphical/audio/text based screens can provide more data, easier to understand data, and the ability to quickly analyze what is being displayed dramatically faster than the old text only techniques.

Some examples of applications that could benefit from ATM
(Really all networks that transfer large amounts of data, or need very fast access to data, will need ATM.)

- High Speed Data Transport
- Engineering Visualization
- Digital Video Production

- Digital Video Distribution
- Scientific Visualization
- Digital Audio Distribution
- Distance Learning
- Geophysical Modeling
- Molecular Modeling
- Animation
- Medical Imaging
- CAD/CAM/EDA
- Document Imaging
- Multimedia
- Supercomputer Processing
- Financial Modeling
- Video Conferencing
- Virtual Reality
- Distributed Computing
- Distributed Database
- Multimedia Database
- Intelligent PBX Integration
- and the assorted mix of voice, video and data (Interactive TV)

Predictions and ATM Futures

Customer acceptance of ATM network technology has exceeded all vendor expectations. The demand for ATM networks is being driven by the user community. We are already witnessing rapid competition in ATM systems and prices have already started their downward trend. ISDN will be the modems of the future, greater speeds and the acceptance by carriers (which was needed in the eighties) is now happening. ISDN is available in over 90% of Germany, Japan, France and the United Kingdom, but is not widely deployed in the United States. But this is changing, and growth has now been steady with more than 60% of the regional phone companies providing ISDN. Frame Relay will be blown away by ATM's large bandwidth capabilities in Wide Area Networks (WAN). FDDI will fade away as ATM takes over the fiber optic line medium, providing greater flexibility, speed and better pricing. Token Ring will die to slow death, fewer people will consider it, manufacturers will slowly eliminate support due to market demand, and existing Token Ring sites will migrate to new technology as upgrades become required. Some Token Ring ATM installations will stay with the emergence of slow

(25mbps) ATM to fast (155-622+ mbps) ATM links, allowing low bandwidth terminals and network edge components access to full bandwidth services. Ethernet will stick around for quite awhile, providing very low cost access for low bandwidth demands but will eventually be replaced due to desires to have one common network type and the cost reduction in components that ATM will see for the next ten years. The telephone and cable industries will standardize on ATM and provide a tighter alliance with the computer industry. ALL of the major computer manufacturers have stated their intent to support and focus on ATM. An example is Sun Microsystems partnership with Texas Instruments to produce a very low cost two chip ATM set, or DEC's announcement that they consider ATM their future network transport. The Gartner Group has estimated worldwide demand for ATM products will explode to \$3.1 billion by 1997.

ATM is not a national concept. ATM has always and will always be a "global information infrastructure, not a national one." Mike Nelson, assistant director of the Office of Science and Technology Policy. The world has migrated from large central mainframes with dumb terminals, to distributed, high performance systems that can be closely linked logically while linked physically over vast areas around the globe. As networks improve globally, other unrelated markets will benefit from expanded markets and access methods. It will also provide a more rapid economic growth for third world countries, as industrial countries improve information access, the easier it will be to assist in growth.

For a perspective on the importance of networking. The networking trade show Interop, gathered around 20,000 attendees a few years ago. In 1994, Interop had 65,000 attendees, almost 500 exhibitors, and over 6000 hosts connected to the show network. Next year, it is expected that another dramatic increase in the attendance will happen. Interactive television is a buzzword, but a real one. Robert Frankenberg from Hewlett Packard foresees "interactive television as the world's largest transaction processing system." Interactive television and all the associated technologies demand the high performance/high bandwidth that ATM provides today.

Seven years ago we were just considering how to run 10mbps Ethernet over unshielded twisted pair wire. Now we are using fiber optics cabling and the limitation on the fiber has yet to be defined! For now it has been

called unlimited bandwidth potential simply because the limit has yet to be reached.

The most important part of the information era, will be the network the information travels on.

Richard Nelson/Vangard Technology Overview

Richard's background spans 15 years with emphasis in UNIX computer systems. He is currently the National Networking Product Manager for Vangard Technology, an international systems integrator from Englewood, Colorado. He oversees all technical, business and project management efforts in networking including Vangard's ATM line from Fore Systems, Warrendale, Pennsylvania. Richard has provided MIS and lab design, network design and integration, and UNIX management, as well as being a corporate liaison for MCI, Martin Marietta, Puritan Bennett, and SuperComputer class manufacturers. Richard is also president of Tantara Technologies, Littleton, Colorado, a manufacturer of UNIX systems and network management software. Tantara was one of the winners of Open Computing's "Best Products of 1994" with their network assessment software called Knock Knock.

Vangard Technology is an International Systems Integrator with 10 years of experience. Vangard focuses on high speed/high capacity disk and tape products, complete network solutions (including ATM), intelligent data management and database techniques, multimedia management, and professional services that provide complete turn key solutions. Vangard is a certified Fore Systems integrator/reseller, capable of providing not just ATM equipment, but associated products, services, and solutions to satisfy all of our customers. Vangard Technology is pursuing an ATM Forum membership in an auditing role.

Fore Systems is the industry leader in ATM local area networking and wide area access products. Founded in 1990 by four professors from Carnegie Mellon University, Fore is a primary member of the ATM Forum and is leading the ATM evolution.



Using PC, HP3000, HP9000 and IBM machines in Distributed Client/Server Applications

Dr. Mike W. Wang

Palo Alto Information Technology Center

Hewlett-Packard Company

3000 Hanover Street, Palo Alto, CA 94304

Phone: (415) 857-4736

Fax: (415) 857-5518

Introduction

In a client/server information collection and retrieval environment, the client machine usually accepts user requests and provides screen displays. Server machines are usually more powerful systems and are used to store information. The client and server will typically reside on different machines, requiring the application programs to include the often complicated network interface and database retrieval operations. The integration of these operations during application development is not only time consuming, but often too complex for the average programmer to handle. This integration process can become even more complicated when the client and server are on different hardware platforms with different operating systems and networking software.

This paper contains the description of the process that was used to develop a set of common high-level subroutine calls (API) for information collection and retrieval client/server applications running on HP Vectra PC, HP3000, HP9000 and IBM MVS systems. The resulting high-level calls use the standard networking services available on the above mentioned hardware platforms and possess the properties of machine independence, ease of use and high-level functionality for development of client/server applications.

Development process

Communication between two computers, especially for machines with different operating systems, has always been a challenge for software developers. Since the early 1980s, the concept of peer-to-peer communication between two computers has been widely accepted as a desirable feature. However, not until the price of PCs and UNIX workstations was significantly reduced, the performance of these machines drastically improved, and windowing environments became popular in the late 1980s and early 1990s, did peer-to-peer communication evolve to be the basic structure for client/server application development. Nowadays, almost all newly developed applications are designed to run under a client/server environment.

In March 1988, when the HP Advanced Program-to-Program Communication (APPC) product became available to communicate with the IBM LU6.2 product for SNA networks, the task of developing a client/server software application system across HP3000 and IBM machines became achievable in a reasonable amount of time for software developers. At this time, the LU6.2 product was also available on the PC platform. After much trial and error, however, the dial-up method used to connect the PC to a host supporting an LU6.2 application proved very costly and cumbersome. We have a large PC installation base at our facility, so these problems forced us to abandon the use of the SNA network as a viable communication backbone infrastructure for client/server application development projects. It should be noted that a token ring solution was not pursued because most of the machines in our shop did not support token ring at the time.

With the decision not to pursue an SNA solution, the process of finding a common ground for communication between the different types of machines led to a LAN based TCP/IP network.

With the introduction of the Berkeley Sockets interface for LAN based TCP/IP networking in the HP ARPA services product in April 1990, and with the popularity of the Microsoft PC Windows 3.0 environment,

the PC instantly found its place as a desirable client machine for distributed computing.

UNIX systems have been equipped with the Berkeley Sockets network communication interface for LAN based TCP/IP networking since the early introduction of the operating system. This communication service and the wide range of raw processing power made UNIX machines an ideal environment for the development of client/server applications.

HP3000 systems allow programmatic communication with other computers over LAN based networks via the Network Inter-Program Communication (NetIPC) interface. NetIPC uses TCP/IP as the underlying communication mechanism for sending and receiving data, so it allows communications with most systems that provide the Berkeley Sockets network interface. This makes the HP3000 computer suitable to be used as either a client or a server machine in a client/server environment.

We aquired the IBM TCP/IP product for their MVS operating system in 1990. This software provides the developer with a Berkeley Sockets interface and can work in conjunction with channel attached, LAN capable hardware. These offerings made our IBM MVS system a viable client or server machine in our TCP/IP network for client/server applications.

With networking products supporting TCP/IP based communications available on the PC, HP9000, HP3000 and IBM systems, a compatible environment existed supporting inter-system communication over a TCP/IP LAN/WAN network. The following text illustrates the methods used at our facility to develop a generic, user callable interface to inter-communicate between PC, HP9000, HP3000 and IBM systems.

Communication methods

As described in the above text, NetIPC and/or Berkeley Sockets are the available programmatic network interfaces in our shop for

developing client/server applications. To further investigate possible similarities between these two interfaces, a comparison table like the one shown in Figure 1, below, was set up listing the call sequences for each interface for a typical client/server application:

Berkeley Sockets		NetIPC	
Client	Server	Client	Server
	socket() bind()	ipccreate() ipcdest() / ipclookup()	ipccreate() ipcname()
socket() connect()	↔ listen() accept()	ipcconnect() ipcrecv()	ipcrecvn()
send()	→ recv()	ipcsend()	→ ipcrecv()
recv()	← send()	ipcrecv()	← ipcsend()
close() / shutdown()	shutdown() close()	ipcshutdown()	ipcshutdown()

Figure 1. Berkeley Sockets and NetIPC Calls

Definitions and rules for using the above listed calls are not included in this paper as they have been documented in detail in many publications, some of which are listed in the reference section of this paper.

Although these calls appear different in sequence and look, they perform the same function of passing information between client and server programs. When the tasks performed by the various calls are broken down by their basic functionality, it becomes clear that they can be grouped into four major functional categories:

1. Start function calls which include

- *socket()* and *connect()* for Berkeley Sockets implemented clients
- *socket()*, *bind()*, *listen()* and *accept()* for Berkeley Sockets

- implemented servers
- *ipccreate()*, *ipcdest()*, *ipclookup()* and *ipconnect()* for NetIPC
- implemented clients
- *ipccreate()*, *ipcname()* and *ipcrecvn()* for NetIPC
- implemented servers

2. Send function calls which include

- *send()* for Berkeley Sockets implemented clients and servers
- *ipcsend()* for NetIPC implemented clients and servers

3. Receive function calls which include

- *recv()* for Berkeley Sockets implemented clients and servers
- *ipcrecv()* for NetIPC implemented clients and servers

4. Termination calls which include

- *close()* and *shutdown()* for Berkeley Sockets implemented clients and servers
- *ipcshutdown()* for NetIPC implemented clients and servers

We have demonstrated that these functional categories exist in all client and server implementations for both Berkeley Sockets and NetIPC communication methods. This observation led to the idea of building four high-level calls to perform the same four functions in order to simplify, and standardize, the client and server communication interface across all of the platforms. Figure 2, below, shows the relationship of these four function calls, irrespective of their underlying implementation:

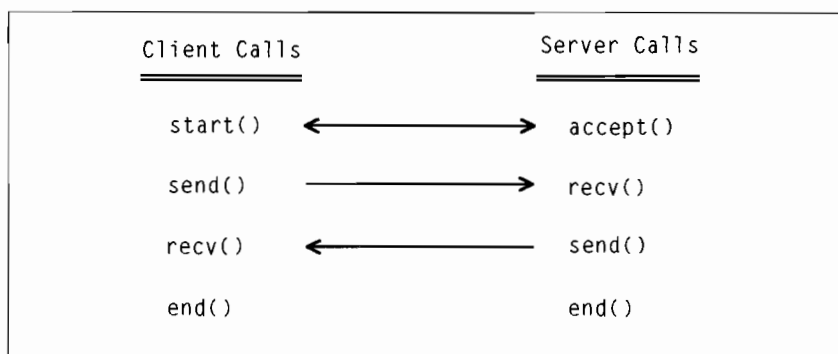


Figure 2. High-Level Calls by Functionality

The *start()* and *accept()* call pair will establish a logical connection between the client and server programs. The *send()* and *recv()* call pair are used by the client and the server programs to send and receive information. The *end()* call terminates the logical connection between the client and the server programs.

The bounds of a session are defined by the *start()* and the *end()* calls in a client program; and by the *accept()* and the *end()* calls in a server program. Within the bounds of a session, the client and server send and receive information using the *send()* and *recv()* calls, respectively.

Please note that we were able to implement the above mentioned high-level calls using both Berkeley Sockets and NetIPC as the underlying communication mechanisms, depending upon which interface was supported on the target platform, and to make the high-level calls the same across all of the platforms.

To add more value to these high-level calls, special capabilities were built into the various calls. The *start()* call, in addition to being used for logical session establishment, also allows certain pieces of information to be exchanged between the client and server such as the maximum amount of data that can be sent in a single send operation, server debugging and logging options, etc.. To uniquely identify a

particular logical connection, a handle is used as a parameter of the *start()* call in a client program. If a second server connection is needed from the same client, a second *start()* call with a different handle is used to uniquely identify the second connection. This flexibility allows a one-to-many client/server architecture to be used in the application design.

To allow asynchronous processing, a timer facility is implemented in the *recv()* call. This functionality permits the developer to supply a specific time interval to the *recv()* call. If the time interval expires before the data has been received, an appropriate status value is set and control is returned to the user's program (i.e. the user's program can continue executing without blocking on the *recv()* call). A *recv()* call can be issued again at a later time to see if the data has arrived. This feature, in conjunction with the ability to have multiple logical connections, provides a software implemented parallel processing environment for the application.

The *send()* call can be used to transmit data or commands to the receiving process. To make it easy for the receiver to determine what kind of information has been received, a special parameter referred to as the transaction ID is transmitted by the sender as part of the *send()* call processing. For example, a specific transaction ID could be used to cause the receiver to invoke a particular command procedure.

A special feature for the *send()* call is to transmit SQL commands to the receiver to access information from a database. For this feature, an SQL database server was developed to receive the SQL commands, to send the SQL commands to the database engine for processing, to receive results from the database engine, and then to send the results back to the client program. This database server potentially allows a client program to access database information from any database machine on the TCP/IP network.

Machine Independence

With the set of high-level calls described above, the underlying

network communication interface, whether it is Berkeley Sockets or NetIPC, is hidden from the application developer. Even as new communication interfaces emerge in the future, the application programs using the described high-level calls could be shielded from code changes if the new interfaces are incorporated into the high-level calls.

Since this set of high-level calls was defined by functionality needed for building client/server applications, the calls were defined with identical names and parameters across all supported platforms. Thus, a program developed using these high-level calls can be ported to any other supported platforms without any program logic or code changes.

Application Case Studies

1. Reseller Profile System (RPS)

A channel reseller profile information client/server application system was created using these high-level calls. This application requires data entry from a PC to build and maintain an HP Allbase database on an HP9000 server. PC windows were built to allow an easy way to perform data entry. Special reporting capabilities were implemented as buttons in the PC application windows. The RPS client/server architecture can be seen in Figure 3, below:

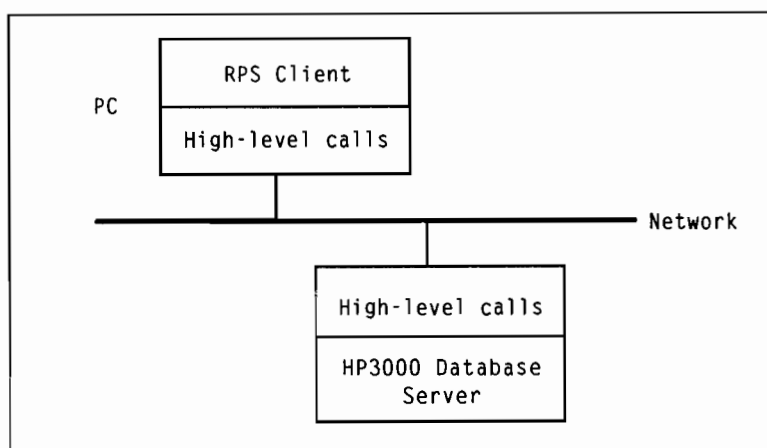


Figure 3. RPS Client/Server Architecture

2. Inventory Control System

The Worldwide Inventory Server (WISE) is a client/server application system used to monitor and evaluate the inventories in our manufacturing facilities. The client program for this application was developed using Informix 4GL on PCs. When the client program is started, it connects to six HP9000 servers using the high-level calls. Four of these servers are located in the United States, one server is in Puerto Rico and another server is in Germany. The client program sends the same SQL query to all six servers at the same time. Thus, the six server machines are working in parallel. When the database query results are obtained, each server returns the results to the client. The WISE client/server architecture is illustrated in Figure 4, below:

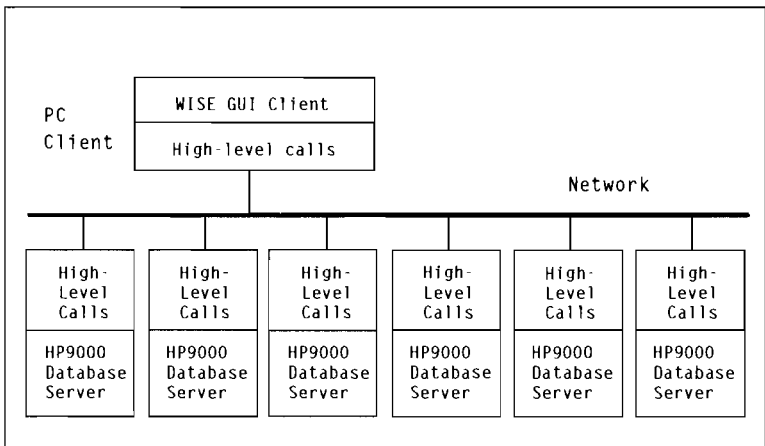


Figure 4. WISE Client/Server Architecture

3. Marketing History Database (Express)

Express is a client/server system used to provide marketing history information for the end users. This application uses Ingres databases residing on HP9000 machines and DB2 databases on an IBM MVS system. A PC client is used to issue user inquiries against the databases. An intermediate server analyzes the requests and determines the best way to obtain the information. The server then sends the SQL command(s) to the appropriate database servers to actually retrieve the database information. The client/server architecture used in the Express application is illustrated in Figure 5, below:

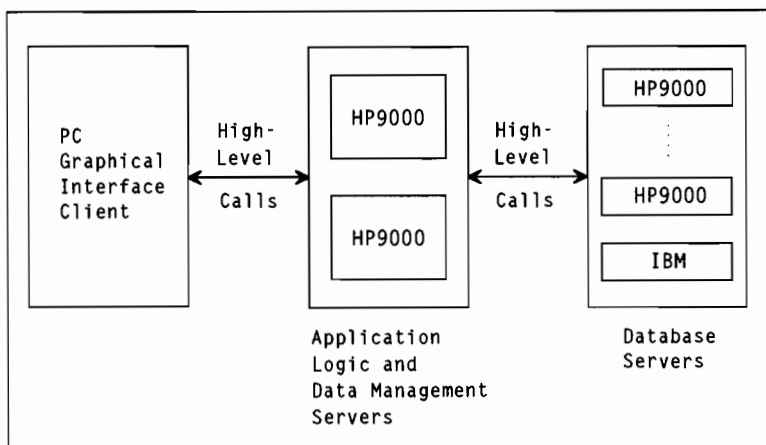


Figure 5. Express Client/Server Architecture

4. FlagShip

FlagShip is a client/server application that keeps track of the shipping events for any shipping requests in our facility. The system creates the shipping database records, retrieves these records and generates reports. The client program is a PC Windows application that was written using Microsoft Visual Basic for the input/output screens and high-level calls for handling the client/server communication. The FlagShip application uses both custom servers and our canned database servers, mentioned earlier in this paper. The FlagShip data is kept in HP TurboImage databases residing on an HP3000 system. Figure 6, below, illustrates the FlagShip system architecture:

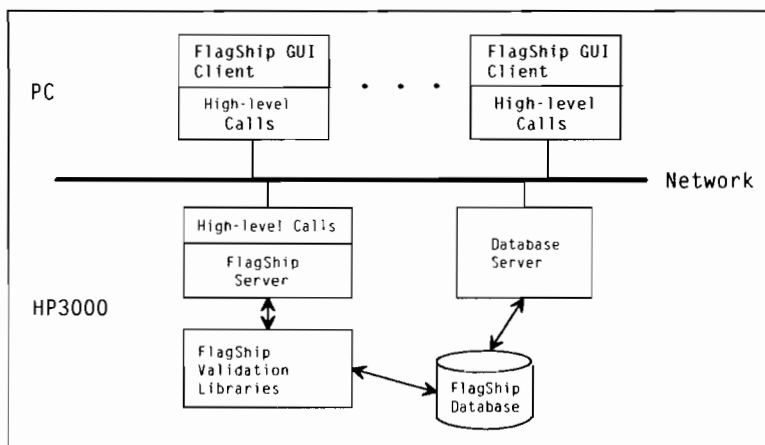


Figure 5. FlagShip Client/Server Architecture

Conclusions

These high-level calls described in this paper have been used in many client/server applications. They have enabled many application programmers to develop client/server programs quickly and easily, without the need to learn underlying networking interfaces and, in many cases, database engine access mechanisms. These high-level calls have therefore widely promoted the client/server technology and greatly benefited the user community at our facility.

References

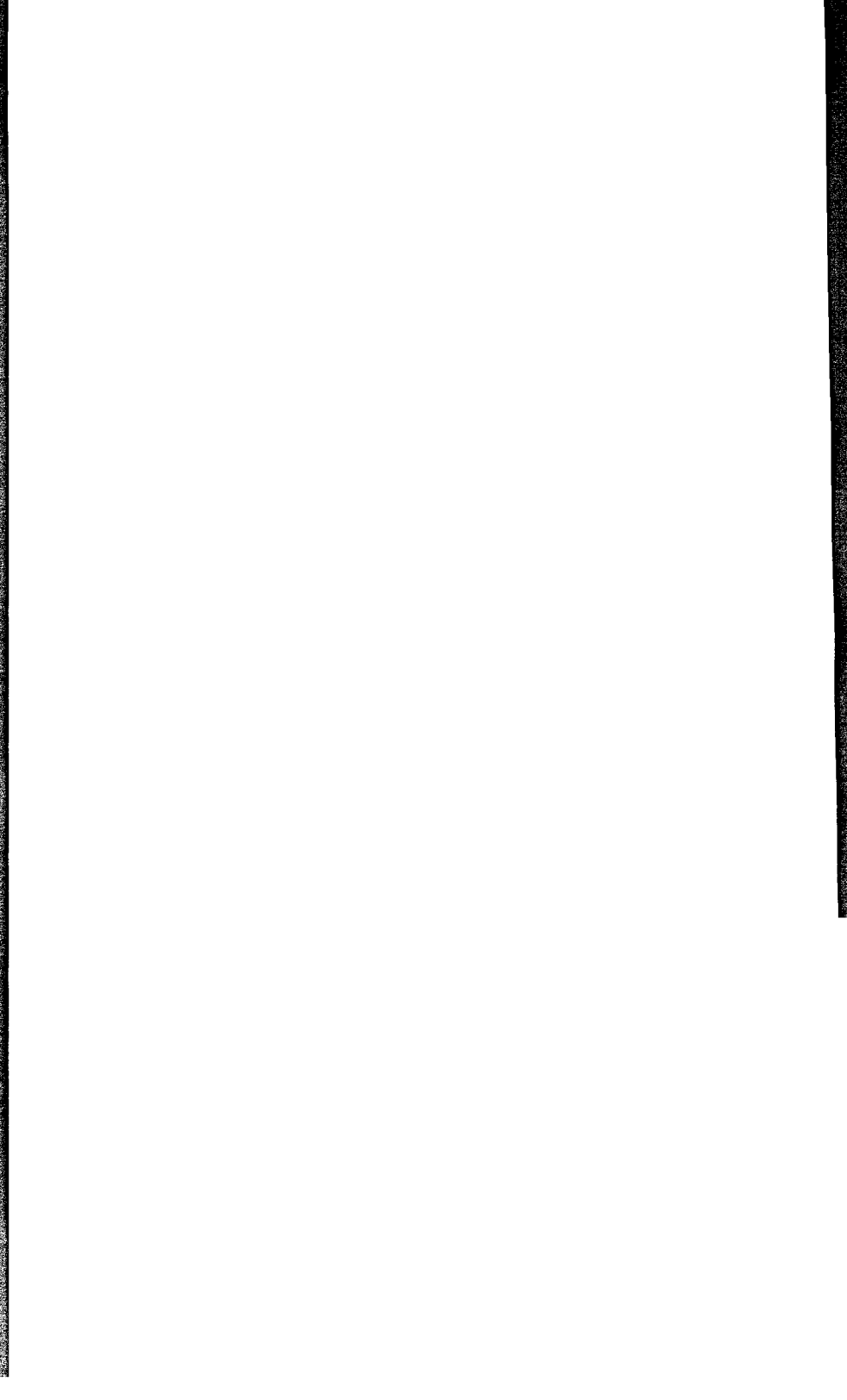
1. NetIPC 3000/XL Programmer's Reference Manual, Hewlett-Packard Co.,
2. UNIX Networking Programming, Stevens, R. W., Prentice-Hall, 1990.

3. NetIPC to BSD IPC (Sockets) Application Note, Hewlett-Packard Co. 1988.
4. Rapid development of Client-Server Applications using RPC, Charles Knouse, Interex 1992.
5. An Introduction to Advanced Program-to-program Communication (APPC), IBM International Technical Support Center, IBM Corp., 1986.
6. VTAM Programming Version 3 Release 1, IBM Corp., July 1988.
7. TCP/IP Version 2 Release 2.1 Programmer's Guide for MVS, IBM Corp., 1992.
8. SNA LU 6.2 API, Information Networks Division, Hewlett-Packard Co., 1988
9. Microsoft LAN Manager: A Programmers Guide, Version 2, Microsoft Press, 1990.

Acknowledgments

The API described in this paper was designed and developed by Arthur Walasek, Christine Hsu, Phil Blocher, Keith Freiberger, Don McKee and Supin Ko of Hewlett Packard. They have devoted much wisdom and hard work, time and time again, to maximize the usefulness and simplicity of this interface. Special thanks also go to Chris Casey and Bill Spangler of Hewlett Packard for their early involvement in putting the procedures and processes in place to make this API available for company-wide use.

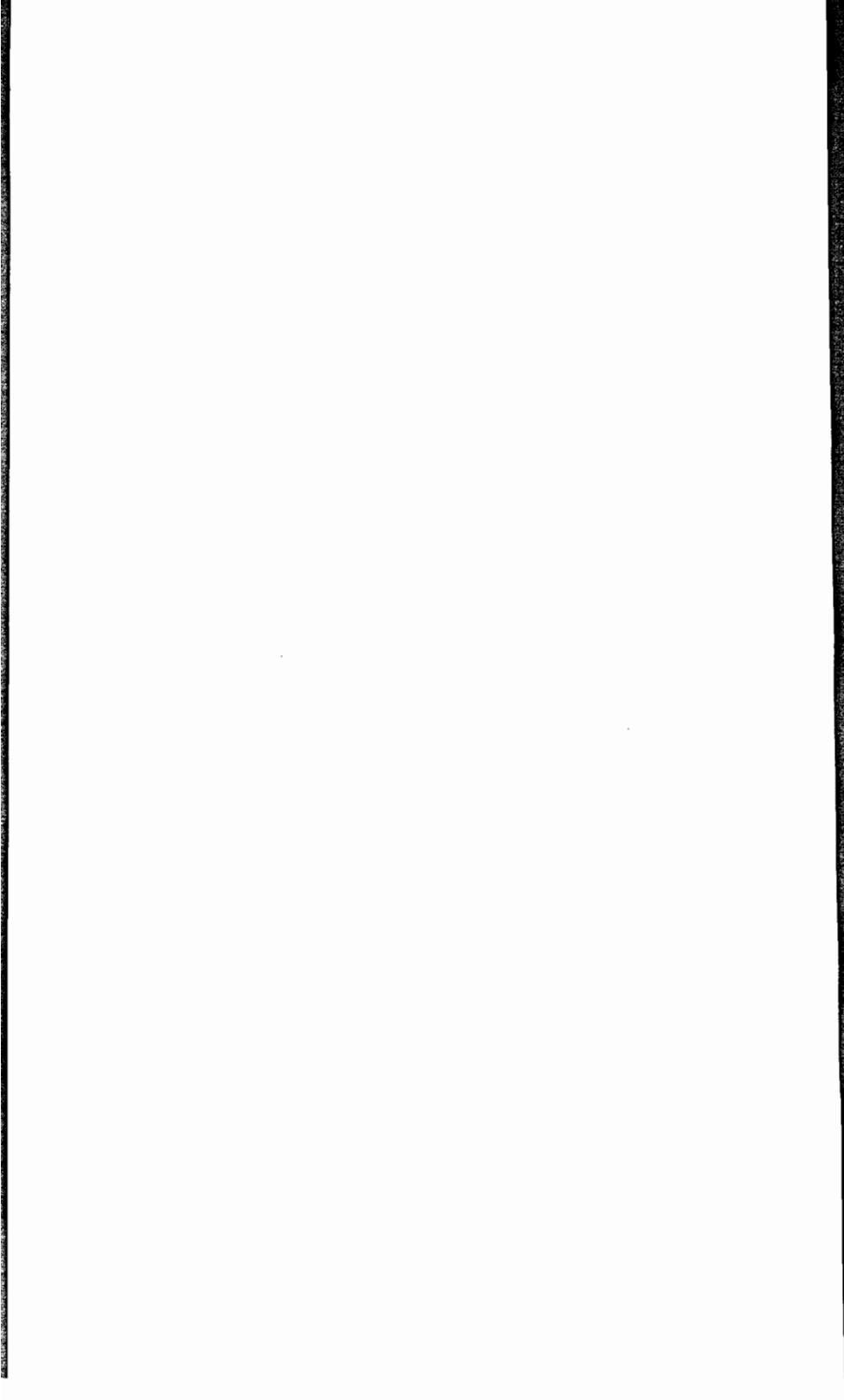
Thanks also go to Sherri Plaza and Jim Lender of Hewlett Packard for their assistance in making the material for the RPS system available; Paul Gladysz and Jim Oberhofer of Hewlett Packard for information pertaining to the WISE application; Yukon Fong and Mark Sturdevant of Hewlett Packard for the Express system materials; and Pei-Lin Hsieh and David Fong of Hewlett Packard for information about the FlagShip system.



Paper Number 5004
The Role of Enterprise Network Services in the Emerging
Global Area Network

Bill Johnson
Banyan Systems
120 Flanders
Westboro, MA 01581

Handouts will be provided at time of presentation



Automating ftp Sessions to an MVS Host

Paper Number: 5005

Presented by:

Rich Case

Nielsen Media Research
375 Patricia Ave.
Dunedin, FL 34698
(813) 738-3000

Objectives

- **Transfer files between UNIX and MVS hosts**
- **Submit jobs (JCL) to an MVS host**
- **Retrieve job output from an MVS host**

Customer Requirements

- Command line invocation
- Execute “quietly”
- Provide for several ftp available options
- Provide security of remote passwords
- Provide help if needed
- Reliable and Portable
- Easy to use and Flexible

Automating ftp Sessions to an MVS host
5005 - 3

Meeting the Needs

- Invoked from the C-shell requiring only 2 arguments
- Does not produce any standard output but it does set \$status upon completion (or exit)
- Input and output directories can be specified
- Allows for ASCII or binary transfers
- “site” arguments can be specified to control the particulars of MVS file allocation

Automating ftp Sessions to an MVS host
5005 - 4

MVS file allocation attributes

- Name
- Logical record length
- Blocksize
- Amount of primary and secondary storage
- Type of storage (blocks, tracks, cylinders)
- Record format (FB, V, VB, ...)
- Retention period
- Model DSN to pattern after
- SMS management class

Automating ftp Sessions to an MVS host

5005 - 5

Meeting more needs

- Remote host passwords stored in users home directory with -rw----- protection
- Log files created automatically and optionally removed
- Log file directory can be specified
- JCL can be submitted with control of the resulting MVS (JES) output

Automating ftp Sessions to an MVS host

5005 - 6

MVS job control options

- Submit a job and exit **or** wait for it to complete and retrieve it's output
- How often to check for job completion
- Retrieve all or a single output spool file
- Name and directory of MVS job output
- Delete or keep token file with job number

Automating ftp Sessions to an MVS host
5005 - 7

Additional benefits

- Help is available through interactive use (invoke with -help or -helpm)
- All options and arguments are checked prior to establishing a connection
- Can be tailored easily by setting default values

Automating ftp Sessions to an MVS host
5005 - 8

Futures ???

- Customize by setting more defaults
- Add “echo” statements and/or prompts to make an interactive version
- Model for unix2??? and ???2unix scripts
- Add a DB2 option

```
#!/bin/csh -f
```

```
#####
```

```
# Name: unix2mvs Created By: Rich Case 1994
```

```
#
```

```
# Usage:
```

```
# unix2mvs -f unixfile -u mvsuser <-h rhost> <-c mvstest>  
# <-d unixdir> <-o mvstest> <-t transtype> <-r> <-e recfm> <-j>  
# <-l lrecl> <-b blocksiz> <-p primary_storage> <-n retpd>  
# <-s secondary_storage> <-a allocation_type> <-y filetype>  
# <-g mgmtclass> <-z sleepsecs> <-k spoolnumber> <-m modeldsn>  
# <-i unixfile_with_MVS_output> <-q> <-x MVS_output_dir>  
# <-w tempfiles_dir> <-help> <-helpm>
```

```
#
```

```
# Arguments:
```

```
# Argument with -a is the allocation type for primary and secondary space.  
# Choose from: BLocks, TRacks, CYlinders. (optional)  
# default = (none) but a default exists on MVS
```

```
# Argument with -b is the blocksiz to use for new MVS datasets.  
# (optional)
```

```
# default = the most lrecls that will fit in 27990 bytes.  
# ie. default for an lrecl of 120 = 27960
```

```
# Argument with -c is high-level nodes to mvstest. Will be used as  
# an absolute name on MVS. (optional)
```

```
# ie. -c HIGH.LEVEL.NODE.MARKET.ONE  
# or -c HIGH.LEVEL.NODE  
# default = mvsuser ( eg. DTARAC )
```

```
# Argument with -d is the UNIX directory path for the file to be  
# transferred. Directories only, relative or  
# absolute. (optional)
```

```
# ie. -d /home/rcase/data
```

```
# default = ones current working directory
```

```
# Argument with -e is the record format for the new dataset. (optional)  
# default = FB
```

```
# Argument with -f is unixfile to be transferred via ftp. (required)
```

```
# ie. -f DATAFILE
```

```
# or -f MARKET.ONE.DATAFILE
```

```
# Argument with -g is the default SMS management class for new datasets.
```

```
# If specified, it can not be validated. (optional)
```

```
# default = (none) but MVS may use one based on  
# the DSN name.
```

```
# Argument with -h is name of remote host (optional)
```

```
# default = mother
```

```
# -help if passed, will cause the begining comment section of this  
# script to be 'headed' out and then exit
```

```
# -helpm if passed, will cause the begining comment section of this
```

script to be 'headed' out, 'piped' to more and then exit

Argument with **-i** is the name of the unix output file in which the retrieved

job's output will be stored. For use with **-j**. (optional)

default = jobname.JOB####.OUT (#### is MVS assigned)

-j indicates that this script will take the steps

required to retrieve held output from a submitted job

during the current invocation of this script.

default = job output will not be retrieved.

Argument with **-k** is the spool number of the held output files to be

retrieved with the **-j** option. (optional)

default = (none) the entire output will be retrieved.

Argument with **-l** is the logical record length to be used on MVS for

new datasets. (optional)

default = length of the longest UNIX file record.

Argument with **-m** is the DSN to use as a model when allocating new

datasets. It can be either fully qualified name in

quotes or appended to the present directory name

prefix. To qualify use: "'dsnname'". When specified,

all other DCB and allocation information passed or

defaulted is overwritten.

default = (none)

Argument with **-n** is the retention period for newly created datasets.

(optional)

default = 0 (meaning none will be specified)

Argument with **-o** is the name used to save the transferred file

as appended to **-c**'s value. (optional)

default = UNIX file name from **-f**

Argument with **-p** is the amount of primary storage to allocate for new

datasets. (optional)

default = (none) but a default exists on MVS

-q if passed, will cause the \$td/P\$\$.OUT file retrieved from

MVS to be deleted after its use. (optional)

default = the \$td/P\$\$.OUT file will not be removed.

-r if passed, will cause the temporary \$td/P\$\$.* (and any

\$td/G\$\$.*) files to be removed after their use.

Note, the files will only be removed if the script

completes successfully. By default the temp files

will not be deleted. This option does not remove the

\$td/P\$\$.OUT file if **-j** was also passed. Use **-q** to remove

the \$td/P\$\$.OUT file. (optional)

Argument with **-s** is the amount of secondary storage to allocate for new

datasets. (optional)

default = (none) but a default exists on MVS

Argument with **-t** is the transfer type to be used: ascii or binary

usually but EBCDIC is available. (optional)

```

#           default = ascii
# Argument with -u is the MVS user id used for login in caps (required)
# Argument with -w is the directory used to place the temporary files:
#           P$$.* (and any G$$.* and P$$.* with the -j option)
#           Must be an absolute path (starts with "/"). (optional)
#           default = ones current working directory
# Argument with -x is the UNIX directory that will be used to place the
#           MVS output file in if the -j option was also passed.
#           (optional)
#           default = ones current working directory
# Argument with -y is the filetype to be used. Choose from: SEQ or JES
#           SEQ is used for standard sequential files
#           JES is used for submitting jobs to MVS. (optional)
#           default = SEQ
# Argument with -z is the number of seconds that will be used with the
#           sleep command while waiting for the submitted job to
#           complete on MVS. It will be used in a loop of 100
#           iterations. For use with -j. (optional)
#           default = 6
# Exit $status values and their meanings:
#
# 5 = an invalid/unexpected argument was passed
# 8 = ftp could not be invoked or completed with errors
# 9 = the value passed with -w is not a directory
#
# 10 = a value for the UNIX file was not passed (unixfile not set)
# 11 = a value for MVS user was not passed (mvsuser not set)
# 12 = the .ftp_info file does not exist in the user's home directory
# 13 = the .ftp_info was not owner read/write only protected (600)
# 14 = a value for the remote host's password was not found in .ftp_info
# 15 = the value passed with -d is not a directory
# 16 = the unix file (-f argument) does not exist
# 17 = the fully specified MVS DSN will be longer than 44 characters
# 18 = the file type JES and an MVS file name or modeldsn were both passed
# 19 = the file type was not SEQ or JES
#
# 20 = a connection could not be made to the specified host (or default)
# 21 = login to MVS was unsuccessful. Invalid userid and/or password.
# 22 = ftp could not "lcd" to the specified local unix dir
# 23 = the transfer type could not be set correctly
# 24 = the remote "working dir" could not be set via "cd mvsdest"
# 25 = an error occurred while trying to establish a model DSN.
#       ie. the file was not found
# 26 = one or more site commands were not accepted

```

```

# 27 = ftp could not successfully "put" (transfer) the file
# 28 = ftp could not quit successfully
# 29 = file not found after transfer (dir unixfile or dir mvsfile failed)
#
# 30 = an alpha letter was specified as or with the number of sleepsecs
# 31 = an alpha letter was specified as or with the spoolnum
# 32 = the local host name could not be found in the .ftp_info file
# 33 = the jobname could not be extracted from the JCL file (-f file)
# 34 = the sleep loop reached 100 iterations waiting for the jobnumber-
#       returning file from MVS. 1) the job did not finish yet
#                               2) there was a JCL error or severe abend
# 35 = the correct job number could not be extracted from the returned
#       file from MVS. (ie. the word EXECUTING could not be found)
# 36 = the value passed with the -x option is not a directory.
# 1## = an error occurred while retrieving the held output via mvs2unix.
#       ## will be an mvs2unix error value.
#       Check the contents of the $td/G$$ftp.* files.
#
# Called From:
#   a shell (prefferably csh so the exiting $status variable can be checked)
#   or any routine that can start a shell script.
#
# Description:
#   This script will invoke ftp to transfer a file from a UNIX machine
#   to an MVS machine.
#   The file may contain "regular" data or JCL statements to be submitted
#   to MVS. The "and back" feature (-j) will retrieve the job's output.
#   A series of arguments are read in to control the ftp process.
#   The UNIX file must be readable by "all others", which includes ftp.
#   Results of the ftp invocation are tested for success to set
#   an exiting status code on error. Arguments can be passed in any order.
#   When transferring larger files (>2M) into new MVS datasets, one should
#   specify correct values with the -a and -p (or -m) options.
#   The user's password on the MVS rhost must be in a file in their (unix)
#   home directory with a name of .ftp_info and protection set to
#   read/write by owner only (600). Data within the file must be in the
#   form: rhost password optional_comments
#   starting in column 1 and separated by tabs.
#
# Notes:
#   When using the "and back" feature, the JCL file must not
#   include "/" as the last line. If it does, the two trailer steps
#   appended to the JCL will not be executed and no output will be
#   returned. The JCL must be "clean" so the trailer steps have a chance
#   to execute. The JOB NAME used must be the MVS USERID plus one (1)

```

```

# additional character. For example: DTARAC3
# The MVS profile switch NOMSGID must be set. This can be displayed
# by typing "profile" from TSO option 6. If it is not set, contact
# your MVS system admin group.
# One's local (unix) host and password must also be included in
# the .ftp_info file. The variable $user must be set to one's local
# user account name. (usually a default)
# If an error should occur while retrieving the job output, the $status
# variable will be set to an "mvs2unix" value plus 100. (ie. 127 -get
# was not successful.)
#
#
#####

```

```

# set ftp related defaults, they may be overwritten by passed arguments
set transtype = "ascii"
set rhost = "mother"
set recfm = "FB"
set retpd = "0"
set ftype = "SEQ"
set sleepsecs = 6
set rmtmp = ""

```

```

# parse options
while ( $#argv > 0 )
  switch ( $1 )
  case -h
    shift
    set rhost = $1
    breaksw
  case -c
    shift
    set mvsdest = $1
    breaksw
  case -f
    shift
    set unixfile = $1
    breaksw
  case -u
    shift
    set mvsuser = $1
    breaksw
  case -d
    shift

```

```
set unixdir = $1
breaksw
case -o
shift
set mvsvfile = $1
breaksw
case -t
shift
set transtype = $1
breaksw
case -r
set removetemps = "Y"
set rmtmp = "-r" # for use with the -j option
breaksw
case -l
shift
set lrecl = $1
breaksw
case -b
shift
set blksize = $1
breaksw
case -p
shift
set prime = $1
breaksw
case -s
shift
set second = $1
breaksw
case -a
shift
set alloctyp = $1
breaksw
case -e
shift
set recfm = $1
breaksw
case -n
shift
set retpd = $1
breaksw
case -y
shift
set ftype = $1
```

```
breaksw
case -m
  shift
  set modeldsn = $1
  breaksw
case -g
  shift
  set mclass = $1
  breaksw
case -z
  shift
  set sleepsecs = $1
  breaksw
case -i
  shift
  set jobout = $1
  breaksw
case -k
  shift
  set spoolnum = $1
  breaksw
case -j
  set andback = "Y"
  breaksw
case -q
  set rmjobfile = "Y"
  breaksw
case -x
  shift
  set joboutdir = $1
  breaksw
case -w
  shift
  set td = $1
  breaksw
case *elpm
  head -202 $0 | more
  exit
case *elp
  head -202 $0
  exit
default
  exit 5
  breaksw
endsw
```



```

shift
end

# verify required arguments, exit if missing
if ( ! $?unixfile ) exit 10

if ( ! $?mvsuser ) exit 11

# verify that if sleep secs were passed, they are non-alphabetic
if ( $?sleepsecs ) then
  if ( `echo $sleepsecs | grep -c '[A-z]'` > 0 ) then
    exit 30
  endif
endif

# verify that if a spool number was passed, it is numeric save "x"
if ( $?spoolnum ) then
  if ( `echo $spoolnum | grep -c '[A-wyz]'` > 0 ) then
    exit 31
  endif
else
  set spoolnum = "x"
endif

# obtain the password from .ftp_info in ones home directory
# (only if it is protected : rw owner only)
if ( ! -f ~/.ftp_info ) exit 12

set protection = `ls -l ~/.ftp_info | cut -f1 -d" "`
if ( "$protection" != '-rw-----' ) exit 13

set mvspass = `fgrep $rhost ~/.ftp_info | cut -f2`
if ( $mvspass == "" ) exit 14

# get local password if "andback" is set. set same as above

if ( $?andback ) then
  set lhost = `hostname`
  set lpass = `fgrep $lhost ~/.ftp_info | cut -f2`
  if ( $lpass == "" ) then
    exit 32
  endif
endif

# determine if a temp_file directory was passed (-w)

```

```

if ( ! $?td ) then
  set td = `pwd`
else
  if ( ! -d $td ) then
    exit 9
  endif
endif

# determin if a unix directory was passed (-d)
if ( ! $?unixdir ) then
  set unixdir = `pwd`
else
  if ( ! -d $unixdir ) then
    exit 15
  endif
endif

# determin if the unix file exists
if ( ! -e $unixdir/$unixfile ) exit 16

# determin if the length of the DSN will be more than 44 charactors
if ( $?mvsdest ) then
  if ( $?mvsfile ) then
    set dsnleng = `echo $mvsdest.$mvsfile | wc -c`
  else
    set dsnleng = `echo $mvsdest.$unixfile | wc -c`
  endif
else
  if ( $?mvsfile ) then
    set dsnleng = `echo $mvsuser.$mvsfile | wc -c`
  else
    set dsnleng = `echo $mvsuser.$unixfile | wc -c`
  endif
endif
if ( $dsnleng > 44 ) exit 17

# make sure an MVS file dest, name or model was not passed for a JES transfer
if ( $ftype == "JES" ) then
  if ( $?mvsdest || $?mvsfile || $?modeldsn ) then
    exit 18
  endif
endif

# make sure ftype is set to SEQ or JES
if ( $ftype != "SEQ" && $ftype != "JES" ) exit 19

```

```

# determin lrecl, if not passed
if ( ! $?lrecl && $ftype == "SEQ" ) then
    set lrecl = `awk 'BEGIN {lrec=0} {if (length > lrec) {lrec = length} } END {print\
lrec}` $unixdir/$unixfile`
endif

# determin blocksize, if not passed
if ( ! $?blksize && $ftype == "SEQ" ) then
    @ blksize = ( 27990 / $lrecl ) * $lrecl
endif

# for a JES transfer and BACK, build trailer steps to return job info
if ( $ftype == "JES" && $?andback ) then

# determin if a job output directory was passed (-x)
if ( ! $?joboutdir ) then
    set joboutdir = `pwd`
else
    if ( ! -d $joboutdir ) then
        exit 36
    endif
endif

set jobname = `fgrep " JOB (" $unixdir/$unixfile | cut -c3-9`
if ( ! $?jobname ) exit 33
# create temp jcl file and then add two trailer steps to it
cp $unixdir/$unixfile $td/P$$.$unixfile

echo "//*****" >> $td/P$$.$unixfile
echo "//*   TSO utility to display JOB number" >> $td/P$$.$unixfile
echo "//*****" >> $td/P$$.$unixfile
echo "//DISPSTEP EXEC PGM=IKJEFT01," >> $td/P$$.$unixfile
echo "//          DYNAMNBR=10,REGION=1M" >> $td/P$$.$unixfile
echo "//SYSTEM DD *" >> $td/P$$.$unixfile
echo "//SYSTSPRT DD DSN=$mvsuser.P$$.OUT," >> $td/P$$.$unixfile
echo "//   DATACLAS=DCCDATAV," >> $td/P$$.$unixfile
echo "//   MGMTCLAS=MC1EX005," >> $td/P$$.$unixfile
echo "//   DISP=(NEW,CATLG)" >> $td/P$$.$unixfile
echo "//SYSTSIN DD *" >> $td/P$$.$unixfile
echo "ST $jobname" >> $td/P$$.$unixfile
echo "/" >> $td/P$$.$unixfile
echo "//*****" >> $td/P$$.$unixfile
echo "//*   FTP the JOB num back to the sending HOST" >> $td/P$$.$unixfile
echo "//*****" >> $td/P$$.$unixfile

```

```

echo "//FTPSTEP EXEC PGM=FTP,REGION=4M      " >> $td/P$$.$unixfile
echo "//SYSPRINT DD SYSOUT=*                " >> $td/P$$.$unixfile
echo "//INPUT DD *                          " >> $td/P$$.$unixfile
echo "$lhost                                " >> $td/P$$.$unixfile
echo "$user                                  " >> $td/P$$.$unixfile
echo "$lpass                                 " >> $td/P$$.$unixfile
echo "cd $td                                 " >> $td/P$$.$unixfile
echo "PUT P$$.OUT                           " >> $td/P$$.$unixfile
echo "QUIT                                   " >> $td/P$$.$unixfile
echo "/"*                                    " >> $td/P$$.$unixfile
echo "//OUTPUT DD SYSOUT=*                  " >> $td/P$$.$unixfile
echo "//                                     " >> $td/P$$.$unixfile

```

endif

create/clear ftp batch files

```

echo "" > $td/P$$.$ftp.in
echo "" > $td/P$$.$ftp.out

```

set protection due to a password in the file

```

chmod 600 $td/P$$.$ftp.in

```

populate batch input file with ftp commands

```

echo "open $rhost" >> $td/P$$.$ftp.in
echo "user $mvsuser $mvspass" >> $td/P$$.$ftp.in
echo "pwd" >> $td/P$$.$ftp.in
if ( $?unixdir && ! $?andback ) then
  echo "lcd $unixdir" >> $td/P$$.$ftp.in
endif
echo "type $stranstype" >> $td/P$$.$ftp.in

```

set mvs directory path, if passed

```

if ( $?mvsdest ) then
  echo cd ""$mvsdest"" >> $td/P$$.$ftp.in
endif

```

set modeldsn to use, if passed

```

if ( $?modeldsn ) then
  echo "quote site dcbdsn=$modeldsn" >> $td/P$$.$ftp.in
endif

```

specify basic DCB information for SEQ files, otherwise, set for JES

```

if ( $ftype == "SEQ" ) then
  echo "quote site lrecl=$lrecl blocksize=$blksize retpd=$retpd recfm="\
  >> $td/P$$.$ftp.in

```

```

else
echo "quote site filetype=$ftype" >> $td/P$$ftp.in
endif

# specify various site commands
if ( $?prime ) then
echo "quote site primary=$prime" >> $td/P$$ftp.in
endif

if ( $?second ) then
echo "quote site secondary=$second" >> $td/P$$ftp.in
endif

if ( $?alloctyp ) then
echo "quote site $alloctyp" >> $td/P$$ftp.in
endif

if ( $?mclass ) then
echo "quote site mgmtclass=$mclass" >> $td/P$$ftp.in
endif

# put temp unixfile with trailer if "andback" is set
if ( $?andback ) then
echo "put $td/P$$.$unixfile" >> $td/P$$ftp.in
else

# set local output filename in put command, if passed
if ( ! $?mvsfile ) then
echo "put $unixfile" >> $td/P$$ftp.in
else
echo "put $unixfile $mvsfile" >> $td/P$$ftp.in
endif
endif

# display info about the file just transferred
if ( ! $?mvsfile ) then
echo "dir $unixfile" >> $td/P$$ftp.in
else
echo "dir $mvsfile" >> $td/P$$ftp.in
endif

echo "quit" >> $td/P$$ftp.in

# initiate a verbose ftp session without autologin
ftp -nv <$td/P$$ftp.in >$td/P$$ftp.out

```

```

if ( $status > 0 ) exit 8

# check the batch output file to see if the transfer was successful
# first set several variables by counting matches with fgrep.
set connectok = `fgrep -ic "connected to " $td/P$$ftp.out`
set loginok = `fgrep -ic "is logged on" $td/P$$ftp.out`
if ( ! $?andback ) then
    set localdirok = `fgrep -ic "local directory now" $td/P$$ftp.out`
endif
set typeok = `fgrep -ic "representation type is" $td/P$$ftp.out`
if ( $?mvsdest ) then
    set mvsvdirok = `fgrep -ic "is working directory name prefix" $td/P$$ftp.out`
endif
if ( $?modeldsn ) then
    set modelok = `fgrep -ic "dcbdsn parameter ignored" $td/P$$ftp.out`
endif
set sitecomok = `fgrep -ic "site command was accepted" $td/P$$ftp.out`
set siteerror = `fgrep -ic ") is not valid" $td/P$$ftp.out`
set putok = `fgrep -ic "transfer completed successfully" $td/P$$ftp.out`
set mvsvdispok = `fgrep -ic "list completed successfully" $td/P$$ftp.out`
set quitok = `fgrep -ic "quit command received" $td/P$$ftp.out`

# set exit status if any are less than one (1) save modeldsn and siteerror
if ( $connectok < 1 ) exit 20
if ( $loginok < 1 ) exit 21
if ( ! $?andback ) then
    if ( $localdirok < 1 ) then
        exit 22
    endif
endif
if ( $typeok < 1 ) exit 23
if ( $?mvsdest ) then
    if ( $mvsvdirok < 1 ) then
        exit 24
    endif
endif
if ( $?modeldsn ) then
    if ( $modelok > 0 ) then
        exit 25
    endif
endif
if ( $sitecomok < 1 || $siteerror > 1 ) exit 26
if ( $putok < 1 ) exit 27
if ( $quitok < 1 ) exit 28

```

```

if ( $mvsdispok < 1 ) exit 29

if ( $?removetemps ) then
  rm -f $td/P$$ftp.in $td/P$$ftp.out $td/P$$.$unixfile
endif

if ( ! $?andback ) exit 0

#####
# continue only if held output is going to be retrieved
#####

# set protection to owner read and write only to temp JCL file
# if not already removed ( could not do sooner because ftp had to read it)
if ( ! $?removetemps ) then
  chmod 600 $td/P$$.$unixfile
endif

# retrieve output from the job just sent after the file with the JOB number
# in it returns. The sleep loop will be iterated 100 times before exiting.

@ i = 1
while ( $i < 101 )
  sleep $sleepsecs
  if ( -f $td/P$$OUT ) then
    break
  else
    @ i = $i + 1
  endif
end

# make sure we "broke" out of the loop, if not exit - timeout
if ( $i > 100 ) exit 34

# sleep a few seconds to allow the MVS job to really finish
sleep 5

# extract the job number from the $td/P$$OUT file
set jobnum = `fgrep " EXECUTING" $td/P$$OUT | cut -c13-20`

if ( $status > 0 ) exit 35

# set job output file to default if not passed
if ( ! $?jobout ) then
  set jobout = $jobname.$jobnum.OUT

```

endif

execute the mvs2unix script to actually get the held output
**mvs2unix -u \$mvsuser -f \$jobnum.\$spoolnum -y JES -o \$jobout \$rmtmp **
-d \$joboutdir -w \$td
set m2ustat = \$status

check exiting status from mvs2unix
if (\$m2ustat > 0) then
@ gstat = \$m2ustat + 100
exit \$gstat
endif

remove job number file if -q was passed
if (\$?rmjobfile) rm \$td/P\$\$OUT


```

#!/bin/csh -f
#####
# Name: mvs2unix                Created By: Rich Case   1994
#
# Usage:
#   mvs2unix -f mvsfile -u mvsuser <-h rhost> <-c mvscat>
#           <-d unixdest> <-o unixfile> <-t transtype> <-r>
#           <-y filetype> <-v> <-w tempfiles_dir> <-help> <-helpm>
#
# Arguments:
#   Argument with -c is high-level nodes to mvsfile. Will be used as
#   an absolute name on MVS. (optional)
#   ie. -c HIGH.LEVEL.NODE.MARKET.ONE
#   or -c HIGH.LEVEL.NODE
#   default = mvsuser ( eg. DTARAC )
#   Argument with -d is the UNIX destination path for the transferred
#   file. Directories only. (optional)
#   ie. -d /home/rcase
#   default = ones current working directory
#   Argument with -f is mvsfile to be transferred via ftp. OR the jobid
#   (and optional spool number) to retrieve from the
#   held output queue.
#   ie. -f DATAFILE or -f MARKET.ONE.DATAFILE
#   ie. -f job00332.2 or -f job00332.x
#   (the ".x" indicates to retrieve all spools for
#   the specified job.)
#   Argument with -h is name of remote host (optional)
#   default = mother
#   -help if passed will cause the begining comment section of this
#   script to be 'headed' out and then exit
#   -helpm if passed will cause the begining comment section of this
#   script to be 'headed' out, 'piped' to more and then exit
#   Argument with -o is the name used to save the transferred file (optional)
#   default = MVS file name from -f
#   -r if passed, will cause the temporary G$.ftp.in and
#   G$.ftp.out files to be removed after their use.
#   Note, the files will only be removed if the script
#   completes successfully. By default the temp files will
#   not be deleted. (optional)
#   Argument with -t is the transfer type to be used: ascii or binary usually.
#   (optional)
#   default = ascii
#   Argument with -u is the MVS user id used for login
#   -v if passed, will cause the RDW from VB/VBS files to be
#   retained as data. (optional)

```

```

#           default = not specified, the RDW is dropped
# Argument with -w is the directory used to place the temporary files: G$$.*
#           Must be an absolute path (starts with "/"). (optional)
#           default = ones current working directory
# Argument with -y is the filetype to be used. Choose from : SEQ or JES
#           SEQ is used for standard sequential files
#           JES is used for retrieving job output from held output
#           (optional)
#           default = SEQ
#
# Exit status values and meanings:
#
#           5 = an invalid/unexpected argument was passed
#           8 = ftp could not be invoked or completed with errors
#           9 = the value passed with -w is not a directory
#
#           10 = a value for the MVS file was not passed (mvsvfile not set)
#           11 = a value for MVS user was not passed (mvsvuser not set)
#           12 = the .ftp_info file does not exist in the user's home directory
#           13 = the .ftp_info was not owner read/write only protected (600)
#           14 = a value for the remote host's password was not found in .ftp_info
#           15 = the file type was not SEQ or JES
#           16 = the file type was set to JES and an mvsvcat was passed
#           17 = the value passed with -d is not a directory
#
#           20 = a connection could not be made to the specified host (or default)
#           21 = login to MVS was unsuccessfull. Invalid userid and/or password.
#           22 = ftp could not "lcd" to the specified local dir
#           23 = the transefer type could not be set correctly
#           24 = the remote "working dir" could not be set via "cd mvsvcat"
#           25 = the specified MVS file does not exist. (ls mvsvfile failed)
#           26 = one or more site commands were not accepted
#           27 = ftp could not successfully "get" (transfer) the file
#           28 = ftp could not quit successfully
#           29 = file not found after transfer (ls <unixdest/unixfile> or
#               ls mvsvfile failed)
#
# Called From:
#           a shell (preferably csh so the exiting $status variable can be checked)
#           or any routine that can start a C-shell script.
#
# Description:
#           This script will invoke ftp to transfer a file from MVS to
#           the calling UNIX box (or its NFS mounted files).
#           A series of arguments are read in to control the ftp process.

```

```

#      Results of the ftp invocation are tested for success to set
#      an exiting status code. Arguments can be passed in any order.
#      The user's password on the remote host must be in a file in their
#      home directory with a name of .ftp_info and protection set to
#      read/write by owner only (600). Data with the file must be in the
#      form: rhost      password      optional_comments
#      starting in column 1 and separated by tabs.
#
#####
#set some defaults they may be overwritten by passed arguments
set transtype = "ascii"
set rhost = "mother"
set ftype = "SEQ"

# parse options
while ( $#argv > 0 )
switch ( $1 )
case -h
  shift
  set rhost = $1
  breaksw
case -c
  shift
  set mvscat = $1
  breaksw
case -f
  shift
  set mvfile = $1
  breaksw
case -u
  shift
  set mvuser = $1
  breaksw
case -d
  shift
  set unixdest = $1
  breaksw
case -o
  shift
  set unixfile = $1
  breaksw
case -t
  shift
  set transtype = $1
  breaksw

```

```

case -r
  set removetemps = "Y"
  breaksw
case -y
  shift
  set ftype = $1
  breaksw
case -v
  set sendrdw = "Y"
  breaksw
case -w
  shift
  set td = $1
  breaksw
case *elp
  head -100 $0
  exit 0
  breaksw
case *elpm
  head -100 $0 | more
  exit 0
  breaksw
default
  exit 5
  breaksw
endsw
shift
end

```

```

# verify required arguments, exit if missing
if ( ! $?mvswfile ) exit 10

```

```

if ( ! $?mvswuser ) exit 11

```

```

# determine if a temp_file directory was passed (-w)

```

```

if ( ! $?td ) then
  set td = `pwd`
else
  if ( ! -d $td ) then
    exit 9
  endif
endif
endif

```

```

# obtain the password from .ftp_info in ones home directory
# (only if it is protected : rw owner only)

```

```

if ( ! -f ~/.ftp_info ) exit 12

set protection = `ls -l ~/.ftp_info | cut -f1 -d" "`
if ( "$protection" != '-rw-----' ) exit 13

set mvspass = `fgrep $rhost ~/.ftp_info | cut -f2`
if ( $mvspass == "" ) exit 14

# make sure file type is set to SEQ or JES
if ( $ftype != "SEQ" && $ftype != "JES" ) exit 15

# if ftype = JES and an MVSCAT was passed exit : user confusion
if ( $ftype == "JES" && $?mvscat ) exit 16

# determin if a unix directory was passed (-d)
if ( ! $?unixdest ) then
    set unixdest = `pwd`
else
    if ( ! -d $unixdest ) then
        exit 17
    endif
endif

# create/clear ftp batch files
echo "" > $td/G$$ftp.in
echo "" > $td/G$$ftp.out

# set protection due to a password in the file
chmod 600 $td/G$$ftp.in

# populate batch input file with ftp commands
echo "open $rhost" >> $td/G$$ftp.in
echo "user $mvsuser $mvspass" >> $td/G$$ftp.in
echo "pwd" >> $td/G$$ftp.in
echo "lcd $unixdest" >> $td/G$$ftp.in
echo "type $stranstype" >> $td/G$$ftp.in

# set mvs directory path, if passed
if ( $?mvscat ) then
    echo cd ""$mvscat"" >> $td/G$$ftp.in
endif

# specify SEQ or JES source
echo "quote site filetype=$ftype" >> $td/G$$ftp.in

```

```
echo "ls $mvsfile" >> $td/G$$$.ftp.in
```

```
# optionally set RDW switch
```

```
if ( $?sendrdw ) then
```

```
    echo "quote site RDW" >> $td/G$$$.ftp.in
```

```
endif
```

```
# set local output filename in get command, if passed
```

```
if ( ! $?unixfile ) then
```

```
    echo "get $mvsfile" >> $td/G$$$.ftp.in
```

```
    else
```

```
    echo "get $mvsfile $unixfile" >> $td/G$$$.ftp.in
```

```
endif
```

```
echo "quit" >> $td/G$$$.ftp.in
```

```
#initiate a verbose ftp session without autologin
```

```
ftp -nv <$td/G$$$.ftp.in >$td/G$$$.ftp.out
```

```
if ( $status > 0 ) exit 8
```

```
# check the batch output file to see if the transfer was successful
```

```
# first set several variables by counting matches with fgrep.
```

```
set connectok = `fgrep -ic "connected to " $td/G$$$.ftp.out`
```

```
set loginok = `fgrep -ic "is logged on" $td/G$$$.ftp.out`
```

```
set localdirok = `fgrep -ic "local directory now" $td/G$$$.ftp.out`
```

```
set typeok = `fgrep -ic "representation type is" $td/G$$$.ftp.out`
```

```
if ( $?mvscat ) then
```

```
    set mvsdirok = `fgrep -ic "is working directory name prefix" $td/G$$$.ftp.out`
```

```
    set siteerror = `fgrep -ic ")" is not valid" $td/G$$$.ftp.out`
```

```
endif
```

```
set mvscat = `fgrep -ic "list completed successfully" $td/G$$$.ftp.out`
```

```
set sitecomok = `fgrep -ic "site command was accepted" $td/G$$$.ftp.out`
```

```
set getok = `fgrep -ic "transfer completed successfully" $td/G$$$.ftp.out`
```

```
set quitok = `fgrep -ic "quit command received" $td/G$$$.ftp.out`
```

```
# set exit status if any are less than one (1)
```

```
if ( $connectok < 1 ) exit 20
```

```
if ( $loginok < 1 ) exit 21
```

```
if ( $localdirok < 1 ) exit 22
```

```
if ( $typeok < 1 ) exit 23
```

```
if ( $?mvscat ) then
```

```
    if ( $mvsdirok < 1 ) then
```

```
        exit 24
```

```
    endif
```

```
endif
if ( $mvsvfileok < 1 ) exit 25
if ( $sitecomok < 1 || $siteerror > 1 ) exit 26
if ( $getok < 1 ) exit 27
if ( $quitok < 1 ) exit 28
if ( $?unixfile ) then
  if ( ! -e $unixdest/$unixfile ) then
    exit 29
  endif
  else
  if ( ! -e $unixdest/$mvsvfile ) then
    exit 29
  endif
endif
endif

if ( $?removetemps ) then
  rm -f $td/G$$$.ftp.in $td/G$$$.ftp.out
endif

exit 0
```


Paper 5007

A Global HP PC Common Operating Environment Implementation

By Brandt Faatz, Corporate Information Systems
Hewlett-Packard Company
3000 Hanover Street, Palo Alto, CA 94304
(415) 691-7374

As the world economy tightens and markets become more competitive, HP needs to get more work done with fewer resources. To increase our productivity, we need to apply new technology at an ever-increasing rate. At the same time, our business organizations are becoming increasingly complex and geographically distributed. Information Technology (IT) organizations are consolidating and downsizing to reduce cost. With all of this going on, how can IT continue to create a competitive advantage for its internal customers? How can IT help to improve productivity and reduce cost in a complex, geographically-distributed company?

The Personal Computer Common Operating Environment (PC COE) is one way Hewlett-Packard has met this challenge during the past three years. PC COE manages desktop and mobile PCs using network-based tools and processes. The PC COE service has been very effective at getting more value from information technology while keeping costs under control. The PC COE concepts, processes and technologies can be applied to other companies as well. To understand how HP implemented PC COE, we'll review the problem PC COE was designed to solve and the strategy used to solve it. We'll discuss a model used for marketing PC COE's services within the company. Then we'll describe PC COE's services and the processes used to provide them. We'll review the benefits to application providers, IT organizations and end-users. Finally, we'll list the steps necessary to create a desktop management service.

The Problem

Before PC COE, HP could not get a satisfactory return on its PC investment. PCs were neither proactively managed nor well-integrated into the overall computing environment. The problem was divided into four key areas:

- ◆ reliability,
- ◆ supportability,
- ◆ ability to share information,
- ◆ effectiveness as a client/server platform, and
- ◆ cost of PC ownership.

Reliability Problems

As the rate of new technology adoption increased, PCs became increasingly unstable. Problems were introduced each time a user installed new software. Software vendors' installation tools required the user to make configuration choices. AUTOEXEC.BAT, CONFIG.SYS and Windows SYSTEM.INI entries were made incorrectly, and PCs sometimes couldn't even boot. Users became increasingly frustrated as they rebooted their hung systems, lost hours of work and spent more time trying to solve technical problems.

Supportability Problems

At the same time, PC support calls swamped HP's IT help desks. Help desk personnel spent an average of one hour per software installation. Configurations became more and more complex, and problems were increasingly difficult to solve. Most support calls could not be handled over the phone because there was no way to know how the PC was configured. At large sites, help desk personnel often spent as much as 20 minutes per problem just walking to and from the user's desk. Geographically-distributed organizations had to increase their staff substantially in order to provide on-site support for their remote sites.

Information Sharing Problems

Meanwhile, users began to understand PC technology, and they wanted the maximum benefit. They wanted to improve their productivity and business processes through automation and information sharing. Office productivity tools such as spreadsheets, word processors and presentation graphics programs became very popular. New tools such as personal information managers, personal databases and project management tools started to catch on. One of the great benefits of the PC is the availability of a wide variety of software tools. Unfortunately, these tools were implemented in a very unmanaged way. Sometimes departments standardized on a set of tools, but more often each user picked a completely different set. Process automation and information sharing were nearly impossible because of the variety of software tools and versions.

Client/Server Platform Problems

The problem got a lot worse. Client/server technology promised a better way to develop software applications, and development groups embraced this exciting new technology. Meanwhile, an increasing number of applications were purchased from third-parties and had to be integrated into Hewlett-Packard's computing environment. The PC was not ready to become a client for mission-critical, client/server applications, but the businesses needed the applications. We had to have a way to make sure the right versions of the operating system, networking software, Application Programming Interfaces (APIs) and utilities were installed

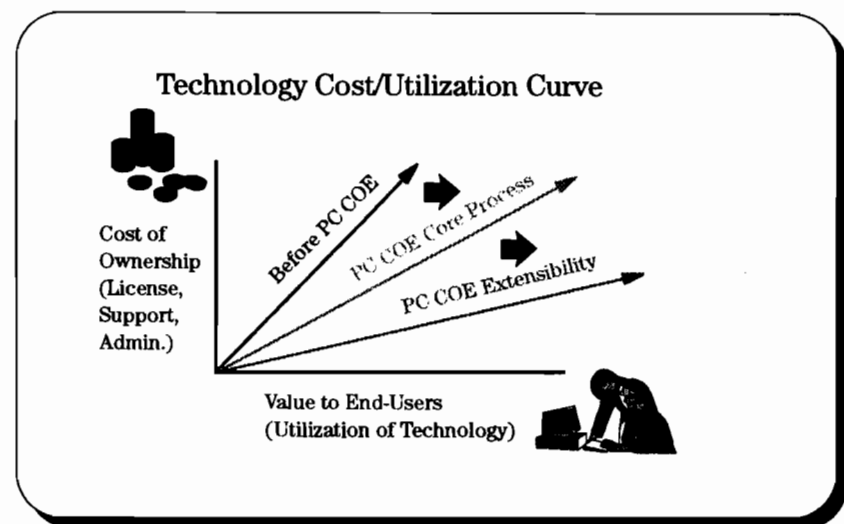
on all PCs. Many applications conflicted with each other, and we needed a way to predict and manage the conflicts. HP began to roll out a variety of client/server applications to a chaotic, unpredictable, unreliable and unsupported base of 60,000 PCs around the world.

Cost of PC Ownership

The steady influx of PC technology drove the cost of PC ownership to an unacceptable level. Because the PCs were so unreliable, end-users lost hours of productivity. Because the PCs were so difficult to support, IT needed one help desk employee for every 50 users. In most cases, IT couldn't staff at that level, so effective support was impossible. Because electronic information was so difficult to share, most users couldn't do it. With mission-critical applications being deployed, critical business processes were disrupted. HP needed to get its PCs under control.

The Common Operating Environment Strategy

The Common Operating Environment strategy was developed to control the cost of technology ownership. Cost is a function of technology implementation. As new technology is introduced, costs increase. As technology matures, it becomes well-managed, and costs decrease. PC COE manages PC technology implementation to minimize cost increases and to maximize technology utilization. The result is a positive return on the PC investment. The technology cost/utilization model was presented to an executive steering committee to get sponsorship for PC COE implementation.



As the diagram shows, the value technology provides to end-users can be mapped against the cost of owning that technology. If the curve is too steep, the return on the technology investment doesn't justify its implementation. Either new technology will not be used, or it will be used very inefficiently.

The PC COE strategy focused on five objectives to flatten the cost/utilization curve.

- ◆ Connect all clients to the network
- ◆ Standardize configurations
- ◆ Manage configurations via the network
- ◆ Distribute a common, core set of office productivity tools
- ◆ Develop a process for integrating applications into the environment

Connect All Clients To The Network

All of the 386/486 PCs were connected to the network. They became known as "PC COE clients" rather than "personal computers." This is an important distinction. It means the PCs are part of a networked computing system instead of independent machines. It was also the source of some resistance from the more individualistic PC users. Some users felt they were giving up control of their computers. They feared a return to the days when the "Electronic Data Processing" department was relied upon for every computing task. An internal public relations effort was needed to help users understand the benefits of the network and to quell their fears.

Standardize Configurations

The second step was to standardize the PC configurations to the highest reasonable degree. This was particularly challenging because of the variety of PC hardware. In Hewlett-Packard, the vast majority of the PCs are HP Vectras, so that simplified the problem somewhat. Still the variety of printers, disk drives, sound adapters and other peripheral devices was somewhat daunting. The software configurations were easier to standardize. Directory structures and network device standards were established. Network software and Windows configurations were standardized. A common set of utilities were made available to each PC via the network. But all this standardization would have been useless without the capability for ongoing configuration management.

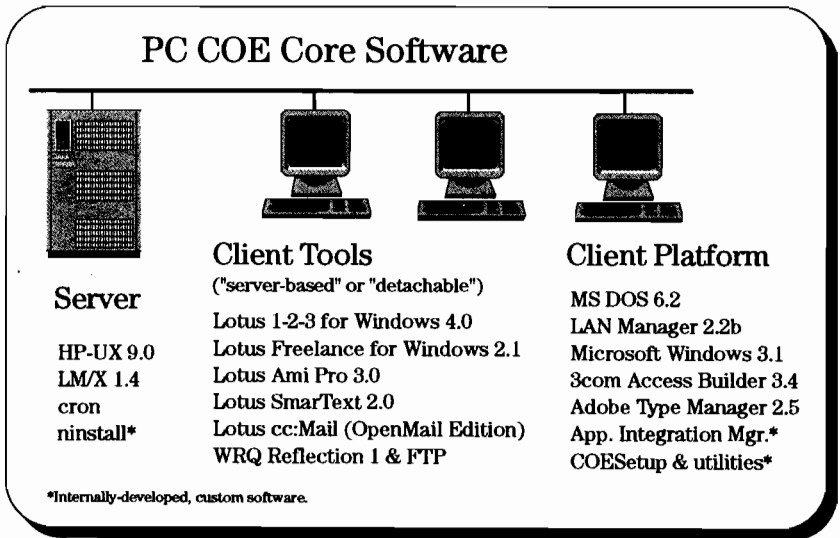
Manage Configurations Via The Network

Standardize configurations was the first step, but new hardware and software is being deployed all the time. A process was developed to manage the configurations via the network. Tools were created to set up, track and change

configurations. Once a PC was set up on PC COE, it would be managed by the PC COE process on an ongoing basis.

Deliver Common Office Productivity Tools

To motivate the users and to help solve the information sharing problem, a common set of office productivity tools was delivered to each PC. Spreadsheet, word processing, presentation graphics, terminal emulation, e-mail and other tools were included. Tools were evaluated based on functionality, reliability, multi-platform (DOS/HP-UX) support and cost. A migration was conducted from the vast variety of office productivity tools to a relatively small set of managed tools. To ease resistance, the common tool set was implemented without requiring the removal of competing tools. Instead, the common set was positioned as the tools to use for data sharing. The combination of server software, client platform software and office productivity tools comprises the "PC COE Core".



The common tool set has been kept intentionally small during the past three years. Instead of adding new tools, we have focused our efforts on a process for integrating tools and applications.

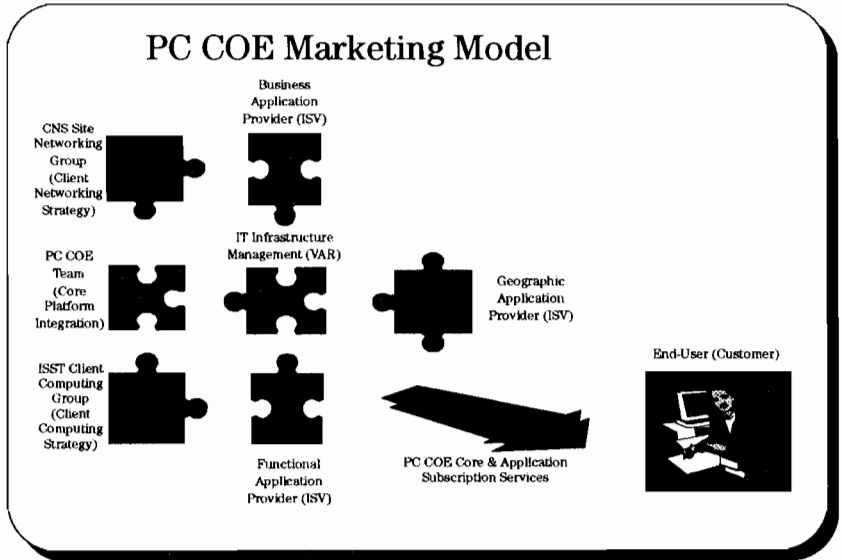
Develop An Application Integration Process

IT organizations throughout the company have taken advantage of the application integration process to deliver everything from office productivity tools to mission-critical, client/server applications. While the common tool set is

distributed to all the users, most applications are needed by a particular business, geography or function within the company.

The PC COE Internal Marketing Model

The focus on the application integration process led to the development of an internal marketing model for PC COE's services. The PC COE processes assemble the pieces of the computing environment and deliver them to the end-user.



Central Service Providers

At the left of the above diagram, three central, platform-oriented organizations are shown. Corporate Network Services (CNS) is the internal organization responsible for managing the world-wide HP Internet. CNS selects the network software and uses the PC COE process to integrate it into the core PC COE platform. Information Systems Services & Technology (ISST) defines the operating system, office productivity and other client computing strategies. The PC COE team manages the integration and delivery processes and engineers the platform configurations.

Application Providers

Around the other edges of the puzzle are the business, functional and geographic application providers. These organizations develop or purchase applications to

solve specific business problems. They use the PC COE process to integrate the applications on top of the core PC COE platform. Application provider organizations exist in each of Hewlett-Packard's major business areas, in each function, in factories and field offices across the world.

Site Information Technology Departments

In the center of the puzzle is the IT infrastructure management organization (site IT). This is the group responsible for managing the local network, supporting the PCs and providing IT services directly to the end-users. The PC COE process delivers the core platform and the appropriate application mix to site IT, who then augments it with any local extensions. The extensions may include additional office productivity tools, local applications and support tools.

End-Users

The ultimate beneficiary of the process is the end-user. By the time the user sees the system, all of the components have been integrated into a robust, reliable system. The various hardware, software and networking components of the computing environment are hidden, and the user can concentrate on the job at hand.

The PC COE Team

The PC COE team is organized to involve the central service providers, application providers and IT organizations in planning and decision making. The core team consists of 8 full-time people and 4.5 "full-time equivalents" (FTE). The core team is funded through a "PC COE fee" charged to all PC COE platform subscribers. The FTE resources are funded by the PC COE fee, but they reside in local IT and application provider organizations. The core team is geographically distributed with members in California, Colorado, Georgia, the United Kingdom, Switzerland and Australia. Efforts are coordinated by weekly phone conferences and bi-monthly face-to-face meetings.

The core team is augmented by an "extended team" of IT and application provider representatives. IT representatives are responsible for local implementation and support of PC COE. Application provider representatives test and integrate their applications with PC COE. Extended team members receive notes from all core team meetings, and they are invited to join the meetings when the agenda is of interest. All extended team members assemble at a semi-annual planning meeting. The results of the planning meeting are used to create a PC COE business plan for the following year. The plan is reviewed by the extended team and key IT managers. This high degree of involvement results in a highly-valued set of services.

PC COE Services

PC COE provides different value propositions for application providers, site IT and end-users. To application providers, PC COE is a predictable target platform and an application integration process. To site IT, it is a way to deliver a complete, integrated, fully-supported set of computing services to the end-user. To the end-user, PC COE provides a robust set of services while hiding the complexity of the technology. PC COE satisfies these three audiences by providing three services:

- ◆ core platform subscription,
- ◆ application integration, and
- ◆ application subscription.

Core Platform Subscription

The PC COE team integrates, tests and delivers the operating system, networking software, Windows and common office productivity tools as a "core platform subscription" service. Users subscribe to the core platform when they first come up on PC COE. From then on, the platform is automatically managed for the user. Some site IT organizations choose to provide only the PC COE core platform subscription to their users, and others prefer to augment the core with additional applications.

Application Integration

Application providers use the application integration process to build on the PC COE core platform. Application integration ensures that the PC COE core platform contains the tools, utilities, APIs and other platform software necessary to run the application. It also facilitates testing to ensure that each application works well both with the core platform itself and with other applications. After integration, each application is distributed to the appropriate server and delivered right to the users.

Application Subscription

Application subscription services provide end-users with applications built on the PC COE core platform. Once a user has subscribed to the PC COE core platform, a range of application subscriptions can be made available. Today, about 30 application providers deliver application subscriptions to PC COE clients, and new providers are signing up every month. Examples include personnel management, configuration/quotation, order processing, strategic business planning and many other applications.

PC COE Processes

Two key processes are used to manage PC COE's services. The core platform engineering process keeps the platform software up-to-date and manages platform configurations. The application integration process makes application subscriptions available to the end-users.

Core Platform Engineering

Core platform engineering begins with a semi-annual planning meeting. Representatives from each application provider and site IT organization are invited. Platform strategy is reviewed, needs are discussed, and deliverables are prioritized. A high-level specification for the next platform release is documented. The high-level specification is then reviewed by two steering committees. One committee focuses on architecture and technology for Hewlett-Packard's computing environment. The other focuses on IT infrastructure management practices. The two committees make their adjustments and approve the specification.

Next an engineering meeting is held to develop the internal specifications for the platform release. The specifications are divided into components. Each component is assigned an owner and one or more engineers. The owner is responsible for the accuracy of the specifications, the completion of the deliverables, and the effectiveness of the component test. The engineers develop code, determine configurations and otherwise integrate the component into the release.

Once the release's basic functionality has been engineered, it moves to its alpha test phase. The alpha test is really a series of prototypes delivered to a group of about 50 testers. Additional functionality is added during alpha, and changes are made based on input from the alpha testers. The alpha phase usually lasts from one to two months. When the release is stable and functionally complete, it moves to the beta phase.

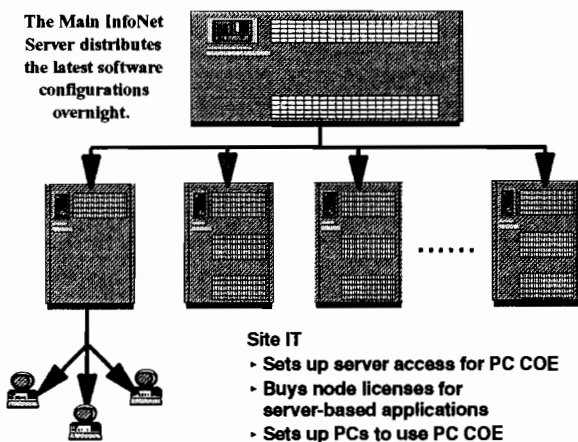
The Beta process automatically delivers the platform release to about 600 beta test users worldwide. The testers subscribe to the beta just as other users subscribe to the core platform. Every application provider must participate in the core platform beta test to allow application integration testing. The beta phase lasts from two to three months. When the number of beta problem reports drops below a predefined level, the beta code is "frozen." No changes are made for two weeks prior to the next phase: pilot.

The pilot phase is a full-scale implementation at a small number of sites. Typically, about 2000 users will receive the platform release during pilot. The purpose of the

pilot phase is to ensure that the distribution and delivery processes are working correctly and to find any problems that only show up in a large-scale implementation. If problems do arise, the PC COE team focuses all of its energies on the pilot sites. Pilot problems are resolved very quickly.

After a two-week pilot, the new platform release is moved to production worldwide. The software is distributed via the HP Internet to 250 servers. Each server is an HP9000 Series 800 running HP-UX and the LAN Manager for HP-UX (LM/X) server software. The distribution process is illustrated below.

PC COE Software Distribution



The new release is then delivered to the client PCs in two steps: a fully-automatic "update," and an "upgrade" that must be enabled by site IT. The automatic update takes place as soon as each user restarts the PC. Hooks are embedded in the AUTOEXEC.BAT file to start the update process. No user interface changes are made during the automatic update since training may need to be scheduled first. Also, any changes with infrastructure dependencies are deferred until the IT-enabled upgrade. For example, new networking client software may require an upgrade to the server software; IT would first upgrade the server software and then enable the client upgrade.

Once the upgrade has been enabled by site IT, each end-user can invoke it through an installation interface on the PC. Alternatively, site IT can "force" the upgrade

by invoking it remotely. Users have about six months to invoke the upgrade, and then it is "forced" worldwide. This process ensures that no more than two PC COE platform versions exist at any one time.

Application Integration and Subscription

Application subscriptions are independent of platform releases. Once an application provider has integrated the application, tested it and distributed it via the HP Internet, the subscription can be made available to the end-users. End-users can subscribe to the application and receive its upgrades at any time.

To integrate an application, the provider first registers in the "PC COE Application Provider Program". The registration process requires the application provider to review the process documentation and to agree to follow its rules. Contacts are identified, and names are added to the distribution lists for platform release information. Beta testers are identified and configured.

Once the application has been registered, integration testing begins. Each application must be tested with the current platform release and re-tested with each new platform release. Integration testing ensures that the application and the platform work well together, but it does not ensure that applications will coexist well with each other. Integration testing of every different combination of applications would be very complex, so the process is simplified through a set of guidelines.

Application providers follow a set of integration guidelines published by the PC COE team and agreed upon by each registered provider. The guidelines identify best practices for managing configurations, define directory structures, and specify common APIs. For example, application providers are encouraged to use the Windows Sockets 1.1 and WIN32 APIs. They are discouraged from modifying LAN Manager and MS DOS configurations directly, and they are discouraged from adding entries to the DOS Path statement. The guidelines need to be updated frequently since the technology is constantly changing.

Once the application has been successfully integrated, it is distributed to the appropriate servers. Application distribution uses the same process as core platform distribution service, but the timing of distribution is independent. An application provider can distribute an application at any time. Once the application is available on the servers, delivery begins.

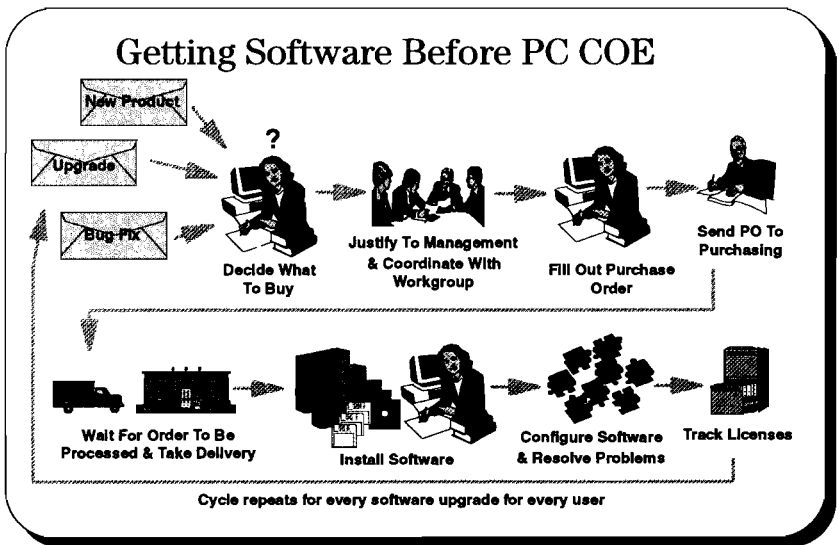
Delivery is the process of installing and configuring software for the appropriate client PCs. Each PC has an installer interface allowing the user to subscribe to applications. If the user is not comfortable with the interface, site IT can also deliver applications from a central interface. Once the user has subscribed, updates can be made available at any time. The application provider distributes a new

version of the software to the server, the site IT server administrator enables the update, and the user is prompted. The user always has the option of accepting the new application update or delaying it until later. The upgrade prompt is repeated each time the user restarts the PC.

Results

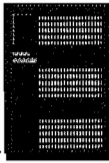
This process of integrating, distributing and delivering the core platform and applications has been very well received. The users like it because they don't have to hassle with their configurations. Application providers like it because it dramatically simplifies the distribution process and their applications have a much greater chance of working correctly on a predictable client platform. Site IT organizations like it because it allows them to offer higher service levels and new technology without increasing costs.

From the end-users' perspective, the software subscription process saves a lot of time and effort, as the following diagram shows.

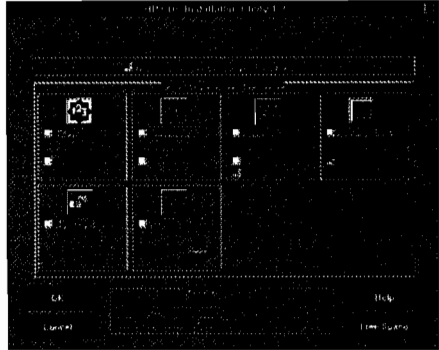
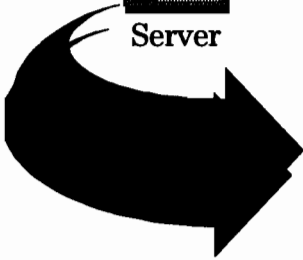


Now the software is delivered automatically via the network. All the user needs to know is a single, simple interface.

Getting Software After PC COE



Server



Additional benefits are described below. The benefits are categorized by the participants in the internal PC COE marketing model:

- ◆ application providers,
- ◆ IT infrastructure management organizations, and
- ◆ end-users.

Benefits to Application Providers

Application providers use Information Technology to solve business problems. PC COE benefits application providers by minimizing the variations of PC platform configurations against which each application must be tested. PC COE also provides hooks and tools to simplify distribution, delivery, installation and update for the applications. Because PC COE provides integration testing processes, the resulting support requirements for the applications are reduced.

Benefits to IT Infrastructure Management

Now that PC COE is nearing full implementation, about 500,000 automated core software component installations and upgrades take place each year. Assuming manual updates would require about one hour each, core upgrades alone save the company's site IT organizations \$25 million per year. Add to that the cost of updating business, geographic and functional applications, and the potential savings rise beyond \$100 million per year. This reduction in software

management cost dramatically increases site IT's ability to deploy and manage software to its PC end-user community.

Benefits to Corporate Information Systems

Without an effective process for managing the PC platform, many Corporate Information Systems strategies, products and services would be difficult to implement. The cost of distribution, maintenance and support for client software would be prohibitive. PC COE provides the means to quickly deploy PC client software in a supportable configuration to the right users at the right time.

Benefits to End-Users

The end-user is the ultimate winner in the PC COE program. The PC COE software management process delivers the right business, functional and geographic set of application software so the end-user can apply technology to solve specific business problems. The PC COE system reliability, hardware/software inventory and license management processes help site IT provide end-users with a reliable, cost-effective platform upon which the applications depend. The following table illustrates potential cost savings. The table assumes support personnel cost \$100K per year and purchase order processing costs \$100. Costs are approximated and are not the actual HP costs.

Cost Driver (per PC per year)	Without PC COE	With PC COE
Licenses (DOS, Windows, office productivity tools)	\$1000	\$300 (based on server-based license sharing and quantity discounts)
Purchasing (order processing)	7.5 (PO's per year assuming avg. 5 PCs per PO) * \$100 (PO processing cost) = \$750	0.02 (PO's per year due to central purchasing) * \$100 (PO processing cost) = \$2
Installation & Update Effort	1 (hour per install/update) * \$50 (cost of support person) * 20 (component upgrades per year) = \$750	[0.1 (hour per automated install/update) * \$50 (per hour cost of support person) * 20 (component upgrades per year)] = \$100
End-User Technical Support	\$2000 (1 support person per 50 employees)	\$500 (1 support person per 200 employees)
Total Cost of PC Ownership	\$4500	\$902

Taking Advantage of PC COE

Any company with significant number of PCs can benefit from a service like PC COE. The following 10 steps provide an initial framework for building a desktop management service:

1. Understand the problem. Count the number of people supporting PCs and compare it to the total number of PCs. Ask help desk personnel where they spend most of their time. If software installation and configuration is high on the list, you're an excellent candidate for PC COE.
2. Formulate a strategy and a set of objectives. For PC COE, the strategy is based on networking and network-based system management. Standardization is important to reduce cost, but over-standardizing will create a lot of resistance. It is critical to create a system that's flexible enough to meet end-users' needs.
3. Sell the strategy to upper management. Desktop management is a new paradigm, and it will impact both IT and end-user organizations. High-level sponsorship lends credibility to the project and helps build confidence during the initial development and implementation.
4. Form a project team to implement the strategy. Organize the team so it is aligned with the business. IT, application providers, business units and geographies should be taken into account. It may be helpful to focus on a particular segment of the business first. Then the service can be expanded after proving its benefit. The key is to involve the organizations who will be impacted by the change. They'll be advocates and supporters instead of critics and adversaries.
5. Evaluate tools and services available from third-parties. When PC COE began three years ago, the computer industry had not yet begun to address the desktop management problem. Now there is a wealth of tools available, and we are evaluating them for our own internal use. Examples include Microsoft's System Administration Server and Symantec's Norton Administrator for Networks. In addition, computer service vendors are beginning to offer desktop management services. HP, for example, has outsourcing services to implement and maintain desktop management services similar to PC COE.
6. Develop and document an architecture for desktop system management. PC COE's architecture is based on distribution to file servers and installation on local PCs. It takes mobile computers into account as well as computers permanently connected to the network. The architecture should

reflect the way PCs are used, and it should be flexible enough to accommodate new technology.

7. Develop a set of services and the processes necessary to provide them. PC COE uses a subscription-based service structure. Companies with fewer PCs or more IT staff may choose to create more centralized "push" services. PC COE separates platform management from application management and helps application providers with integration and testing. Other companies may want to manage all applications as part of a single process. The service structure must be understandable by all the participants, and the processes must accommodate them.
8. Create and execute an internal promotional campaign. Make the benefits clear to end-users, IT organizations and application providers. Provide training to help bring new participants into the process. Provide technical training to familiarize IT and application providers with the desktop management architecture. Provide end-user training on the new software delivery process and any new tools being delivered.
9. Establish support for the environment. The end-user sees the entire computing environment through the desktop, so you'll get blamed for every problem. End-user support should be provided for the entire computing environment, not for individual hardware or software products. We've had excellent results using the HP Response Center's "Help Desk" service to outsource end-user support.
10. Establish an ongoing maintenance process. The process should stress communication among IT, application providers and central service providers. It should facilitate prioritization, planning, tracking and reporting.

Clearly substantial effort is required to build a successful desktop management service. The problem is complex, so simplifying it takes time. HP addressed its PC desktop management problem with PC COE, and the benefits far outweigh the costs.

An Introduction to TCP/IP Network Security

Douglas G. Conorich
Raxco, Inc.
Orem, Utah

The term network security seems to be an oxymoron. The main purpose a host makes a network connection is to increase the access to that host. Conversely, the main purpose of security is to decrease the access to the host. In terms of authorized access to the system networks increase authorized access and security decreases the potential of unauthorized access. Network security balances open access and security. The way you set up security on your system affects, for good and bad, not only your system, but the rest of the network.

Network security protects the host from the network; it does not place security on the network itself. Think of the network as the streets in a city and the hosts as homes in the city. The streets allow the users to travel from their house to any other house in the city. Each house has locks on the doors and windows. The homeowner can determine whom to let in and whom to lock out. Placing security on the network is like placing a road block across the street.

NETWORK SECURITY PLANNING

The first step in developing a network security plan is to assess the threats associated with network connectivity. Request for Comment (RFC) 1244, the Site Security Handbook, identifies unauthorized access, disclosure of information, and denial of services as the main threats from network connectivity.

Unauthorized access is defined as any unauthorized user gaining access to your system. Unauthorized access may come from either inside or outside users. A Wall Street Journal survey reports that insiders cause 86% of deliberate security violations. Insiders have the greatest access to your system and are rarely questioned about their actions. Insiders include current or prior employees and contractors. Unauthorized access may be the result of deliberate snooping, experimentation, or inadvertent disclosure. Whether intentional or incidental, the result is the same; your information has been compromised.

Disclosure of information results from any event that gives unauthorized users access to files or other information. Highly classified information or extremely sensitive information should never be kept on machines directly connected to the Internet. Other information must also be protected from disclosure, such as personnel records, medical information, credit reports, and strategic plans. The

threat associated with disclosure of information and the risk of litigation due to disclosure determines the degree of network security necessary.

Any event that makes it difficult or impossible for a system to continue to perform productive work creates a denial of service problem. Denial of service may impact a few users on a single machine or create a major shutdown of your network. The amount of security necessary for your installation depends on how vital the systems at risk are to the survival of your company.

Passwords

The account password is the single most important security feature on your computer. No matter how well you have secured the file system, the Internet connections, and all the other security features, if you have a guessable or worse yet no password on an account, everything you have done can be circumvented. The Computer Emergency Response Team (CERT) estimates that 80% of all network Therefore, education and regulation of password generation is of paramount importance.

Individual account passwords are intended to restrict access, but their primary function is to verify the identity of the user. Many systems need only the account identification to let the user login, but require a password as additional authentication, or proof of the user's identity. If someone knows your account identification and password, they have the means to impersonate you, and you become accountable for whatever that person does in your name. Therefore, it behooves all system users to learn good password management practices.

An acceptable password meets the following requirements:

- You can remember and type it quickly.
- It cannot be easily guessed.

You want a password that you can comfortably type in and get correct on the first try. If you are not comfortable with it, you will either change it to something you can type, or you will make so many mistakes that your almost-correct passwords will start showing up in security logs. Some simple rules are:

- Never tell anyone your password.
- Never write down a password.
- Avoid the obvious choices.
- Make them work for it.
- Change your password often.
- Use different passwords for different systems.

Most passwords can be guessed by looking at objects within ten feet of the computer (picture of family, pet, or car, a sports poster, etc.). Try to come up with a password that has nothing to do with you personally. This rules out the

name of anyone you know, birthdays and anniversaries, your organization, address, social security and phone number, hometown, and other bits of personal information you may have memorized. As a rule avoid anything that someone could lookup or guess. The obvious choices are usually the first things a password guesser will try.

Communication Protocols

UNIX hosts use two basic protocols to communicate peer-to-peer: Transmission Control Protocol/Internet Protocol (TCP/IP) and User Datagram Protocol (UDP). TCP/IP is a reliable, connection-oriented, byte-stream protocol. Data sent with TCP/IP is reliable, guaranteed to reach the other end of the network correctly. UDP is an unreliable datagram delivery service. The delivery service has no technique to verify the data reached the intended host correctly.

The question is often asked, "If UDP is unreliable and TCP/IP is reliable, why would you want to use UDP?" For short messages, the overhead in insuring a reliable delivery may be greater than the work of re-transmitting the entire message again. Many applications use a query-response model. If a positive response is not received in a specified time frame, the message is re-transmitted. These applications are perfect candidates to use UDP.

The International Standards Organization (ISO) has developed a communications architecture called the Open Systems Interconnect (OSI) Reference Model. This model provides a common reference to discuss data communications. The model consists of seven layers representing the different functions performed during the data communications process. Each layer may contain several different protocols. Each protocol provides a service that is compatible with the definition of that layer. The OSI Reference Model is often referred to as the protocol stack, or simply the stack.

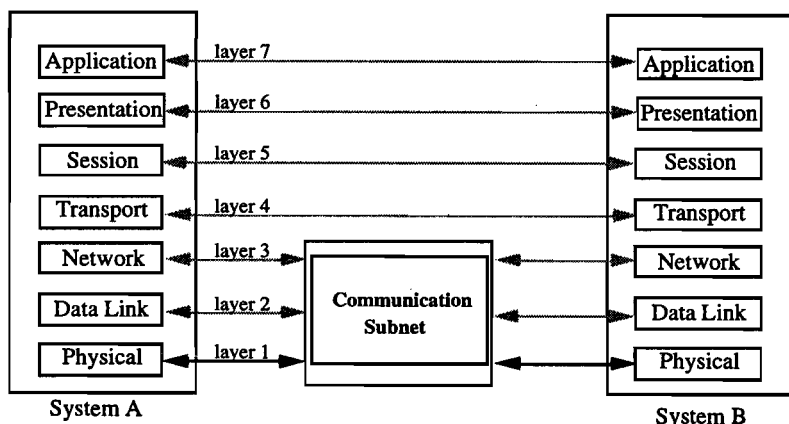


Figure 1. OSI Reference Model

Every protocol communicates peer-to-peer. A peer is the exact same protocol functioning in the same layer on another host. Peer-to-peer communication cares only about the layers in which it is working; therefore those layers must be standardized for successful communications.

The TCP/IP protocol architecture provides the same functionality as the OSI Reference Model, but differs in appearance. The OSI's application and presentation layers are combined into the TCP/IP application layer. The functions of the session and transport layers are handled by the host-to-host transport layer. The network and physical layers correspond to the functions of the Internet and network access layers, respectively. TCP/IP rarely creates a protocol operating in the data link layer. Where necessary, these functions would be handled by existing protocols.

Application Layer consists of applications and processes that use the network.
Host-to-Host Transport Layer provides end-to-end data delivery service.
Internet Layer defines the datagram and handles the routing of data.
network Access Layer consists of routines for accessing physical networks.

Figure 2. The TCP/IP Protocol Architecture.

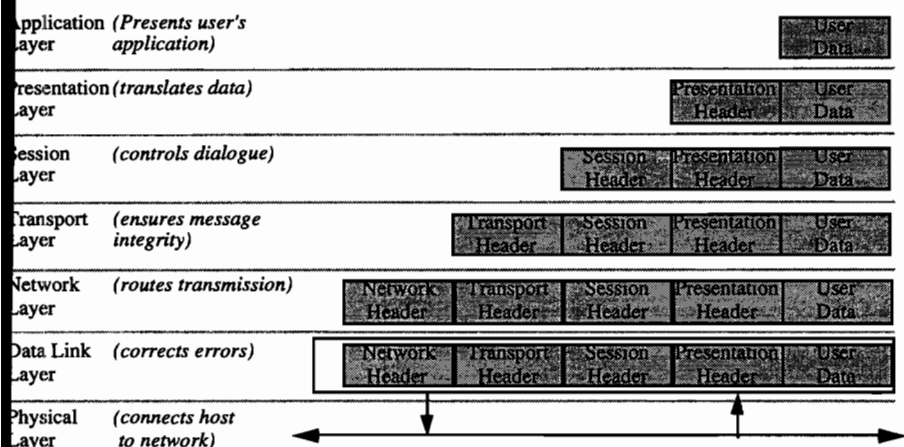


Figure 3 Encapsulation.

As data is sent from the application layer, control information is appended to the beginning of the data by each layer to insure that the data arrives at the proper host and connects to the right protocol. This control information is called a header and the process of appending the control header by each layer is known as encapsulation. When the message arrives at the designated host, the headers are stripped off of the data by the appropriate layer.

The Ethernet

Many networks today operate using ethernet connections. The ethernet is defined by IEEE Standard 802.3. It uses a Carrier Sense Multiple Access/Collision Detection (CSMA/CD) technology and creates an environment very similar to party telephone line. Every station (including PC) has the ability to broadcast and listen to all traffic.. Any system can attempt communication with any other system. Because of the CSMA/CD technology, a station could be set up to continuously broadcast and degrade the network's ability to communicate. The ethernet operates on the data link layer.

Network Security Threats

Network security threats can be classified into three categories:

- Eavesdropping,
- Masquerading, and
- Repudiation.

Other threats like fraudulent data entry, also put at risk the confidentiality, and accountability of the networked computer systems. However, such threats are basically the same whether the system is networked or not.

Eavesdropping is probably the easiest and oldest of all the security threats. Intelligence agencies have employed this technique for decades. Methods of eavesdropping include intercepting radio, microwave, and satellite transmissions, physically tapping wires, and logging into routing hosts.

Radio, microwave, and satellite transmissions were all designed for eavesdropping. A radio would not be much good if you could not tune in to your favorite station. We don't think of this as eavesdropping, but the principle is the same. No matter what you do to secure your transmission (antenna tuning, narrow-beam transmission, etc.) you can receive any transmission given the right equipment.

Physically tapping wires requires physical proximity to the network, therefore, it is harder to do than intercepting radio transmissions. Most networks transfer information via wires, not radio, so more information is available to be intercepted in this manner. Wire tapping does not require that the hacker physically make a connection to the wire. Every wire, even shielded coaxial cables, creates an electromagnetic field around the wire up to a meter away. This field, although weak, can be intercepted in the same way as radio transmissions.

Eavesdroppers use mainly two types of monitoring: Monitoring for message content, and Traffic analysis. Monitoring for message content yields a great wealth of information, but there can be hundreds or thousands of meaningless messages to wade through. Many hackers have the time and desire to do this. Traffic analysis studies message lengths and frequency. Sudden increases in messages between two competitors may indicate an impending merger.

With masquerading the intruder assumes the identity of a valid user or host. The intruder usually starts with an anonymous log-in, such as into a guest account. Once logged-in, the intruder can steal the list of valid users, encrypted passwords, and trusted hosts if these are not properly protected.

If you are the superuser on a system, you can control what your computer is named and what the Internet address is. If an intruder has stolen the list of trusted hosts and valid users, the intruder can rename their system. If a host ever went down, another host could change its network access and use the host's name. An intruder's host pretends to be a valid host and therefore, gaining the valid host's access rights, receives the valid host's messages, can send messages and the valid host, and gains the trusted log-in access of the trusted host.

Repudiation is the refusal of someone to acknowledge or pay a debt. The valid sender can deny sending any message. The valid sender could claim masquerading or a duplicate message. Duplicate messages are known as playback attacks. The intruder can record the valid message and later send the same message through the network. This type of attack even works with

encrypted messages. This is particularly dangerous in a financial transaction (e.g., deny having electronically ordered \$10,000 of goods).

Host Security

Much of network security depends on how well the individual hosts are secured. There are several things you should look for to protect each host:

- Easily guessable passwords—use a password guesser to insure passwords are not obvious. Also, make use of any features within the operating system to force users to choose good passwords. VMS and some UNIX implementations have such features.
- Default accounts—avoid using default accounts (DECNET for VMS and guest for UNIX). If they must be used, use access control lists and file protections to restrict the files that can be accessed. With UNIX, you can also tell the anonymous ftp facility to restrict file access to only certain directories. Make sure the audit trail records as much connection information as possible for default accounts.
- Weakly protected files—all command procedures, programs and scripts should be write protected, especially from users other than the owner. Write protection includes ensuring that a new version of a file can't be substituted for the original. Sensitive files should be read and write protected from general access.
- Proxies and trusted hosts—on UNIX systems restrict the use of the .rhosts files and avoid trusted systems in the hosts.equiv file. You may use proxies with VMS systems, but never with wildcards for the host name or local user name and never for local privileged accounts. Following these guidelines will keep a privileged user on a remote system from automatically being granted privileges on the local system.
- Remote execution—the ability for a user on a remote system to cause something to execute on a local system, without logging in to the local system, should be restricted or removed entirely. On VMS systems this is done by removing the TASK object. On UNIX systems remove the rsh ability. Also, the ability to remotely queue procedures to a local batch queue should be restricted or removed.
- Electronic mail—disable any debug features or other features that would allow a remote sender to execute commands locally.

Firewall Machines

Firewall machines can be used to increase security and decrease outside access to your network. A firewall is a host that separates the internal network from the rest of the world. Firewalls must be setup properly to work.

Characteristics of Firewall Machines

- In order for an internal or external host to connect to the other network they first have to login on the firewall host.
- All electronic mail is sent to the firewall, which in turn distributes it.
- Firewalls should not mount file systems via NFS nor should any of its file systems be mounted.
- Firewalls should not run NIS.
- Only required users should have accounts on the Firewall host.
- The Firewall host should not be trusted nor trust any other host.
- The Firewall host is the only machine with Anonymous FTP.
- Only the minimum service should be enabled on the Firewall in `inetd.conf`.
- All syslogs on the Firewall should log to a separate host.
- Compilers and loaders should be deleted on the Firewall.
- System directories on the Firewall host should be 711 or 511.

Conclusion

One thing is constant in today's computer environment, today's networks are a risk. Low cost solutions are available, but they only mitigate the risk. Cryptography is a fuller solution, but it too is not the full answer. Host security and user education are increasingly essential to any security solution.

The *OSF/Motif™ Style Guide* and You: Developing and Certifying *OSF/Motif™* Applications

Paper No. 6000

Natasha Flaherty
Senior Technical Staff

Oracle Corporation
HP Products Division

500 Oracle Parkway
Box 659408
Redwood Shores, CA 94065
(415) 506-7000

Abstract

The introduction of Graphical User Interfaces (GUIs) ushered in a new era in the realm of software development. With a graphical approach, end-user application developers must think in terms of visual presentation and a consistent use of metaphors when designing a user interface for their application. Suddenly, the way an application looks and behaves is tantamount to the tasks that the software performs.

With all of the features and functionality that a GUI provides, application developers have a multitude of options when designing the user interface of their application. So much so that a user who must learn multiple software applications can easily become confused. To address this problem, a set of guidelines and standards has evolved for each GUI to specify display attributes, input models, navigation, and application actions. This set of behavior specifications and the widgets that are provided to create the user interface help define what is commonly referred to as the "look and feel" of an application.

The *OSF/Motif™ Style Guide* provides a framework of behavior specifications to guide application developers, widget developers, user interface system developers, and window manager developers in the design and implementation of new products consistent with the *OSF/Motif™* user interface. This paper will examine the *OSF/Motif™ Style Guide* and show how its behavior specifications define the "look and feel" of an *OSF/Motif™* application. The *OSF/Motif™* Level One Certification Checklist will be outlined and the certification process for *OSF/Motif™* compliant applications will be discussed. Examples will be drawn from the author's own experience in developing and certifying an *OSF/Motif™* end-user application.

GUI. Say this acronym aloud (pronounced "goo-ey") in casual conversation and you may invoke images of ice cream melting on a hot summer day. However, mention this to someone involved with software development, and most likely they will think of their favorite Graphical User Interface, where point-and-click has revolutionized the way that people interact with computer applications.

The advent of Graphical User Interfaces (GUIs) has opened up a new realm of possibilities for visual presentation in user-centered applications. In previous times, application designers could only assume that users would have a standard block mode terminal, usually 80 columns wide and 24 lines high. Simple applications would have a command-line interface, where users typed their responses to the application prompt line by line. More sophisticated applications might employ a fill-in-the-form interface, where users relied on the TAB, RETURN, arrow keys, and perhaps special function keys to navigate around the screen to various fields, where they entered the appropriate information. These applications, however, were still restricted by the 80x24 screen dimensions and the fixed width character spacing, to name a few things. The hardware characteristics of a character mode terminal do not give application developers much opportunity to create innovative interfaces to their applications.

GUIs have introduced several new concepts in interface design, including: workspaces, modelessness, usage of real-world metaphors, direct manipulation in the object-action paradigm, and application aesthetics.

A workspace is defined in the *OSF/Motif™ Style Guide* as the area on which the windows of a user's environment appear. It is also commonly known as the desktop or root window. Since there is usually only one window in the workspace of a character mode terminal, it is very difficult for a user to work on several tasks simultaneously in a character mode environment. The user cannot bring up two applications side-by-side for comparison and transfer of data, for example. Most character mode applications are not designed for side-by-side operation even within the same program, although some have built features which allow users to "flip" back and forth between screens. With Graphical User Interfaces, however, navigation between windows in order to accomplish various tasks is a simple matter of pointing, or point-and-click, to change the location of the cursor. With system clipboards and primary selection techniques, data can be quickly transferred between applications. And since a workspace allows multiple windows to run simultaneously, users can easily multi-task between different applications. Users may not even be restricted to a single workspace. For example, the Motif-based Hewlett-Packard VUE user environment takes the concept of workspaces one step further by providing six distinct workspaces to help users organize their work more efficiently.

The concept of modelessness becomes much more important when users are running multiple applications simultaneously. A well designed application will avoid overuse of modal structures, i.e., those components which demand a user's immediate response and will not allow other parts of the application or the system

processing to continue. In some cases this may be unavoidable, or it may be the correct thing to do, but the application designer should strive to keep the user in control and able to continue with various tasks while one task is processing.

Metaphors from the everyday world become much more essential in a GUI. The old adage "a picture is worth a thousand words" is especially important in a graphical user environment. For example, if files are to be displayed in a graphical window, it makes sense to represent them with icons that look like paperwork. Similarly, if you group files together in certain directories, users can easily understand the hierarchies if they are shown as file folders containing lots of files and perhaps other folders. In Motif, as in other GUIs, while the application is busy on a certain task which precludes further user input, the cursor should change to a watch or hourglass while the user is in that window to show time passing while the application is working. And perhaps most importantly, widgets in the user environment should mimic the actions of their real-world counterparts. For example, PushButtons should visibly push down when pressed, Sliders should slide up and down in their track, and the Wastebasket is a logical place to dispose of old files. Widget designers should be especially conscious about how well knowledge gained from the real world and from other parts of the interface applies to the operation of their new widget. The more intuitive the widget is, the more natural the user interface will be.

Direct manipulation becomes a very powerful tool in a GUI environment. Users like to feel in control and the object-action paradigm supports this by allowing users to select the objects that they wish to act on, and then choosing to act on them. Whether it's selecting text to be underlined inside a text widget or dragging files to the wastebasket, the user is able to see exactly what will be affected before performing the action. Seeing the change immediately is essential to reinforcing this paradigm. For example, when a user moves the scroll bar knob up and down in a text object, the text should visibly scroll in the window. If a color palette is being used to create colors, users should see the change in colors as the controls are manipulated. Small details such as animated icons can greatly enhance the feedback that an application gives to the user and help reinforce the feeling of control. For example, dragging and dropping a file icon onto a printer icon and seeing the printer icon create an icon for a printed page gives the user immediate feedback that their request for a printout was processed.

Finally, since a GUI environment is by definition graphical, users are now expecting much more in terms of application presentation. An aesthetically pleasing environment is tantamount to a successful application. This not only includes the window itself but the fonts, PushButton labels, and colors used, to name a few items. And even though use of the mouse may allow the user to navigate all over the screen with a point-and-click ease, any application should be well thought out with navigation paths designed in a logical manner.

With the many additional degrees of freedom that a GUI provides to application designers, consistency among applications becomes a key concern. In order for users to become comfortable and proficient with a particular environment,

that environment must be perceived as being orderly and stable, with certain actions producing expected results on a consistent basis. To minimize the learning curve, users should be able to transfer their knowledge of the real world to the application workspace; this helps interfaces seem natural and intuitive. Most importantly, different applications that are built to run under the same type of user interface should behave in a consistent manner, enabling users to apply knowledge gained in one application to a subsequent application. This issue of consistent behavior and conformance to standards is the heart of defining the "look and feel" of a user interface.

The *OSF/Motif™ Style Guide* provides a framework of behavior specifications to guide application developers, widget developers, user interface system developers, and window manager developers in the design and implementation of new products consistent with the OSF/Motif™ user interface. The *Style Guide* assumes by default that applications are written in English and designed for a left-to-right language environment; it notes that many of the guidelines should be modified based on language and scanning direction.

The *OSF/Motif™ Style Guide* is organized into a preface, nine chapters, and two appendices. The preface serves to introduce the OSF/Motif™ model keyboard and mouse conventions. It also defines the support levels that are assigned to each guideline. Currently with Release 1.2 of Motif, all guidelines are supported at the trial-use level. This means that removal of a guideline or a proposed change for any guideline that is not upwardly compatible will be announced and the guideline left unchanged for one full revision of the *OSF/Motif™ Style Guide*. Then, the next version of the *Style Guide* will contain the modification. This is a less protected category than the full-use support level in which incompatible modifications will be announced for two successive revisions before the change will occur. These support levels define the Open Software Foundation's (OSF) commitment to a particular interface definition.

Chapters 1 through 7 of the *Style Guide* cover the following topics: user interface design principles, the input model, the navigation model, the selection model, the activation model, user interface components, and window manager design. These sections are very comprehensive and describe the OSF/Motif™ interface in minute detail. Chapter 8 contains a brief introduction of internationalization and localization concepts which gives a few basic ideas to consider when designing applications for the international marketplace, but should only be considered a starting point. Chapter 9 is a reference chapter for the concepts and components described in earlier chapters of the *Style Guide*.

An important item to note is that "must," "should," and "can" have distinct meanings when used in the *OSF/Motif™ Style Guide*. Guidelines with "must" in them are requirements for *OSF/Motif™ Style Guide* compliance. Guidelines with "should" in them are recommendations which OSF considers important for interapplication consistency. Guidelines with "can" in them indicate optional elements of user interface style.

Appendix A contains some handy tables that show the correspondence between OSF/Motif™ widgets and the various components described in the *OSF/Motif™ Style Guide*. Appendix B is the most useful component of the *Style Guide*. It contains the OSF/Motif™ Level One Certification Checklist, a comprehensive summary of all of the requirements for *OSF/Motif™ Style Guide* compliance. The Checklist is useful as an index to all of the guidelines that "must" be followed in order to achieve *Style Guide* compliance. It includes guidelines from the Preface and Chapters 2 through 7. The sections in the *Style Guide* on User Interface Design Principles and International Market Design do not contain any absolute requirements for *Style Guide* compliance.

Certifying an application as *OSF/Motif™ Style Guide* compliant involves self-review and comparison of your application against the OSF/Motif™ Level One Certification Checklist. To be compliant, you must be able to check "Yes" for each requirement, or "Not Applicable" if that functionality is not present in your application. Checking "Yes" must indicate that that particular requirement is satisfied in all areas of your application. The sections that describe the Navigation and Selection models are particularly detailed and time consuming and when you consider how many windows and navigation paths your application has, this can be a monumental task.

Although certification may seem a daunting task, in comparison to guidelines established for the Microsoft Windows™ and Macintosh™ user interfaces, the *OSF/Motif™ Style Guide* provides the most comprehensive interface description and verification process. By scrupulously following the *Style Guide* and using OSF/Motif™ widgets in your design, you can be sure that your application is *OSF/Motif™ Style Guide* compliant.

Bibliography

- Apple Computer, Inc. *Macintosh Human Interface Guidelines*. California: Addison-Wesley Publishing Co. 1992.
- Microsoft Corporation. *The Windows™ Interface. An Application Design Guide*. 1992.
- Open Software Foundation. *OSF/Motif™ Style Guide*. Revision 1.2. New Jersey: PTR Prentice-Hall, Inc. 1993

Evolving ODBMS Standards

Paper number #6001

By

Dr. Andrew E. Wade

**Objectivity, Inc.
301B East Evelyn Avenue
Mountain View, CA 94041
415.254.7100
drew@objy.com**

for

Interex '94

September, 1994

Why an Object Database Standard?

The computing industry has changed over the past couple of decades. Where a typical user formerly relied on a major system vendor such as IBM to supply their full spectrum of needs, today's users demand the freedom to choose vendors of both hardware and software, not only for the best performance and functionality, but for competitive pricing. Instead of being held hostage by a single system vendor, the user may now choose among vendors, and independently mix together hardware and software components much as one might mix stereo system components. This has driven the downsizing or rightsizing effort from the mainframes with their proprietary operating system, databases, and applications, to the open systems, with standard operating systems such as Unix and Windows, standard database interfaces such as SQL, and applications that run on any standard platforms.

Users with information that is large, requiring management to and from secondary storage, is shared among multiple users, or requires guaranteed integrity, need a database management system to provide these services. The same needs for flexibility to choose vendors and competitive pricing drive the needs for standard interfaces to such DBMSs, allowing users to choose the best solution.

Database systems have evolved over the past couple of decades from indexed files systems (ISAM, etc.), to networked and hierarchical systems (CODASYL), then relational (RDBMS), and now object (ODBMS). Each has met the needs of a class of users, and each new generation has allowed new types of users to benefit from DBMS services. With each previous generation, standards evolved; e.g., for RDBMSs, IBM proposed the SQL standard query language. In the ODBMS world, however, no such standard existed, so the ODBMS vendors joined together to create one. This standard, ODMG-93, now published, provides a single common application interface, and its support is committed by all vendors, making it a *de facto* standard.

Object Databases

ODBMSs have emerged to meet the needs of a new class of applications. While RDBMSs support typical simple business applications with simple spreadsheet-like data and simple processing, many new applications go beyond this to new areas, requiring new DBMS support. Three types of applications come to ODBMSs:

- Object users
- Complex and interconnected information
- Distributed and heterogeneous environments

The first group has chosen object technology (see side bar). Before, software was built in the style of medieval craftsman, each component

hand constructed from scratch. As Winchester showed with rifles, and Ford with automobiles, use of standard components, built to be re-used, allows much higher production levels as well as higher quality.

Object encapsulation and abstraction allow management of higher-level constructs, mapping directly to the application entities, hiding the internals, so changes can be made without breaking large systems. Extensibility allows adding new classes and objects so that systems can grow. Re-use allows leveraging previously built and tested objects, changing only what is necessary for their new application. These result in the benefits of shorter development time, after the learning curve is passed, significantly reduced maintenance costs over product lifetime, and higher quality due to re-use of already-tested objects.

Object Technology Benefits

- Short Development Time (after learning curve)
- Decreased Maintenance Cost
- Higher Quality

Object Technology Terminology

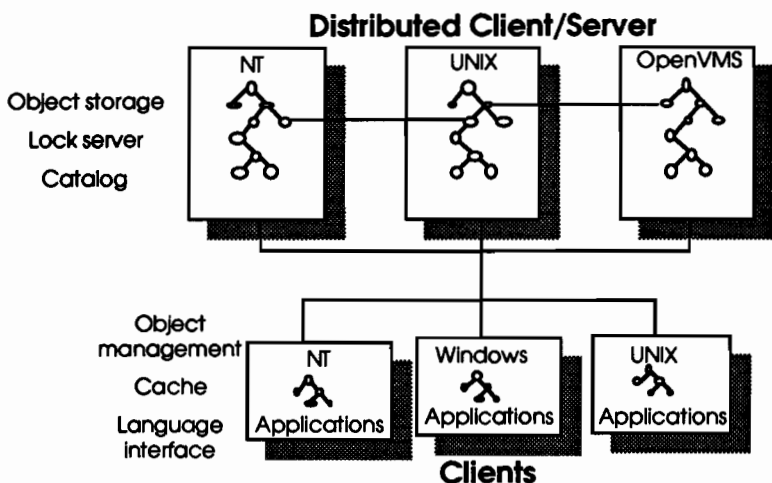
- **Identity:** Unique addressability of an object independent of its location and state; e.g., the object representing a person is the same even if that person changes social security number or hair color.
- **Encapsulation:** Combining together code and data into a single unit, the object, accessible only through well-defined external interfaces.
- **Abstraction:** High-level objects, often built from other objects, that directly represent application entities.
- **Class:** A common set of object characteristics, including behavior, structure, and relationships.
- **Inheritance:** Adding new classes, based on changes to existing classes.
- **Composition:** Constructing a new object instance by connecting other object instances from the same or, often, different classes.
- **Polymorphism:** Invoking a behavior of an object for which the actual code executed is chosen dynamically at run-time depending on the object; e.g., "draw" might work equally well, though via different implementation, for a spreadsheet, graphics, or text.

Object technology applies to all aspects of software. For a user of objects in analysis and design, in the graphical user interface (GUI), with the language, debug tools, even operating system, it is easier and faster to use objects directly the DBMS, rather than translating to records or tuples.

The second application need driving users to ODBMSs is managing new kinds of information. To contrast, if the information is simple, fixed length, fits nicely into rows and columns and tables, then it works well with RDBMSs, because that's what they store. Similarly, if the operations are simple joins, projects, and selects, or simple on-line transaction processing (OLTP), RDBMSs do that well. However, for arbitrary information such as images, graphics, audio, video, and dynamically changing information such as time series data, it is very difficult to pack into relational tables. In an ODBMS this information is represented directly and efficiently.

Similarly, if the information contains many interconnections, RDBMSs become quite slow due to joins. In any ODBMS such interconnections can be represented directly and efficiently, sometimes giving orders of magnitude better performance.

The third drive towards object databases comes from a distributed environment of heterogeneous hardware, operating systems, networks, and languages. Most traditional (RDBMS and some ODBMS, too) database systems are built around a central server, which, just like the mainframe architecture of the 1960s, treats users as simple remote terminals, with all processing and storage on the server, creating a bottleneck. If the server becomes inadequate, the only option the user has is to replace it. Meanwhile, desktop workstations have evolved to the point where the range in processing power from the desktop to the server is only 3 or 4 times. An ODBMS can take advantage of this by distributing the objects among databases on all the different platforms, with objects and users anywhere, and transparent access to objects. Users may buy the latest price-performance leading computer, slide it next to their existing computers, move some of the objects to that new computer, immediately taking advantage of it, without disrupting any of the applications or users. This Distributed Client/Server architecture extends the first-generation mainframe-like client server to a new level of flexibility and performance.

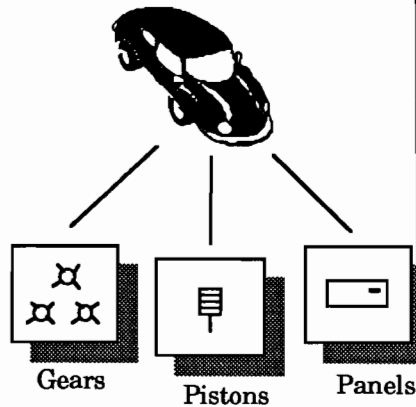


In short, an ODBMS gives the benefits of traditional DBMSs (persistence, recovery, multi-language and query support, integrity maintenance, concurrency, etc.) as well as the benefits of object technology, support for complex and interconnected information, and transparent distribution over heterogeneous environments.

A simple analogy helps illustrate the advantages of an ODBMS for complex information. Suppose you wish to store your car in the garage at the end of the day. In an ODBMS, the car is an object, the garage is an object, and there is a single operation to store the car, and it's done. In a RDBMS, since each type of data is stored in its own table, the car must be disassembled, flattened, storing the gears in the gear table, the pistons in the piston table, and so on for wheels, panels, etc. In the morning, before going to work, you must re-assemble the car from these flattened parts. Now, there is an advantage to such flattening: you can just as easily assemble it into a motorcycle. However, unless you expect to do that often, the process of disassembly and reassembly is time consuming for the developer, error prone, and very slow at runtime. The ODBMS, instead, directly accepts the high-level car object, and understands that it is composed of many other, connected objects, representing the frame, engine, wheels, etc., even when some of these components are shared with other cars.

Object DBMS: An Analogy

- ▲ Store car in garage
- ▲ RDBMS
 - Decompose car into gears, pistons, slide-panels, etc.
 - Store each type in a bin (table)
 - Re-assemble in morning (flexible - could re-assemble into motorcycle . . .)
- ▲ ODBMS
 - One operation: store car in garage



Users of object databases are found in a variety of industries, including telecommunications, manufacturing and process control, document management, financial analysis, multi-media, and transportation and logistics. Over 50,000 end user license are deployed in production use, including 24hr/day fault-tolerant and real-time telecommunication applications, high-end multi-media systems, and process control.

Example Users of ODBMSs

- | | |
|---------------------------|---------------------|
| • Kintetsu: | Transportation |
| • Bell Northern Research: | Repository |
| • Digital Equipment: | CAD/CAM |
| • Osaka Gas: | Process Control |
| • Digital Switch: | Real-time telecom |
| • British Telecom: | Maintenance system |
| • Siemens: | Telecom |
| • Sybase GainMomentum: | Multimedia |
| • QuickTurn: | Hardware emulation |
| • Mead Data, Lexus: | Document management |
| • CitiCorp: | Trading system |
| • Cimplex: | CIM |

ODMG

Understanding the market need for portability, the ODBMS vendors, in June, 1991, formed The Object Database Management Group (ODMG) to cooperatively create one standard interface they could all support, and thus provide application portability. Modelling their organization after the SQL Access Group (SQG), they hoped to both accelerate the adoption of a standard, and encourage commonalty among all the various organizations, standards groups, and consortia who need to interface to ODBMSs. Published in September, 1993 [see box], the ODMG-93 standard has now been adopted by the Object Management Group (OMG) as one of the levels of protocol in its Persistent Object Service. All voting members must contractually commit to supporting this standard, and the current membership includes virtually the entire industry.

ODMG Membership

- **Voting members:** Objectivity, Object Design, Ontos, O2, Poet, Servio, Versant
- **Reviewer members:** Hewlett-Packard, Intellitic, Itasca, MICRAM, Persistence, Sybase, Texas Instruments
- **Executive Director:** Doug Barry
- **Chair:** Rick Cattell
- **Contact:**
13504 Clinton Place
Burnsville, MN 55337
(612) 953-7250
(612) 397-7203 fax
info@odmg.org for information and membership
question@odmg.org for questions
proposal@odmg.org for proposals
- **Book**
R. Cattell, Ed, The Object Database Standard: ODMG-93, ISBN 1-55860-302-6, Morgan Kaufmann Publishers, San Francisco, California, 1993. (800) 745-7323. In Europe, available from Afterhurst, +44-273-207259, +44-273-205612 fax

ODMG-93

The ODMG-93 specification includes all the capabilities necessary to build a complete, production application, which will execute, with re-compilation, on any compliant ODBMS. Basic functionality includes:

- Define, Create, Modify, Share objects
- Transactions
- Relationships
- Collections

It does not yet include some advanced technology, such as versioning, long transactions, and meta-data access, which will be considered in the future.

The specification is delivered with the following components:

- Object Model, a superset of OMG's
- Object Definition Language (ODL), a superset of OMG's IDL
- Object Query Language (OQL), based on SQL SELECT
- C++ binding
- Smalltalk binding

Object Model

The Object Model is based on OMG's Core Object Model, adding components to fill out the profile for ODBMS use. As in OMG, it describes an object as having identity, well-defined behaviors, and abstract state accessed through such behaviors. ODMG adds to this the specification of inter-object relationships, including bi-directional integrity, and collections, including sets, lists, bags, and varying-sized arrays. It also adds the DBMS concepts of transaction for maintaining integrity of objects in a distributed environment of concurrent users.

ODL

The Object Definition Language provides a concrete, programming-language-independent syntax to define object classes, including state, behavior, and relationships. This achieves schema interoperability, allowing users to build systems that share objects among different languages. It is a strict superset of the OMG IDL, allowing cooperation with CORBA systems, and adding syntax for defining bi-directional relationships, collections, keys, and extents.

The following figure shows how a user can employ ODL to build an interoperable schema, automatically generate the metadata, header files, and methods required, and then build applications in multiple, standard languages.

Using ODMG-93

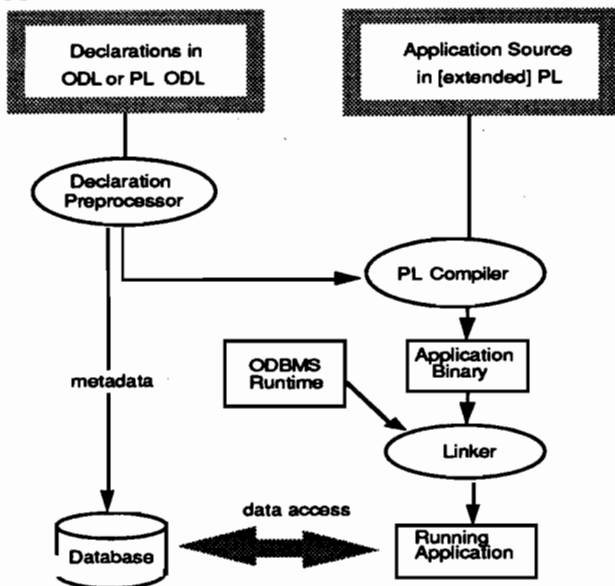


Figure 1-2: Using an ODBMS

An example of ODL shows how a bi-directional one-to-many relationship is declared between a professor and his students.

ODMG '93: ODL Example

```
interface professor : employee {
    attribute string <32> name;
    unique attribute lang unsigned ssn;
    relationship dept works_in inverse faculty;
    relationship set <section> teaches inverse taught_by;
    . . . operations . . .
}

interface section : class {
    . . .
    taught_by: professor . . . ;
    . . .
}
```

OQL

The Object Query language provides predicate query functionality. Based on the well-established SQL standard's SELECT statement, it supports the traditional constructs such as tuples and joins, and adds support for accessing arbitrary objects, identity, inheritance, navigation of relationships, collections, and methods. With OQL simple high level declarative statements produce ad hoc queries against arbitrarily complex objects.

ODMG '93: OQL Examples

- `Select x from x in faculty
 where x.salary >x.dept.chair.salary`
- `sort s in
 (select struct (name: x.name, s:x.ssn)
 from x in faculty
 where for all y in x.advisees:y.age<25)
 by s.name`
- `Chair.salary`
- `Students except TAs`
- `list (1,2) + list (count (jse.advisees), 1+2)`
- `exists x in faculty [1:n]: x.spouse.age<25`

C++

C++ has emerged as the most widely-used object language, so ODMG-93 includes a tight binding that allows C++ programmers to use objects naturally while the ODBMS transparently manages the objects. The ODMG approach is based on the following key design concepts:

- Standard Language Compilers
- Persistence-Capability via Inheritance
- Mixed Transient and Persistent Instances via `new()`
- Relationships via Smart Pointers

Rather than requiring proprietary language syntax and semantics, and proprietary compilers, ODMG works with off-the-shelf compilers and tools. It achieves language integration via the mechanisms C++ provides for extension: class libraries and operator overloading. The behavior of persistence capability is applied via the standard object technology approach of inheritance, giving the user the flexibility to define transient-only classes, when desired. By overloading the new() operator, both transient and persistent instances can be created at runtime from the same persistent-capable classes, providing maximum flexibility. Inter-object relationships are declared via a Ref<> template class, and overloading of the dereference operator -> allows their use, through smart pointers, just like normal pointers. This gives the natural integration desired, but allows implementors the freedom to implement systems differently, either with raw pointers or with protected smart-pointers, and also allows high-performance implementations. Some examples follow.

ODMG '93: C++ Example

```
class Professor: Employee {
    long ssn;
    char* name;
    int age;
    Ref<Department>dept inverse faculty;
    Set<Section> teaches inverse taught_by;
    . . .
    void grant_tenure()
    void assign_course(section)
}

. . .
Ref<Professor>prof;
. . .
prof = new(db, Professor);
prof->name="Smith"
prof->age+prof->age+1;
```

Smalltalk

Smalltalk is the other popular object language, preferred by some for its purity, support of dynamic use, and rich GUI classes. ODMG-93 similarly supports Smalltalk with a tight binding that allows Smalltalk programmers to use objects naturally and receive the

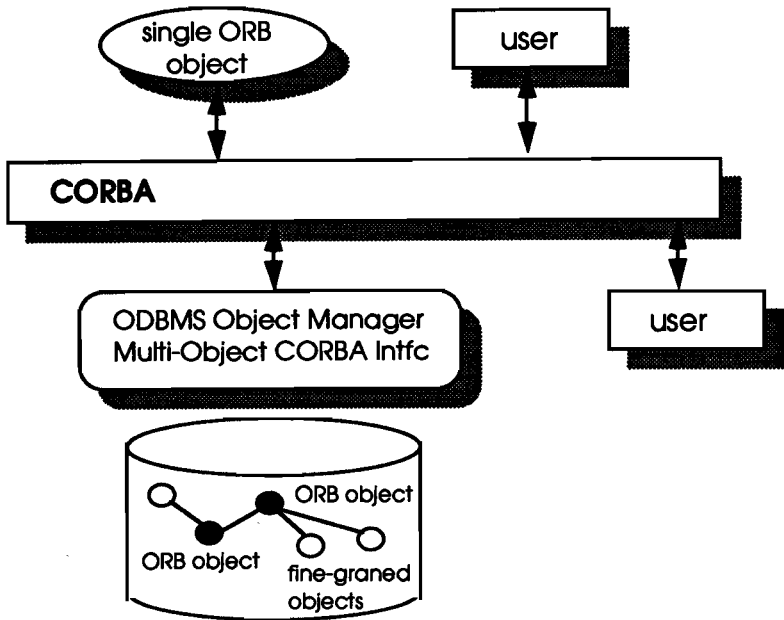
benefits of ODBMSs. It supports the same object model and ODL, allowing the same objects to be shared by Smalltalk and C++ users and queried by OQL. It supports the Smalltalk type system and syntax, allows untyped fields, and works with the single inheritance of Smalltalk.

Relation to Other Standards

Many other standards relate to or involve the use of object databases; e.g., STEP, CFI, TINA-C, ISO ODP, ANSI X3H7, OpenGIS, etc. Before, any such systems had to deal with the mapping of their standard to a half-dozen different, proprietary ODBMS interfaces. Now, with ODMG, they can do one mapping to ODMG-93, and immediately support any ODBMS. Two other standards are worth more detailed description: OMG and SQL.

OMG is the leading consortium addressing interoperable object standards. In fact, ODMG members began within OMG, but wanted to accelerate the progress towards an ODBMS standard. OMG began by producing the Common Object Request Broker Architecture (CORBA), a means to dispatch operation requests among objects and users, and more recently has begun to add some object services. Since OMG, by charter, does not create interfaces, but rather issues Requests for Proposals (RFPs) of existing products and then chooses one, ODMG members foresaw a future where OMG would be choosing between the half-dozen different proprietary ODBMS interfaces. Far better would be to have all vendors agree on one interface, with the best of each system, and jointly support it. Now accomplished, this ODMG interface has been adopted within OMG's Persistent Object Service, allowing users of OMG CORBA-based systems to take advantage of ODBMSs which may contain mixtures of OMG objects, accessible like any other OMG objects, along with millions of fine-grained objects, accessible directly through ODMG interfaces.

ODMG ODBMS with OMG CORBA



In the area of declarative query SQL is the accepted standard. In order to leverage the expertise of the many users and systems built on SQL, ODMG's OQL began with its predicate query statement, **SELECT**, and then added object functionality. ANSI X3H2 is now pursuing SQL3, which will address object functionality as well as adding a computationally complete programming language. Unlike ODMG, it does not plan to provide an ODL or bindings to object languages like C++ and Smalltalk, which are so important to ODBMS users. However, its work in query support of objects will likely overlap the ODMG OQL. Today, SQL3 is in its very early stages, considering basic questions of how to model objects. ODMG decided to address the immediate needs among its users for basic object query, and hopes to feed this work and experience into the SQL3 development. Eventually, when SQL3 is available, hopefully supporting the OMG and ODMG object models, ODMG will likely do a mapping to the computationally complete SQL3 language, just as it has done today for C++ and Smalltalk.

What next?

ODMG is continuing its work in three areas: implementation feedback, extension of the standard, and dissemination.

The earliest product support of parts of the standard (ODL and OQL) are already available (respectively, from Micram/Objectivity and O2), and all vendors are working on implementation. Based on this work, and on feedback from users, fine-tuning of the specification is expected. In fact, when the publisher ran out of print last February and requested a reprint, ODMG provided a V1.1 update.

ODMG-93 is a first step. Although it provides all basic capabilities, and more than other first standards such as SQL1, there is much more that could be added. Some possible future areas includes:

- Dynamic meta-data access
- Versioning
- Long transactions and persistent locks
- More language bindings

By continuing its process of small working groups of key technologists, and building on experience of actual implementations and users, ODMG plans to evolve and extend the specification over time, but always supporting the basic ODMG-93 interfaces.

In its first couple of years, ODMG consisted mostly of concentrated technical meetings to produce the specification. Now that ODMG-93 is released, ODMG is expanding its efforts to make its work known to groups of potential users and with liaisons to related standards organizations and consortia. Now incorporated as a non-profit organization, with open membership rules, ODMG has grown significantly in membership and visibility. As implementations emerge over the next year, ODMG-93 will take the object database industry to a new level of maturity and capability.

Paper #6002

ABCs of ODBC

by

Dirk Huizenga

Dynamic Information Systems Corporation

5733 Central Avenue

Boulder, Colorado 80301

303-444-4000

**ABCs of ODBC
6002-1**

ABCs of ODBC

The purpose of this presentation is to provide exposure to the ODBC as a standard for accessing data. It covers the components that are provided with the Microsoft® Open Database Connectivity™ Software Development Kit version 2.0 (ODBC SDK).

History of ODBC

Structured Query Language (SQL) has become a standard for creating, manipulating, and managing relational databases. By itself, SQL is not a complete programming language. Normally it is used in conjunction with another language such as COBOL, C, or fourth-generation languages. SQL was first standardized in 1986 by the American National Standards Institute (ANSI). The earlier standards provided several programmatic interfaces of which the embedded SQL proved the most popular.

Embedded SQL defined in the ANSI 1989 standard only supported static SQL. Static SQL allowed for using host variables to transfer data from the SQL request to the host language at run-time. The rest of a static SQL statement must be defined at compile time and link to a particular DBMS. Although this is efficient, it is not flexible enough for client-server applications doing ad hoc request.

The most recent SQL ANSI specification defined in 1992 contains several new features including connections to database environments, scrollable cursors, and dynamic SQL. This allows an application to create SQL statements and retrieve data without knowledge of the database structure prior to compilation.

The Microsoft Open Database Connectivity (ODBC) is a standard based on the X/Open and SQL Access Group Call Level Interface specification. It has been heavily promoted by Microsoft to provide a standard way for applications to access data. Applications can be written without a specific database management system (DBMS) in mind.

Installation

Installing the ODBC SDK version 2.0 will provide you with a set of tools, for developing either ODBC drivers or ODBC-enabled applications. This version offers support for 16-bit (Windows 3.1) and 32-bit applications (WIN32s™ and Windows NT). A new enhancement is the driver-independent Cursor Library for support of scrollable cursors. A set of sample ODBC drivers for accessing Access®, Btrieve®, dBase®, Excel®, FoxPro®, Paradox®, and text files are provided to help you develop ODBC applications. Sample code is provided to aid in developing drivers, Visual

Basic® and C-language applications. Utilities are provided to test drivers (ODBC Test) and debug drivers and applications (ODBC Spy).

Accessing Data

When an application has been written to access data using ODBC the source of the data is limited only by the availability of the appropriate driver. How flexible the application is at accessing data from the ODBC source depends on how the application was designed. Microsoft has enhanced Excel to use ODBC data sources to retrieve external data.

Microsoft Excel can access external data in any of the following ways: using the PivotTable Wizard to create a pivot table with results; using the Microsoft ODBC Add-in to directly access ODBC data sources; using Microsoft Query application access ODBC data sources and to return data to Microsoft Excel; and using the Query Add-in to access Query from within Microsoft Excel. On the Apple Macintosh, Microsoft Query data access is provided through the Data Access Language (DAL) ODBC driver. To use Query and the Apple DAL ODBC driver requires Macintosh System 7 or later.

Programming with ODBC

The ODBC programming interface defines:

- ◆ SQL grammar based on the X/Open and SQL Access Group SQL CAE 1992 specification.
- ◆ A library of function calls that allow applications to connect to a data source, execute SQL statements, and retrieve results.
- ◆ A standard way to connect to a DBMS.
- ◆ Standard data types.
- ◆ A standard set of error codes.

The components of the ODBC architecture are the application, the Driver Manager, the driver, and the data source. The Driver Manager maps the data source name to a specific driver dynamic-link library (DLL) and loads that driver. The driver implements ODBC functions and interacts with a specific data source. The data source is a specific DBMS and the network and platform operating system needed to access it. ODBC defines two types of drivers: a single-tier where the driver access the data source directly and a multiple-tier driver where the components of the architecture reside on two or more platforms.

ODBC defines conformance levels for drivers in two areas: the ODBC API and the ODBC SQL syntax. Applications can determine the levels supported by a driver by calling several information functions. Three levels are defined for the **ODBC API conformance**: the **Core** (X/Open and SAG CLI specification); **Level 1** includes the Core API and send and retrieve values in multiple calls and more flexibility interacting with data sources; and **Level 2** includes the Core, Level 1 API, and scrollable cursor, arrays of parameter and result values, lists of data sources. **SQL syntax conformance** is also divided into three levels: the **Minimum SQL** grammar supports character types and simple expressions; **Core SQL** grammar (X/Open and SAG CLI specification) includes the Minimum grammar, most C-language data types, more complex expressions, and set functions; and **Extended SQL** grammar supports Minimum, Core SQL grammar, BINARY and DATE data types, batch SQL statements, procedure calls, outer joins, positioned UPDATE and DELETE, SELECT FOR UPDATE, and unions.

Programming an ODBC client uses a Call Level Interface (CLI) that support SQL statements. A CLI doesn't require a precompiler as does Embedded SQL. To submit an SQL statement using the ODBC CLI, the SQL command is placed in a text variable and passed as a parameter in a function call. Errors are detected by a function return code. Once an error has been detected further information can be obtained by calling an error function.

An application using ODBC:

- ◆ Obtains an environment handle (HENV) and initializes ODBC.
- ◆ Requests a connection handle (HDBC) and connects to the data source. An application can maintain multiple connections.
- ◆ Process one or more SQL statements after obtaining a statement handle (HSTMT).
 - ◇ The SQL text is placed in a buffer.
 - ◇ If the statement contains variable arguments, the values are bound to the statement.
 - ◇ If the statement returns results, the application can assign a cursor name.
 - ◇ Submit the statement for immediate or prepared execution.
 - ◇ Check for error or results information.
 - ◇ If appropriate, assign storage and fetch results.

- ◆ Complete each transaction by committing it or rolling it back.
- ◆ Disconnect from the data source.

Testing ODBC Applications

ODBC Spy is a utility included with the ODBC SDK to debug drivers and applications. With ODBC Spy you can spy on an application-driver connection and save the ODBC calls to a log file. You can use this log file information to emulate either a driver or an application. Emulating a driver can be useful when the data source is a remote connection (multiple-tier driver). Emulating an application can be useful for testing drivers from different sources. The ODBC Test utility can also be used to test driver conformance.

Conclusion

The Open Database Connectivity Software Development Kit provides a standard way to write applications that are data source independent on Windows 3.1, Windows NT, and the Apple Macintosh (System 7). This results in applications that can be configured for different platforms and database management systems at installation or even at run-time. An application can make use of data sources that did not exist when it was developed. Because it uses industry standard SQL an application written using ODBC has maximum flexibility and life.

Porting Qedit from MPE to HP-UX

Paper #6003

By David J. Greer

Robelle Consulting Ltd.
Unit 201, 15399-102A Ave.
Surrey, B.C. Canada V3R 7K1
Phone: (604) 582-1700
Fax: (604) 582-1799
E-mail: david_greer@robelle.com



Introduction

Our Qedit full-screen text editor was first written in 1977 and has been continually enhanced since then. We recently ported Qedit from MPE to HP-UX. This paper discusses our research strategy and the technical details of how we actually managed to port approximately 100,000 lines of code that had been written over the last fifteen years.

Original UNIX Goals

For a few years, our customers (especially in Europe) had started doing development on UNIX machines. We didn't know anything about UNIX, but we decided that we had better start learning. Our goal was to learn as much as we could about UNIX by seeing how much of our source code and programs we could convert to HP's version of UNIX: HP-UX.

Why HP-UX

We decided on HP-UX for a couple of reasons:

1. It was made by HP, so it was likely to be more familiar to us than another vendor's UNIX (and at least the Response Center telephone number would be the same). Time and again, this proved to be true and greatly helped our development effort.

2. Qedit is written almost entirely in SPL/SPLash!. Because of our compiler technology, we wanted to focus on the HPPA RISC architecture. For this reason, we restricted our research to HP's 700 and 800 series of computers (they have HPPA RISC CPUs).
3. We did not want to rewrite all of the source code into C (the usual language of choice for UNIX programs). We felt that rewriting Qedit into C would be too large an R&D project for Robelle. Automated tools for doing the conversion were reported to be ineffective.

Major R&D Phases

Our HP-UX research was done in different phases. We had a short-term goal for each phase and worked towards completing this goal in a reasonable time frame. The major phases were as follows:

1. **Basic Research.** In this phase we borrowed an HP-UX machine and investigated HP-UX object-code and run-time libraries.
2. **MPE and Robelle Libraries.** Robelle's software development is based on a concept of layers that build on each other. This phase involved getting our own machine, writing necessary MPE replacement routines, and porting our fundamental subroutine libraries.
3. **Rebuild Qedit.** Once the basic libraries were running we needed to look at Qedit. In this phase, we reengineered Qedit so that it has little knowledge about which computer platform it is running on.
4. **Polish and Release Qedit/UX.** After the first version of Qedit was working on HP-UX, we needed to do a series of enhancements to make Qedit work in a more UNIX-like manner. We also needed to complete all the details that turn an R&D project into a product.

Personnel

Three people worked on porting Qedit to HP-UX.

Dave Lo Dave was the general programmer's helper. He was assigned specific research tasks (e.g., how to get the value of an environment variable in SPLash!).

Robert Green Bob is the original author and chief architect of Qedit. His main job was to keep Dave's and David's spirits up and to reengineer Qedit so that it didn't depend on MPE any more.

port object-code to HP-UX. In order to port Qedit from MPE to HP-UX, we would first need to port many of our own libraries and investigate how many MPE library routines were available on HP-UX.

Phase One: Basic Research

We wanted to start our research by doing two things:

1. Learn more about HP-UX (e.g., how to log on!)
2. Prove whether or not the object-code formats on MPE and HP-UX were the same.

Get a Machine

You can't make an omelet without breaking eggs, and it's hard to learn about HP-UX without having access to a machine. Because we had no idea how successful our initial research would be, we didn't want to purchase one.

We asked HP if we could rent an HP-UX machine for a week, but HP doesn't have short-term rentals. Since we've had good relations with HP/Canada, they arranged for us to have one of their demo machines for a month. It's a good thing we had the machine that length of time, because it took us about three weeks to get it up and running.

We could tell HP-UX was made by HP. When we first brought the machine up, one of the first things it did was ask us if our console was a 2392 or a 700/92. We can't imagine any non-HP machine asking this question.

Networking

We learned a lot by just installing the borrowed HP-UX machine and getting it connected to our network. On our HP 3000 machines, we had HP's NS software and used Thinlan to connect all of our machines together. We lost a lot of time because the demo HP-UX machine had a jumper switch on the network interface card turned off. It took us two weeks to figure out that it was a hardware problem and not a configuration problem.

By default, HP-UX is not able to communicate with MPE machines. On MPE, we are used to having front-end software that handles configuration (NMMGR being the one for configuring network software in MPE). In UNIX, it's common for configuration to be done by changing scripts (although some common configuration tasks can be done with HP's SAM tool). Scripts are similar to MPE/iX command files. One of the files that is used to configure networking is called `/etc/netlinkrc`. By default, it contains a line like this:

```
/etc/lanconfig lan0 ether
```

For HP-UX to communicate with MPE machines in the Thinlan/NS environment, this line needs to be:

```
/etc/lanconfig lan0 ether ieee
```

Notice the addition of "ieee" to the end of the line. We also needed to the following:

1. Fill in the DOMAIN and ORGANIZATION in /etc/netlinkrc.
2. Add machine names and IP addresses to /etc/hosts.
3. Reboot our machine for these changes to take effect.

Since our machine had the NS software for HP-UX, we could now use vt3k to log on to our MPE machines from HP-UX (vt3k provides virtual terminal access from HP-UX to MPE). After using NMMGR to add our new HP-UX machine to our MPE configuration, we could also use dscopy to copy files from MPE to HP-UX and vice versa.

Object-Code Format

Our next major goal was to test our theory that object-code files were the same on MPE/iX and HP-UX. To show this was the case we planned to compile a SPLash! program on MPE, copy the object-code to HP-UX, link the object-code, and run the resulting program.

One of the first programs we tried was written by Randy Medd of Telamon (510-987-7700). This program calls the HP-UX routines puts (which writes a string to stdout) and exit (which terminates the program). It looked like this:

```
$native, nocc, unix
begin

equate line'feed = 10;

byte array buf(0:14);

double procedure puts(buf);
    virtual byte array buf;
    option native, nocc, external;

procedure exit(int);
    value    int;
    double  int;
    option  native, nocc, external;
```

```

move buf := ("Hello, World!",line'feed,0);

puts(buf);
exit(0);

end.

```

Compiling and Linking

We compiled the above program using SPLash! and copied the resulting object-code to HP-UX with these commands:

```

:splash testsrc,testobj
:dscopy testobj to /users/rob/test.o:daffy[rob:pass]

```

The next part took about a week of work to figure out: the exact link command to use in HP-UX in order to have our SPLash! program run. The HP-UX command for linking is called ld. To link our example program, we used this ld command:

```
ld /lib/crt0.o test.o /lib/libc.a
```

This linked the three files /lib/crt0.o, test.o, and /lib/libc.a. The first file is /lib/crt zero dot oh (don't mix up the zero and the oh), which is the C run-time startup library. The test.o file is our SPLash! object-code that we generated on MPE. The /lib/libc.a file is one of the HP-UX run-time libraries. Like most HP-UX command names and commands, everything must be in lower case. The general form of the ld command is:

```

ld /lib/crt0.o objfiles... /lib/libc.a
  ^                      ^ C library
  C run-time startup

```

Running Our Program

By default, the ld command creates a program file called a.out. To run this program, we only had to type its name (like using implied run on MPE/iX). We tried running the resulting program and got:

```

a.out          (run our sample program)
Hello, World! (result of running a.out)

```

This was exactly what we expected. The buffer that we initialized with "Hello, World!" was printed out on stdout and the program stopped. We had now answered one of our most basic questions -- we could compile programs on MPE, copy the object-code to HP-UX, link the object-code into a program and run it on HP-UX.

Run-Time Libraries

Once our basic research question was answered, it was time to start exploring more of HP-UX. We had no idea how many of the MPE routines that we needed were available on HP-UX. We used two HP-UX tools to search for routines: man and nm.

Man Pages

The man pages are similar to MPE's on-line help. For example, if you want to see if there is a description of a routine called binary, you do the following:

```
man binary
```

Once done, you will see this:

```
No manual entry for binary.
```

Another way to search the man pages is to have the man utility search a keyword index from all the man pages for a string (see below for how to create the man page index file):

```
man -k binary          (-k requests a keyword search)
.
.                       (many entries printed)
.
```

There were many routines with a description that included the word "binary", but none were similar to the MPE binary routine. The man page index is in the file `/usr/lib/whatis`. You can rebuild this file by doing:

```
catman -w
```

You must be logged on as super user to run catman. It takes catman a long time to create the index, so we suggest that you run it in the background or in a batch job.

Searching Object-Code

Just because a routine is not documented doesn't mean that it does not exist. Many MPE compiler-library routines are available in HP-UX, but not documented. The tool to use for finding symbolic names in object-code is nm. The nm utility is similar to the MPE/iX Linkedit Listobj command. The nm output is similar enough to the Linkedit listobj output to make MPE users feel at home (another advantage of using HP-UX). We combined nm with two other HP-UX tools to search entire directories of files. For example, to search the directory `/usr/lib/` for any files with a symbolic entry for binary we did the following:


```
cd /usr/lib
nm -r /usr/lib/* | grep binary | more
```

If you do this on HP-UX, you get a large number of error messages (not everything in /usr/lib is an object-code file) and the following line is found:

```
libnsfmt.a:binary |1073789772|extern|data |$SHORTDATA$
```

Unfortunately, this is "data". What we needed was "code" or "entry". Libraries on HP-UX are also stored in the directory /lib, so we checked those libraries too:

```
cd /lib
nm -r * | grep binary | more
```

We did this search and still did not find any routine called binary. So it appears that there is no binary routine for HP-UX. Since this routine is used everywhere in our tools, we ended up writing our own "binary" routine in C (note that the HP-UX routine `atoi` is not the same).

Floating-Point Numbers

In MPE, we use the compiler-library routines `hpinext` and `hpextin` to convert floating-point numbers from human-readable form to the internal IEEE format and vice versa. These routines have many options and handle different sizes of floating-point numbers. Rewriting these routines would be a lot of work.

Qedit does not use floating-point numbers very often, except in the calculator. In order to implement the calculator, we would need the functionality of the `hpinext` and `hpextin` routines. A search of the man pages revealed no documentation for either routine, although there were other routines to handle floating-point conversions. All of our code assumes that `hpinext` and `hpextin` are available. So we searched for these routines using `nm`:

```
cd /usr/lib
nm -r * | grep hpinext | more
```

The result of this search was as follows:

<code>libcl.a:hpinext</code>	<code>13836</code>	<code>extern</code>	<code>entry</code>	<code>\$CODE\$</code>
<code>libcl.sl:hpinext</code>	<code>421568</code>	<code>extern</code>	<code>code</code>	<code>\$CODE\$</code>
<code>libcl.sl:hpinext</code>	<code>421524</code>	<code>extern</code>	<code>entry</code>	
<code>libf.a:hpinext</code>	<code>13836</code>	<code>extern</code>	<code>entry</code>	<code>\$CODE\$</code>
<code>libf.sl:hpinext</code>	<code>421568</code>	<code>extern</code>	<code>code</code>	<code>\$CODE\$</code>
<code>libf.sl:hpinext</code>	<code>421524</code>	<code>extern</code>	<code>entry</code>	

Success! We had found the `hpinext` routine. We also found the `hpextin` routine in the same libraries (using the same technique). These routines were located in

either the `libcl.a` or `libf.a` library. Once we started converting the calculator to work on HP-UX, we had to change our `ld` command to include one of these libraries:

```
ld /lib/crt0.o test.o /usr/lib/libcl.a /lib/libc.a
      ^
      New run-time library
```

This was one example of how familiar HP-UX is to MPE programmers. It is highly unlikely that any other version of UNIX has the `hpinext` or the `hpextin` routines. Even if they did, they are unlikely to have the same parameters and work the same way as the HP version.

Phase One Summary

We succeeded in our basic research goals. We obtained access to an HP-UX machine (due to the kind loan from HP) and we proved that SPLash! code compiled on MPE could be linked and executed on HP-UX. The time that elapsed for this phase was about three months (February to April). The work time:

Dave Lo	One person-month.
Bob Green	Half person-month.
David Greer	One person-month.

Phase Two: MPE and Robelle Libraries

The goals for phase two were:

1. Obtain our own HP-UX machine.
2. Learn more about HP-UX.
3. Implement the necessary MPE routines on which we rely.
4. Start porting the Robelle library to HP-UX.

Obtaining an HP-UX Machine

After we had returned our demo machine to HP, we started work to obtain our own HP-UX machine. We did some investigation and decided to rent a Series 705 workstation from HP, complete with the necessary networking (NS) and development (C) tools (don't attempt C development with the "free" C compiler that comes with HP-UX). Since Bob and I work at home most of the time, we

really planned on using the workstation as a server. Because HP-UX is a multi-user operating system, we decided that it would work just fine as a development server. Even today, our Series 705 is used as a workstation by one user in the office and as our primary development machine by those of us doing HP-UX program development.

Implementing MPE Routines on HP-UX

In phase one, we had already started implementing some common MPE routines (binary, dbinary, ascii, and dascii were now done). We continued with other routines such as print, readx, ccode, and hpsetccode. Once these routines were completed we could start porting our most fundamental library source modules.

Porting the Robelle Libraries

With a basic set of MPE routines now available, several of our source modules started working on HP-UX. However, we would often try to port a new Robelle library only to find that we needed to implement additional MPE routines. So the process went back-and-forth between porting Robelle source modules and writing new MPE replacement routines. In total, we implemented about twenty different MPE routines.

MPE File System Routines

Things went well, until we had to port our first source module that needed to access files. Using sample code written by Stan Sieler of Allegro Consultants, Inc. (415-369-2303), we investigated how difficult it would be to implement the MPE fopen, fread, fwrite, and fclose routines.

The HP-UX file system is simple compared to the MPE file system. Files are collections of bytes. There is no implied structure to any file (although you may deduce the structure by examining the filename extension or by looking at some of the data in the file). In MPE, we have fixed-length, variable-length, ascii versus binary, record-length, blocking-factor, and many other file attributes.

Our software depends on a number of assertions that are provided by MPE's file system routines. For example, if you open a new file with a filecode of 111 and a record size of 256 words, write some records to it, close it, open the file again, then ask about its structure, you expect to get a filecode of 111 and a record size of 256 words. In HP-UX, there is no place to store this information (without using an auxiliary file).

It was at this point that we made a pivotal decision about how we would port our software. We would not emulate the MPE file system routines. Instead, we would reengineer our products to invoke an interface layer that would provide file system functionality. We would isolate the interface layer for each module that needed file system access into a separate source file (in object-oriented

terminology this is called encapsulation).

Recognizing Qedit Files

On MPE, Qedit files are easily recognized as those with a filecode of 111 and a record length of 256 words. But how were we going to recognize Qedit files on HP-UX? HP-UX doesn't have filecodes or record lengths. Bob and I designed a routine that would examine the first 1024 bytes of an HP-UX file. By looking at various relationships in this file, we hoped that we could determine if the file was a Qedit file or not.

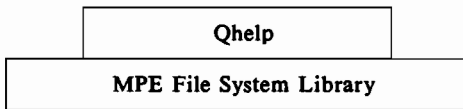
We managed to implement such a routine. To test its effectiveness, we used dscopy to copy one Qedit file from MPE to HP-UX. We wrote a test program that opened every file on our Series 705 workstation and checked to see if it was a Qedit file. Our first implementations found many HP-UX files that the routine thought were Qedit files when they were not. But our final implementation found the one and only Qedit file that we had copied from MPE.

Porting Qhelp to HP-UX

Qhelp is the software that provides all of Robelle's on-line help facilities. Qhelp was nicely isolated into one source module. It uses a Qedit file to store all of the help text and the help keywords. We decided that Qhelp would be our first HP-UX module to use an interface layer to provide file system access on both MPE and HP-UX.

File System Layer

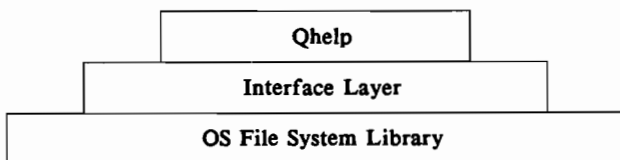
The MPE version of Qhelp invoked the MPE file system directly. It looked basically like this:



Original Qhelp

We reengineered Qhelp to call an interface layer to provide all file system functionality. This interface layer would open a file, let us know whether it was a Qedit file or not, read the file, and close the file. Once Qhelp called the interface layer, it would no longer "know" whether it was running on MPE or on HP-UX.

The new Qhelp looked like:



Revised Qhelp Structure

Implementing the Qhelp Interface Layer

In our revised Qhelp diagram, we added an interface layer that is not operating system specific. The job of the interface layer is to hide the details of the underlying operating system. Each interface layer has two implementations: one for MPE and one for HP-UX. The MPE interface was written in SPL/SPLash! and the HP-UX interface was written in C.

A lot of research went into learning how to map parameters in SPL to parameters in C. We also had to learn all of the HP-UX file system calls, how they worked, how to detect and report errors and so on. All of this took considerable time, but once we got Qhelp working we knew that future rewriting of other modules would be faster.

Phase Two Summary

During this phase we made a lot of progress. Approximately 20,000 lines of existing SPL source code were ported to HP-UX (e.g., the calculator and Qhelp modules) and about 1,500 new lines of C code were written. This provided a strong foundation to begin the port of Qedit to HP-UX. The time that elapsed for this phase was about six months (August to January). The work time:

Dave Lo Two person-months.

Bob Green Half person-month.

David Greer Three person-months.

Phase Three: Rebuild Qedit

By this point, things were looking pretty good. We had a number of useful MPE routines written, a lot of the Robelle library was running on HP-UX, and we had one module (Qhelp) that did file I/O using an interface layer. It was now time for the big show: porting the Qedit source code to HP-UX. This was a much bigger project than what we had completed so far, since Qedit itself consists of about 75,000 lines of code.

The First Attempt

We started by compiling all of Qedit's source modules using SPLash! and copying the resulting object-code to HP-UX. Our first attempt to link Qedit using the ld command had many errors about missing procedures (e.g., all of the MPE file system routines). You can run an HP-UX program with missing externals, but the program aborts if any missing routine is called. To work around this problem, we wrote one source module that had all of the MPE routines that we had not rewritten for HP-UX. Each dummy routine returned a failure status. A portion of this file looked like this:

```
integer procedure fopen;
begin
    fopen := 0;
    hpsetccode(ccl);
end'proc;    <<fopen>>

procedure flock      ;begin hpsetccode(ccl); end'proc;
procedure fpoint    ;begin hpsetccode(ccl); end'proc;
procedure fread      ;begin hpsetccode(ccl); end'proc;
procedure freaddir   ;begin hpsetccode(ccl); end'proc;
procedure freadlabel ;begin hpsetccode(ccl); end'proc;
```

Redo Module

Qedit supports the Do, Redo, and Listredo commands. After our first attempt to link Qedit on HP-UX, we realized that we needed the redo module to work. The redo module also requires a "new" file (these are files that on MPE you cannot see with Listf or Listftemp). HP-UX does not have new or temporary files, but there is a solution. For the invisible scratch files needed by a program, you use the following algorithm on HP-UX:

```
get a temporary filename (use tmpnam or tempnam)
open the temporary filename
unlink the filename from the file system
```

If your program stops for any reason, the scratch file will disappear and the space the file occupied will be returned to the HP-UX free space list.

With the redo module working, we were now able to link our first version of Qedit/UX. However, this version couldn't do much. It accepted commands, but most failed immediately. For example, the command:

```
/open file
```

would fail because the open command was still calling fopen. Therefore, the next step was to completely reengineer Qedit so that no direct calls were made to MPE routines. Like Qhelp, we began to reengineer Qedit to call an interface layer.

Implementing the Qedit Interface Layer

Bob and I took about one month to completely reengineer Qedit so that it called an interface layer for all operating system services. Bob changed Qedit and wrote the MPE version of the interface. I wrote the equivalent interface for HP-UX in C. A lot of our time involved Bob asking me if it was possible to do certain things in HP-UX (e.g., how to obtain exclusive access to an opened file in HP-UX). By now, I had ported enough source code to have a pretty good idea of what was possible and what was not. Writing the interface layer was not our only problem. There were many other parts of Qedit that had to be changed.

Upshifting Filenames

Inside Qedit, filenames were always upshifted, often in several places. On MPE this didn't matter, since filenames are not case-sensitive. On HP-UX, this was now important since filenames are case-sensitive.

Bob had to go through Qedit to find every single location where filenames were upshifted and remove the upshifting code. We also had to cope with a different filename syntax (e.g., `./init.c` is a valid HP-UX filename).

While we made these changes for HP-UX, we are now taking advantage of these same changes for POSIX on MPE. In MPE/iX 5.0, POSIX filenames have the same syntax as on HP-UX (they are mixed-case and can contain directory names). So our efforts have paid off twice.

The 32-Bit Alignment Problem

The HP-UX C compiler prefers to have 32-bit variables aligned on 32-bit boundaries. In SPL and SPLash!, it is common to have 32-bit variables aligned on 16-bit boundaries. The interface layer for Qedit was much more complicated than any of the other interfaces that we had written. Because the interface was

more sophisticated, large data structures had to be passed between SPLash! and C routines. We had to carefully implement these structures so that all 32-bit variables were aligned on 32-bit boundaries. We also had to insure that these structures themselves started on a 32-bit boundary.

You can force SPLash! to allocate arrays on 32-bit boundaries by making the arrays **virtual**. This often has unexpected side-effects (e.g., you cannot pass a virtual integer array to an option splash procedure that is only expecting an integer array). In Qedit, we ended up doing the following. We would allocate one more word of storage than was necessary and then would have code that looked like this:

```
if @ext'record mod 2 <> 0 then
    @ext'record:=@ext'record(1); !assumes integer array
```

This code fragment adjusts the start of the ext'record structure if it does not start on a 32-bit boundary. We also had to be very careful of 32-bit variables that were passed by reference (these are often status parameters). It was often very difficult to detect a 32-bit alignment problem: due to the nature of SPL and SPLash!, the problems can come and go.

The End-Of-File Problem

Many parts of Qedit need to know how many records are in a file. For example, to print the line numbers for a command like:

```
/list somefile last-10/
```

requires that Qedit know how many records are in **somefile**. MPE remembers how many records are in a file while HP-UX does not.

Bob realized that rewriting Qedit so that it did not need to know how many records were in a file would take longer than everything we had done so far. This meant that in the HP-UX interface layer we have to read a file from beginning to end in order to count the number of lines in it. We do this every time that we open an external file in Qedit/UX. It's not very efficient, but it was the only way to provide this information to Qedit.

Implementing Full-Screen

By this point we had many of Qedit's line-mode commands working on HP-UX. The big question left was what to do about Qedit's full-screen mode. On MPE, Qedit uses block-mode to implement full-screen. On HP-UX, we didn't know what we were going to do about full-screen.

One Friday night after a long day, my two children came into my office (at my home). They wanted to play the computer game "The Playroom". I soon had one child on each knee (they were 4 and 2 at the time), happily playing on my

computer. I was surrounded by the HP-UX reference manuals, which I had been using all day. As my children were playing, I started flipping through one of the manuals. I wasn't really concentrating, but suddenly I noticed an entry for **blmode**. The opening description read:

"This terminal library interface allows support of block mode transfers with HP terminals."

By accident, I had found a set of routines that implemented block-mode. Bob liked the HP-UX block-mode interface enough that he reengineered the Qedit block-mode interface on MPE to look like the one on HP-UX. In a few days time, we had full-screen working on HP-UX.

Phase Three Summary

Qedit was now running on HP-UX. The process we used helped ensure that future Qedit enhancements would appear in both versions, since only one set of source code is used for the MPE and HP-UX versions of Qedit. Most of Qedit no longer knew which platform it was running on. The time that elapsed for this phase was about three months (February to April). The work time:

Dave Lo One person-month.

Bob Green Two person-months.

David Greer Two person-months.

Phase Four: Polishing Qedit/UX

With the basic research complete, we now had to polish Qedit/UX to turn it into a product. Up to this point I had to write all of the C code on HP-UX using the vi editor. I was certainly pleased when we had a working version of Qedit/UX that I could start using. I wasn't sorry to see the end of vi.

Porting the Test Suite

For years, we have developed test suites for all of our products. The Qedit test suite has a few hundred tests organized into sixty-three batch jobs. We knew that in order for Qedit/UX to be reliable, we would have to have a good portion of our test suite working on HP-UX.

There was no automated way to convert our tests from MPE to HP-UX. We had to go through each test manually and convert MPE commands to their equivalent HP-UX commands. We also had to implement several of our test tools on HP-UX (e.g., the program that compares two Qedit files one line at a time looking for differences).

Temporary Files

On MPE, Qedit uses temporary files for a scratch file and for "hold" files. The hold filename on MPE was chosen so that you could use the name in commands. For example,

```
/open file1
/hold 15/20           (save lines 15/20 in file "hold")
/open file2
/add 75=hold         (add lines from the "hold" file)
```

Initial versions of Qedit/UX created files in the current directory called QEDITSCR, HOLD, and HOLD0 (note the upper case filenames). The Add command in the previous example would not work, since there was no file called "hold": it was "HOLD".

We modified Qedit/UX to create the qeditscr, hold, and hold0 files in /usr/tmp. HP-UX system managers expect temporary files to be created in the /usr/tmp directory (our nightly backup script removes all files from /usr/tmp more than twenty days old). We also had to modify Qedit/UX so that when it held lines or when you wanted to look at a file called "hold", it substituted the temporary filename. For example,

```
/open file1
/hold 15/20           (save lines in "qholdBAAa23359")
/open file2
/add 75=hold         (add lines from "qholdBAAa23359")
```

On HP-UX there really are no temporary files. HP-UX programs just create permanent files in a directory considered to be temporary. The different-looking filename is a result of the HP-UX routine tempnam which creates a file with a unique name in /usr/tmp.

More HP-UX Features

Qedit/UX was modified to execute shell commands (a shell is like the MPE/iX CI.PUB.SYS program). One of the most common HP-UX commands is ls (like the MPE Listf command). This conflicted with the Qedit command LSort. In Qedit/UX, we made ls mean list files and LSO mean sort lines (although the LSort command is not yet implemented).

HP-UX users can have different shells. We changed Qedit/UX so that when it executes shell commands it does so with the user's preferred login shell.

We could not obtain command line arguments in our SPLash! programs. Fortunately, Stan Sieler worked on the problem and modified SPLash! to make the argument count and a pointer to the argument list available to SPLash! programs. HP-UX programs expect options to be specified by a leading dash. We changed Qedit/UX to accept command line arguments such as this:

```
qedit -s -cvi file1
```

This example asks Qedit/UX to edit a single file (-s), go directly into visual-mode (-cvi), and to edit file1. This also made it possible to set the HP-UX EDITOR variable, so that any HP-UX program that invoked an editor would invoke Qedit/UX on behalf of the user.

Tab Characters

One problem that we have not solved adequately is how to handle tab characters. In MPE, tab characters are used as a short-hand way of moving the cursor across the screen. The tab character is always translated into spaces and never stored in files. On HP-UX, the tab character is also used to move across the screen, but it is also used as data. Some programs (e.g., make and sendmail) require that certain lines have tab characters.

In Qedit/UX, we automatically expand tabs to spaces when we text a file. This is not what HP-UX users expect, especially if you are editing a Makefile or the sendmail configuration file (important data gets removed by Qedit/UX). Qedit/UX has a feature to prevent tabs from turning into spaces, but it is not the default. If you leave tabs as data and go into full-screen, you will lose the tab characters that were in the lines shown on the screen.

There are various technical problems that make it difficult for Qedit to treat tabs as data. How to handle tab characters is similar to our number-of-records problem in the file system interface. It's a fundamental assumption that is different on HP-UX from MPE. We continue to work on ideas in Qedit/UX to solve this problem.

Releasing a Product

Besides all of the technical and programming problems, there were many other issues to deal with. We had to create a whole new set of software to make tapes. I spent two full weeks rewriting the Qedit documentation so that we could have both a Qedit/UX user manual and on-line help. There were data sheets to write, advertising, training (every technical person at Robelle has taken the week-long HP-UX introductory course), and pre-release customers to work with. It was all worth it when we first demonstrated Qedit/UX at the San Francisco Interex User Group Meeting. We had many excited customers and just a week later we started shipping Qedit/UX.

Phase Four Summary

Our final phase took a working R&D program and turned it into a new product. Many changes were made to integrate Qedit/UX into HP-UX. The supporting documentation and technical support were created to provide a high quality

product for our customers. The time that elapsed for this phase was about five months (May to September). The work time:

Dave Lo Two person-months.

Bob Green Two person-months.

David Greer Two person-months.

The Future of Qedit/UX

We have proven that the initial concept of cross-platform development using SPLash! was feasible. The object-code format on MPE/iX and HP-UX are the same. It is possible to compile SPLash! code on MPE, copy it to HP-UX, and then link and run the resulting program on HP-UX. In total, we ported over 100,000 lines of code. We could never have rewritten Qedit in the same amount of time.

Full-Screen

Our use of the blmode routines enabled us to get full-screen mode up and running quickly. Now that Qedit/UX has been released for several months, it is clear that even on HP-UX many users have decided to use vt terminals or vt terminal emulators. Because full-screen uses HP block-mode, it requires an HP terminal and will not work with vt terminals. One major research goal for 1994 is to reengineer full-screen to use a terminal-independent method of communicating with the user.

Shell Commands

On MPE, users are used to living inside Qedit all day. You cannot do this in Qedit/UX, because many common shell features are missing (e.g., setting environment variables). Some of these are due to limitations of UNIX, but we expect to implement many more shell features in Qedit/UX. It took us more than ten years to fully integrate MPE command features into Qedit/MPE, so it isn't a big surprise that it will take us a year or two to add all the necessary shell features to Qedit/UX.

MPE Flavor

Qedit was originally written for MPE. There are many features of Qedit/UX that retain an MPE flavor. If we are to expand the potential market of Qedit to users who have never seen MPE, we will have to remove most of this MPE feel and introduce more of an HP-UX feel. This is a challenge, since we still want Qedit/UX to look familiar to former Qedit/MPE users.

The End

Qedit/UX has been both a technical and a marketing success. We now have one set of source code that can be used to generate two different products: Qedit/MPE and Qedit/UX. If you have a lot of SPL/SPLash! code, perhaps our solution will work for you as well.

Copyright Robelle Consulting Ltd. 1994

Permission is granted to reprint this document (but not for profit), provided that copyright notice is given.

On-Line Analytical Processing (OLAP): Spoken Here

**Gary Farner
IRI Software**

**Presented to Interex 1994
Conference and Expo**

On-Line Analytical Processing (OLAP): Spoken Here

Relational databases, which use the Structured Query Language (SQL) as a standard to organize, manage, update, and access data, have dominated database technology for nearly 15 years. Thousands of relational databases have been sold, and most drive applications known as on-line transaction processing (OLTP). If you don't use a relational database, you undoubtedly considered using one.

Today, a different database model is supplementing relational databases to make their rich data much more useful. The model is called multidimensional, and it drives applications known as on-line analytical processing (OLAP).

Multidimensional products derive their name from the multidimensional nature of most business inquiry. In a Research Note called "The Trouble with SQL," the Gartner Group described the many dimensions in which analysts typically examine data:

"Finance managers of large organizations don't ask, 'How much have we spent?'—a zero-dimensional question. Instead, they ask, 'How much have we spent on health benefits, by month, in division X, in each state, compared with plan?'—a five-dimensional question."

By answering such questions, multidimensional databases supporting OLAP turn corporate data into business intelligence (BI). BI systems give users access to many sources of data, but they also enable users, through a process of discovery, to understand the forces that shape the data. Equipped with this understanding, professionals can solve business problems and capitalize on business strengths.

This paper explores the business needs that inspired hundreds of Fortune 500 companies to supplement their relational databases with multidimensional OLAP servers. It describes why and how the relational and multidimensional technologies, with their characteristic applications, coexist in Information Systems (IS) infrastructures. It details the architectural differences that suit relational systems to transaction processing, and suit multidimensional systems to analyzing data. And it describes an OLAP session during which an analyst uses multidimensional tools to derive business intelligence from corporate data.

Multidimensional Momentum

Multidimensional databases are not new. In 1972, Management Decision Systems (MDS), a consulting firm founded by professors at MIT and the Wharton School of Business, pioneered the use of multidimensional tools in decision support systems that it created for clients. These clients included firms from the consumer packaged goods industry, which analyzed massive stores of sales and marketing data.

Multidimensional analysis, however, is gathering momentum in the broader marketplace. One cause of this momentum is the explosion of information, supplied in large part by relational tools. Another factor is the pressure of global competition.

As the Aberdeen Group explained, the challenge facing organizations today is not collecting data, but interpreting it:

"Aberdeen believes enterprises must work smarter, not just harder. To do this, enterprises must separate information wheat from data chaff, then turn the information into competitive advantage-gaining Business Intelligence."

The Hurwitz Consulting Group also acknowledged the role that the bottom line plays. Judith S. Hurwitz described two contributors to the momentum of multidimensional products:

"Recent technological and business developments are causing companies outside of the consumer products niche to begin to look at complex multidimensional data modeling products. From a technology perspective, systems that once required a mainframe can now operate on a PC. From a business perspective, more executives are recognizing that information is power; understanding subtle changes in buying patterns and your customer base, whether you are selling cereal or money market funds, can make a big difference in the bottom line."

There is another factor in what Hurwitz calls "technological developments." It is the technical limitations of SQL for analytical applications.

In "The Trouble with SQL," the Gartner Group explained, "SQL and the two-dimensional relational model of data have severe limitations in business intelligence. The best multidimensional products are significantly more productive when applied to real-world problems."

E. F. Codd, the original developer of the relational database concept in 1970, explained why SQL is not a productive BI tool in a recent issue of *Computerworld*:

"As enabling as RDBMSs (relational database management systems) have been for users, they were never intended to provide powerful functions for data synthesis, analysis and consolidation (functions collectively known as multidimensional data analysis)."

OLTP and OLAP: Contrasting and Complementary Applications

In general, relational databases were designed for on-line transaction processing (OLTP). These are record-oriented applications, such as billing, order entry, accounting, and inventory.

In contrast, multidimensional databases were designed for on-line analytical processing (OLAP). These are array-oriented applications, such as market analysis and financial forecasting. An *array* is multiple occurrences of items, such as product sales across markets, or profit over time.

The Aberdeen Group described OLTP, OLAP, and BI as follows:

"Aberdeen sees relational databases used most frequently today for on-line transaction processing (OLTP) applications that seek operational efficiency. BI technology is most useful for intense decision support—a class of applications now being called on-line analytical processing (OLAP)—that is being used to achieve better returns on investment or greater market share."

Table 1 describes the general differences between relational (OLTP) and multidimensional (OLAP) applications.

Characteristics	Application	
	OLTP	OLAP
Typical operation	Update	Analyze
Screens	Unchanging	User-defined
Data per transaction	Little	Lots
Data level	Detail	Aggregate
Age of data	Current	Historical, current, and projected
Orientation	Records	Arrays

Table 1. Application Contrasts

As different as they are, OLTP and OLAP are elements in an information triad that results in BI. Figure 1 shows the dependency of decision-makers on the two categories of applications, along with the emerging category of Groupware, which includes products like Lotus Notes.

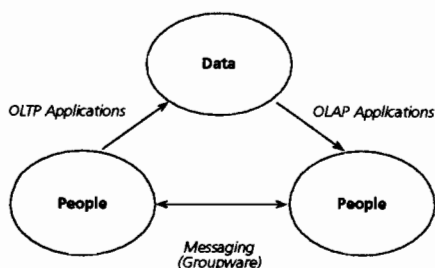


Figure 1. Interdependence of OLTP and OLAP in information triad.

Organizations can use relational tools for simple data analysis, but when they try to analyze complex data with relational tools, the tools lose their effectiveness. Applying SQL to BI applications is like applying a spreadsheet to document processing. Roadblocks result when users try to generate something, like the index of a document, that spreadsheets were not designed to produce. Similarly, if SQL is applied to complex analysis, roadblocks result from using the wrong tool for the job.

Evolution of SQL-Based Analysis

SQL roadblocks to business intelligence vary with relational configurations. Many organizations tried the following type of configuration:

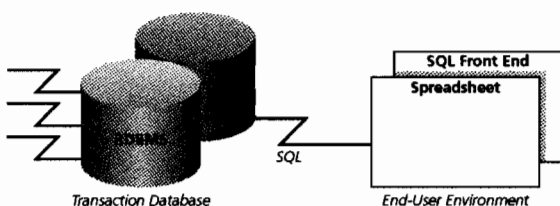


Figure 2. One relational database for transactions and analysis.

In Figure 2, analytical users directly query the transaction database, where users also enter orders and perform other data-gathering tasks. The transaction database is *normalized* for fast processing and efficient data storage; it is broken into many tables.

For years, professionals hoped this configuration could support both transactional and analytical applications. But users had problems querying the data, whose table structure they were forced to understand to perform relational joins, which combine data from different relational tables. And queries were slow. Faith, however, was placed in two developments that promised to deliver decision support: relational query and reporting tools, and hardware power, such as parallel processing.

Advantages of Querying Transactional RDBMS

The advantages of this design include:

- Users see up-to-date data at all times.
- Relational database products are available for a variety of platforms.
- ANSI standards exist for the query language SQL.
- There is only one system to maintain.

Disadvantages of Querying Transactional RDBMS

The disadvantages of this design include:

- Analytical users can lock out, or slow down, transaction processors, and vice versa.
- Additional indexes, which speed data access, often need to be maintained for analytical users.

These indexes have two disadvantages. They slow data updates. And because of the relational requirement that the key must be stored with every indexed record, they consume disk space.

- SQL has severe analytical limitations.
 - The two-dimensional view of relational data is not intuitive, and is difficult to use because it requires performing joins on normalized data. (If a relational database were completely denormalized, all data would appear in one huge table, and users would not have to join tables. Storage requirements for such a denormalized database, however, would skyrocket.)

Data dipper tools are designed to help users understand and access normalized data. These front-end query tools, however, handle only relatively simple queries.

- SQL is not designed to handle time-series data or perform sophisticated mathematical functions.

Time-series calculations, like a three-month moving average, or other operations performed on ordered sets, like net present value, typically require extensions to the ANSI-standard SQL. These extensions are rarely found in commercial products.
- Queries that require joins are slow.

Software solutions to speed relational queries include specialized relational add-ins that are coming on the market. Vendors like Oracle and Ingres Corp. are beginning to offer hardware-related solutions that take advantage of parallel processing.

- IS personnel who create end-user applications must write and maintain substantial, often complex SQL code.
- For users who do not want to construct SQL statements, IS must try to anticipate analytical needs.

The IS problems inherent in anticipating end-user needs apply to all relational configurations, including those that support the relational data warehouse. So long as IS must predict what users will analyze, and must consequently design databases based on that prediction, users who want to be shielded from SQL are likely to be frustrated by the limitation of their queries.

- The design implies centralized data storage, which results in communications charges for remote users who access the centralized data through, for example, leased lines.

Organizations that have been using relational databases for years probably have separate relational databases for their transaction processors and analysts:

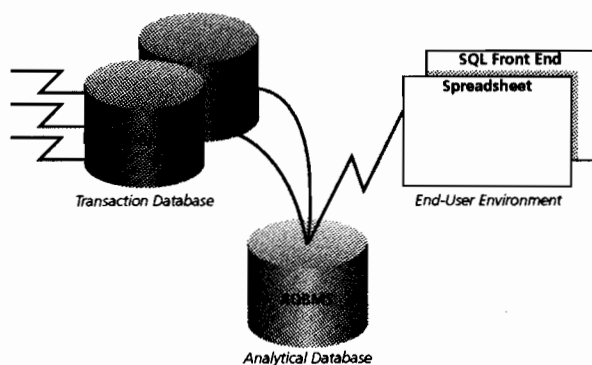


Figure 3. Two relational databases for transactions and analysis.

In Figure 3, the analytical database generally duplicates needed data from the transaction database. (Alternatively, the analytical database has views of the transaction database, but performance considerations often make this configuration impractical.) Whereas data in the transaction database is normalized, for efficiency, data in the analytical database is denormalized, for ease of access.

Advantages of Querying Analytical RDBMS

The advantages of this design over the previous configuration, which uses one relational database for both transactions and analysis, include:

- Since data is duplicated, transaction processors do not contend with analysts for access to data.
- Queries can be sped by the creation of as many indexes as are necessary in the analytical database.
- Updates to both the transaction-processing and analytical databases are fast because they are relational, unless the analytical database has many indexes to support user queries.
- A single technology language, SQL, is required to maintain both databases.

Disadvantages of Querying Analytical RDBMS

The disadvantages of this design include the analytical limitations of SQL and the burden on IS that characterized the previous configuration. In addition to those disadvantages, others are:

- Analytical users must rely on updates to their data for current information.
- Because every user query can't be anticipated, a high level of optimization is unlikely. Indexes to support all possible analytical angles can't be created. And data can't generally be denormalized to the point where users never have to perform joins.

- Two databases must be maintained instead of one.
- Because of denormalized data and indexes, analytical relational databases consume more disk space than transaction databases.

Configuring Relational and Multidimensional Tools

The relational roadblocks to BI might have been considered inevitable, and therefore acceptable, had not the data structures of multidimensionality presented a path to BI that preserves investments in relational tools and data. Multidimensional tools deliver powerful, fast queries that users can run with minimal IS support, as we will see in subsequent sections of this paper. But how do relational and multidimensional tools coexist? A multidimensional OLAP server fits into existing corporate infrastructures (Figure 4).

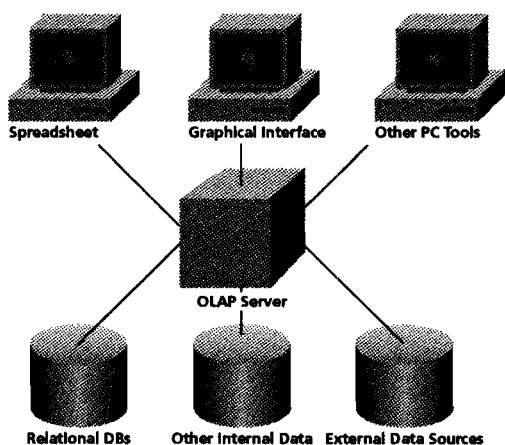


Figure 4. OLAP server integrates multiple data sources.

Options for Data Location

In an architecture that includes relational and multidimensional tools, data can be stored in a multidimensional server, or the multidimensional server can be used as a cache for relational data. Businesses can use a combination of these two approaches to minimize the amount of data that moves between the relational and multidimensional environments.

Regardless of where data is stored, the multidimensional server can access relational data either in real time, or through a batch process.

Common Environments

Because relational databases operate in software and hardware environments where multidimensional servers also operate, organizations can use existing hardware and software environments for both transaction processing and analysis. For example, some multidimensional servers operate in environments that include Windows, DOS, OS/2, most variations of Unix, VAX/VMS, MVS, and VM. Like relational servers, multidimensional servers can run in standalone, local area network (LAN), or wide-area network (WAN) configurations.

Typically, the relational database has more detailed data than the multidimensional server. For example, in a banking application, data for all tellers' transactions is stored in the relational database, while summary daily data for a teller's transactions is stored in the multidimensional server. The flexibility of multidimensional systems, however, allows users to "reach through" to relational data when they want the details behind the summary.

How Coexistence Is Possible

Several options exist for configuring relational and multidimensional systems. Organizations can design knowledge of relational data in the multidimensional server, or design knowledge of data requests into the relational database.

In the first case, the data dictionary, or system catalog, of the multidimensional server includes definitions to construct dynamic SQL statements. Those statements access the relational data sources when required.

In the second case, the relational database acts as an extension to the multidimensional data dictionary. The multidimensional data dictionary identifies tables in the relational database where the extended dictionary information is kept. Those relational tables have a predefined format that the multidimensional server expects.

In either case, the multidimensional equivalent of SQL *stored procedures* is created. These procedures populate, on an as-needed basis, the multidimensional server with values from the relational database.

Why Coexistence Is Possible

Relational and multidimensional *data models* are mathematically very similar. A data model is the logical representation of how the data is structured, and how operations are performed on it. This similarity allows easy mapping of relational to multidimensional data.

For example, the *variables* in a multidimensional model can be derived from the columns in a relational model. Values for the multidimensional variable Units in Figure 5 map on a one-to-one basis to the values in the relational table whose column is Units. Similarly, the dimensions of multidimensional systems, such as Time, relate directly to the keys that uniquely identify the rows in relational tables.

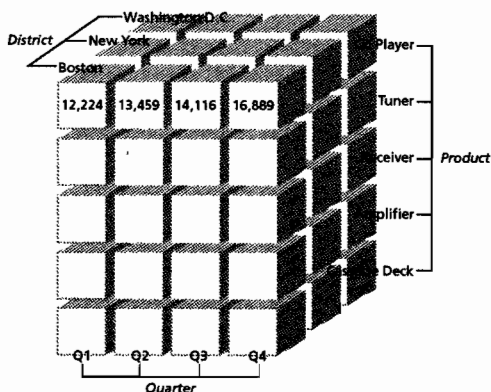
If the two data models are mathematically similar, how do their structures result in functional differences?

Flat Tables or Cubes Affect Analysis and Performance

We've already said that a data model is logical representation of how the data is structured, and how operations are performed on it. The data model also determines what users see, what they can calculate, how fast calculations run, and what IS personnel maintain. Both relational and multidimensional data models can support analysis, but using, writing, and maintaining analytical code for relational databases requires comparatively greater effort and time.

Understanding the Basics of the Two Data Models

The data model for multidimensionality is a cube. The data model for relational technology is a flat table (Figure 5).



Product	Market	Time	Units
CD Player	Boston	Q1	12,224
CD Player	Boston	Q2	13,459
CD Player	Boston	Q3	14,116
CD Player	Boston	Q4	16,889
CD Player	New York	Q1	20,167
CD Player	New York	Q2	19,785
CD Player	New York	Q3	21,398

Figure 5. Units variable in two data models: multidimensional cube and relational flat table.

The cubes of multidimensional databases associate variables with *dimensions*. A dimension is a set of similar entities. A Market dimension, for example, is a set that includes Los Angeles and New York. These similar entities are known as *dimension values*.

Dimensions, such as Product, Time, and Market, run along the edges of a cube. The variable is the actual data. Each block in the cube in Figure 5 is a value for the Units variable for a particular Time period, Product, and Market by which Units is dimensioned.

The following command from a multidimensional data manipulation language (DML) defines the variable Units in relation to its dimensions.

```
DEFINE Units VARIABLE DECIMAL <Time Product Market>
```

We mentioned before that an array is multiple occurrences of items, such as the multiple values for Units. In the multidimensional data model, all data becomes elements of arrays. These arrays stress data relationships.

Some multidimensional tools let you define many variables that share some, but not all, dimensions. Such multidimensional tools flexibly model your business by supporting the different arrays needed for analysis in one database. For example, in the same database that contains the variable Units, the variable Price is defined only by the Time and Product dimensions, because product prices are consistent across markets:

```
DEFINE Price VARIABLE DECIMAL <Time Product>
```

To calculate variables such as Dollar Sales, it's important to support both the Unit and Price arrays.

The dimensions of the multidimensional model are like *keys* in the relational model. Dimensions, however, are stored only once and are used repeatedly. Dimensions enforce *referential integrity*, which is the internal consistency of the database.

Although it is easy to visualize a single cube with three dimensions, it's harder to visualize the four, five, or six dimensions that the multidimensional database supports. Fortunately, users don't have to. Users generally need to examine only a two-dimensional slice on the screen at one time, and displays of multidimensional data don't require users to understand the database structure.

Joins Affect Users and IS Support of Users

A great advantage of the multidimensional model is that it obviates multiple-table queries, or *join operations*. Analysts working with a multidimensional model, for example, don't have to find the table that contains the column Units, and then perform a self join to display the column Units Year Ago.

Such joins are implicit in multidimensional data. Because the arrays of data are subject-oriented in the multidimensional model, users easily group "like items." As the analysis at the end of this paper illustrates, users of multidimensional tools just choose what columns and rows to display in reports and graphs; a row in one report easily becomes a column in another report.

Grouping like items resembles slicing the cube (Figure 6).

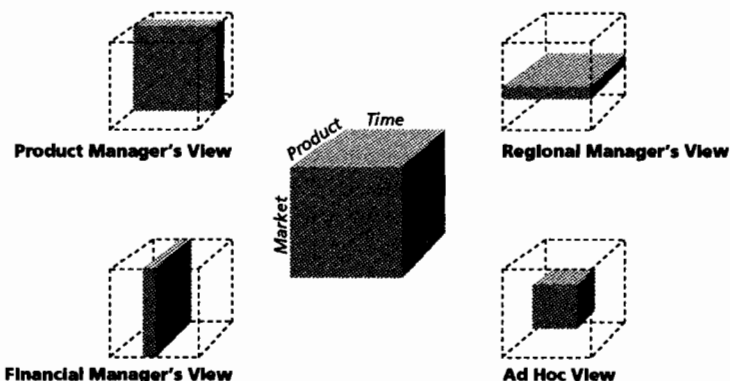


Figure 6. Sample views supported by multidimensional data.

Product managers study one product across many time periods and markets. Financial managers focus on the current and previous time period for all markets and all products. Regional managers examine all time periods and all products across some markets. And strategic planners might focus on a subset of the corporation's data, such as the current and next quarters for an innovative product being sold only in the West. Users display the views of data in Figure 6 without having to specify the cube's structure.

In contrast, the relational data model forces users to understand the flat table structure for two reasons. One, users have to join tables with different columns of data before they can display all the columns of data. And two, because rows in a relational database cannot be converted to columns in a presentation, users have to create self joins to display those rows as columns.

Let's look at a typical join, and then consider how joins affect both users and the IS personnel who support them.

Suppose that you have a relational database. You are using one of the popular SQL query tools to report on budgeted vs. actual expenses. The columns of expense summary data that interest you include Account Number, Account Description, Month, Year, Budgeted Amount, and Actual Expense.

First, you need to know what tables contain these columns, and then you need to choose those tables. You therefore display the **Choose Tables/Views** dialog box (Figure 7).

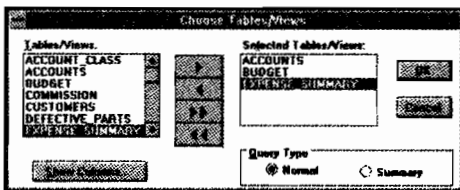


Figure 7. Finding tables that contain columns you want to view.

When you choose the **Show Columns** button in Figure 7, you see a list of columns per table. Perusing the columns in the available relational tables, you select three tables for your expense report: BUDGET, ACCOUNTS and EXPENSE_SUMMARY.

The next step you take is joining the tables. For example, you link the ACCOUNTS and BUDGET tables (Figure 8A), and the BUDGET and EXPENSE_SUMMARY tables (Figure 8B).

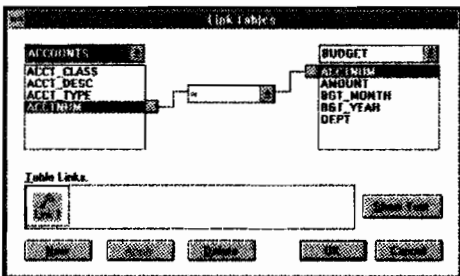


Figure 8A. Joining tables ACCOUNTS and BUDGET.

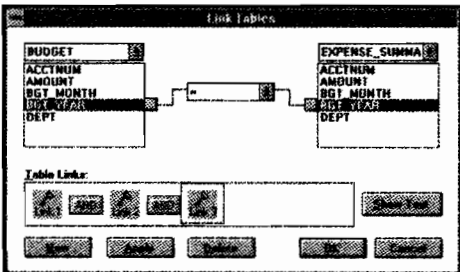


Figure 8B. Joining tables BUDGET and EXPENSE_SUMMARY.

Finally, you select the columns from the three joined tables that you want to display (Figure 9).

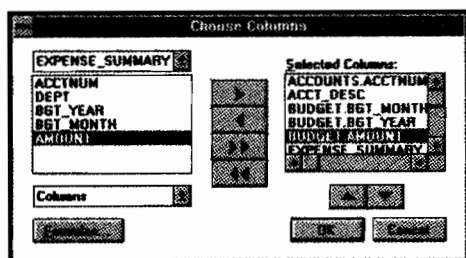


Figure 9. Selecting columns for report.

The System vs. the Business View

Joins like the ones in Figures 7-9 are not difficult to perform, but they force users to consider the structure of the data before they can access and analyze it. This detour to the analysis is not ideal. According to a Gartner Group *Research Note*, the difference between multidimensional and relational queries is the difference between concentrating on the computer system and concentrating on your business:

"Unlike most databases, multidimensional databases better represent the business view of data (rather than the system view). This enables users to navigate through the many dimensions and levels of the data, via tables or graphs, to uncover important trends."

Some highly skilled SQL users are not discouraged by the system view. As Judith S. Hurwitz wrote in *Client/Server ToolWatch*:

"...users can create incredibly complicated SQL queries to force relational two-dimensional databases to mimic the behavior of more complex models."

Most relational end users, however, prefer to concentrate on the business view. They therefore rely on IS personnel to design views for them. It is exactly this reliance that places a burden on IS.

Burdening IS

In a *Datamation* article, Peter Kastner, vice president of the Aberdeen Group Inc., described the nature of the problem for both users and IS when a relational database must be queried.

"Human beings ask questions that are very straightforward, like 'What is our 12-month rolling sales total?' That's a question any executive will ask. And the IS guy turns around and says, 'Well, that's going to take three months to program.'"

Multidimensional environments relieve some of the burden on IS by enabling users to answer their own questions. The *Datamation* story for which Kastner was interviewed, called "Multidimensional Analysis: Winning the Competitive Game," described a multidimensional environment at Quaker Oats Co., in Chicago.

Quaker's IS manager, David Breig, installed a multidimensional database and application. According to *Datamation*, "...Breig gave decision makers tools that let them ask their own questions—even the gnarly ones—with less fear of tying up database servers in endless joins, without having to master Structured Query Language and without a lot of assistance from IS."

Impact of Joins on Query Speed

Joins also affect the complexity and speed of queries. Figure 10 plots the different execution times of two queries run against both relational and multidimensional databases, each with about 6,000 products, 100 time periods, 50 markets, and less than 100 variables. The queries were executed on an IBM ES/9000-9121 with 128MB RAM and a 15.7 MIPS uniprocessor.

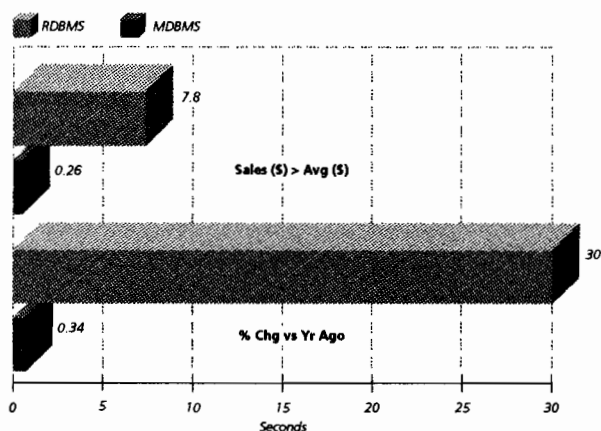


Figure 10. Run times of two queries executed in multidimensional and relational environments.

To calculate greater than average sales, the multidimensional database performed nearly 30 times faster than the relational database. When calculating Percent Change vs. a Year Ago, the multidimensional tool was almost 88 times faster. Perhaps the most significant factor in the sluggish processing of the SQL queries is their joins.

Joins slow queries by making the access of relational data complex. There are simple joins (equi-joins), non-equi joins, inner joins, outer joins, and self joins. The access of relational data for analysis invariably involves joins.

- To display a report that involves a hierarchy, relational queries require one of two types of joins. The query might involve multiple tables, and would therefore require a simple join, such as joining the Employee table to the Department table. Or the query might involve a recursive table, which would require a self join. To manage a hierarchy in relational systems, developers create database structures in anticipation of joining those structures.

In multidimensional environments, hierarchical relationships are explicit objects in the database. These relationships define, for the end user, the levels of the hierarchy and the members within each level. As a result of these relationships, multidimensional queries and reports reference the relationships rather than creating the relationships via joins. Users select which hierarchy they want to display, and select which levels of the hierarchy to display. We'll see an example of this later when an analyst at a fictitious firm, Saturn Electronics, switches from the Standard to the Parent Account hierarchy of the Geography dimension in a multidimensional database.

- Period-to-period comparisons in relational databases involve a self join.

In multidimensional systems, period-to-period comparisons are accomplished by the matrix arithmetic that the model supports. Multidimensional matrix operations can perform calculations on both columns and rows of data.

- Establishing null rows of data in relational systems, to answer questions like, "Which products didn't sell last quarter?" and "Which customers didn't buy?" requires an outer join. In relational databases, because the existence of records indicates the occurrence of a data point, an outer join is required to define all possible data points.

In multidimensional systems, users can see values of missing, or not available (NA), null data. The multidimensional array can define a logical view of the universe of all possible data points, including what isn't there. The NA values, however, are not stored in most multidimensional databases, saving disk space through various compression techniques.

Impact of Other Relational Structures

Joins in relational queries are not the only factor that contributes to slow execution. Other elements in the two data models affect both analytical power, and IS support requirements.

Using Views and Stored Procedures

IS personnel create *views* of relational data to save users from having to perform joins. Views are fairly effective for this purpose. Complex analysis, however, may require more advanced techniques, like stored procedures. Stored procedures can enhance query performance and shield end users from involved SQL statements.

There are two disadvantages to views and stored procedures. First, if additional indexes are maintained to facilitate adequate performance of the views or stored procedures, then update performance of the underlying relational tables can be adversely affected. Second, views and stored procedures are typically maintained by IS personnel.

Neither views nor stored procedures are required in multidimensional analysis, although both are generally available for other purposes. For example, their multidimensional equivalents can automate repetitive tasks or common queries.

Storing Summary Data

Although indexes speed relational data access, they can't make up for the slowness that results when relational tables break summary information across physical database pages. Summary information is what analysts most often need.

For example, Figure 11 shows the typical difference between relational and multidimensional storage of summary variables, such as Sales and Cost: whereas summary data variables are stored together in multidimensional databases, they are fragmented across pages in relational systems.

Relational						Multidimensional			
Prod	Mkt	Date	Sales	Cost	Units	Sales	Sales	Sales	Sales
Prod	Mkt	Date	Sales	Cost	Units	Sales	Sales	Sales	Sales
Prod	Mkt	Date	Sales	Cost	Units	Sales	Sales	Sales	Sales
Prod	Mkt	Date	Sales	Cost	Units	Sales	Sales	Sales	Sales
Prod	Mkt	Date	Sales	Cost	Units	Cost	Cost	Cost	Cost
Prod	Mkt	Date	Sales	Cost	Units	Cost	Cost	Cost	Cost
Prod	Mkt	Date	Sales	Cost	Units	Cost	Cost	Cost	Cost
Prod	Mkt	Date	Sales	Cost	Units	Cost	Cost	Cost	Cost

Figure 11. Storage of summary variables in two relational and two multidimensional database pages.

How do these storage styles affect query speed? Figure 12 shows that accessing eight values for Sales involves two relational database pages, but involves only one multidimensional database page. As more variables are added to the database, these differences become greater.

Relational						Multidimensional			
Prod	Mkt	Date	Sales	Cost	Units	Sales	Sales	Sales	Sales
Prod	Mkt	Date	Sales	Cost	Units	Sales	Sales	Sales	Sales
Prod	Mkt	Date	Sales	Cost	Units	Sales	Sales	Sales	Sales
Prod	Mkt	Date	Sales	Cost	Units	Sales	Sales	Sales	Sales
Prod	Mkt	Date	Sales	Cost	Units	Cost	Cost	Cost	Cost
Prod	Mkt	Date	Sales	Cost	Units	Cost	Cost	Cost	Cost
Prod	Mkt	Date	Sales	Cost	Units	Cost	Cost	Cost	Cost
Prod	Mkt	Date	Sales	Cost	Units	Cost	Cost	Cost	Cost

Two Pages Accessed
One Page Accessed

Figure 12. Storage method affects number of database pages accessed.

Calculating Columns and Rows

Relational queries are often less powerful than multidimensional queries because the relational data model does not support matrix operations as efficiently as the multidimensional model does. You can think of *matrix arithmetic* as operations performed on a whole array at once.

Furthermore, multidimensional applications perform matrix operations on both the rows and columns of multidimensional data. In contrast, SQL performs operations only on columns within the same row.

In relational, for example, you can't subtract one row from another. You need to trick the relational database, by using a self join, into thinking it has two sets of columns on which it performs the operation. In multidimensional, both rows and columns are just cells in a matrix. Those cells can be referenced randomly.

A simple example of matrix arithmetic is performing a period-to-period comparison. The Saturn Electronics analyst will examine Units, Units Year Ago, and Units % Change Year Ago. You can think of Units and Units Year Ago as two cube slices. Units % Change Year Ago resembles a third slice representing the variance between the two.

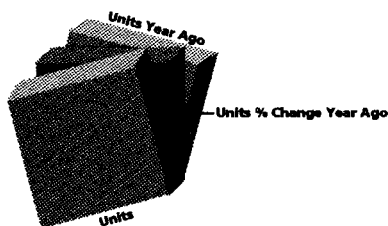


Figure 13. Calculating Units % Change Year Ago is matrix arithmetic.

You can write SQL code to create this comparison, but the code is likely to take 10 or 100 times longer to calculate the variance. As the database grows, and as the number of rows that the SQL code needs to reference multiplies, the speed of such a query becomes geometrically slower. SQL has no mechanism for the type of sequential processing that calculates a Units % Change Year Ago, or that calculates more complicated functions, like moving average or net present value.

Storing Formulas

Because multidimensional formulas are based on cubes of data, the definition of a formula is stored and dimensioned in the same way that the data is stored and dimensioned. This storage of formulas and data in the database has two advantages.

First, it ensures that formulas are consistent across the company. Second, it enables one formula to refer to another formula, which in turn can refer to another formula; this cross-referencing is a powerful feature of multidimensional formulas.

In relational environments, formulas must be stored in views, or in the spreadsheets that users create. Formulas that are stored in views are less robust than formulas in multidimensional databases because once you create a relational column as a formula in a view, you can't reference it in that view. To reference the formula, you

must create a new view. To some extent, you can use stored procedures to work around this cross-reference limitation, but each stored procedure increases maintenance.

Formulas stored in spreadsheets have a different disadvantage. They are prone to variation from user to user.

Sample SQL and Multidimensional Queries

Given the requirements for joins, views, and stored procedures in the relational data model, and given the greater ease with which the multidimensional data model handles matrix arithmetic and formulas, typical analytical queries written in SQL are more complex than the same queries written in a multidimensional DML. This lengthy SQL code affects both IS and users.

The more code you write, the more code you must maintain. And the longer and more complex the query code, the longer users wait for query execution. Let's look at two examples comparing relational and multidimensional code.

For a report that displays current market share for a product, compared to market share a year ago, Figure 14 shows the relational and multidimensional code.

RDML (SQL)	<pre>CREATE VIEW CAT_COL (MKT, CAT, PROD, PER, VOL) AS SELECT V.MKT, CP.CAT, V.PROD, V.PER, V.VOL FROM VOL V, CAT_PROD CP WHERE V.PROD = CP.PROD</pre>
	<pre>CREATE VIEW CAT_VOL_NOW (MKT, CAT, PER, VOL) AS SELECT V.MKT, V.PROD, V.PER, V.VOL FROM VOL V, CAT C WHERE V.PROD = C.CAT</pre>
	<pre>CREATE VIEW SHARE (MKT, CAT, PROD, PER, SHR) AS SELECT CC.MKT, CC.CAT, CC.PROD, CC.PER, (CC.VOL/CVN.VOL) * 100. FROM CAT_COL CC, CAT_VOL_NOW CVN</pre>
	<pre>WHERE CC.MKT = CVN.MKT AND CC.CAT = CVN.CAT AND CC.PER = CVN.PER</pre>
	<pre>SELECT S2.MKT, S2.PROD, S2.PER, S2.SHR, S1.SHR, S2.SHR - S1.SHR FROM SHR S1, SHR S2 WHERE S1.MKT = S2.MKT AND S1.PROD = S2.PROD AND S1.PER = S2.PER - 13</pre>
MDML	<pre>DEFINE SHARE FORMULA (VOL/VOL(PRODUCT CAT.PROD)) * 100. REPORT DOWN PRODUCT SHARE LAG(SHARE, 13, PERIOD) LAGDIF(SHARE, 13, PERIOD)</pre>

Figure 14. Relational and multidimensional DMLs for report about product market share.

For a report that displays the top five accounts based on total units sold last month, Figure 15 shows the SQL and MDML code.

RDML (SQL)

```
DECLARE top5_cur CURSOR FOR
  SELECT a.account_group,
         a.units,
         b.units
  FROM account_summary a,
       account_summary b
 WHERE a.product_group = 'TOTAL' AND
       a.level = 2 AND
       a.month = '1993-01-01' AND
       a.product_group = b.product_group AND
       a.account_group = b.account_group AND
       a.month = Months ( b.month, 1 )
 ORDER BY a.units DESC;
 OPEN top5_cur;
 FOR i=1 TO 5
   FETCH top5_cur INTO :lsAccount[i], :llUnits[i], :llUnitsLag[i];
 NEXT
```

MDML

```
LIMIT month TO jan93
LIMIT account_group TO TOP 5 BASED ON units
REPORT DOWN account_group sales units
      LAG(sales, 1, month) LAG(units, 1, month)
```

Figure 15. Relational and multidimensional DMLs for report about top five accounts.

Supporting Analytical Discovery

The final aspect of the multidimensional data model to consider is its effect on end-user analysis. We've talked about how users of multidimensional tools don't have to visualize the cube's structure, don't have to perform joins, don't have to rely on IS to code their queries, and don't have to wait hours for multidimensional queries to execute. But there is more at stake when users work with the multidimensional data model.

In the same way that it's virtually impossible to produce a document index with a spreadsheet, it's virtually impossible to give users what one analyst called "a holistic or a visceral sense of the business" through relational data.

Users of multidimensional tools arrive at this business sense by a process of *analytical discovery*. Discovery involves flexible displays with which users manipulate data to examine the values critical to their current focus. Discovery includes tasks such as finding top and bottom performers, making period-to-period comparisons, drilling down from aggregate data to examine its details, and performing what-if analyses.

A typical question that launches discovery might be, "What are my top 10 products based on Unit Share Change today, what were they a year ago, and how do those figures compare to the performance of my major competitor?" We've seen why such queries are difficult to write in SQL, but we haven't yet seen how users pose and answer these questions themselves with multidimensional tools.

Sample OLAP Session at Saturn Electronics

To understand the concepts and interfaces common to on-line analytical processing, suppose that you are a national marketing manager for the fictitious consumer electronics firm, Saturn Electronics.

Having completed a quarter (it's early April 1993), you routinely examine the performance of your products this year as compared to one year ago. You are looking for potential weaknesses and strengths in your product lines, which include video components, audio components, and their accessories.

Structural Elements of Multidimensionality

The multidimensional server at Saturn Electronics models the business. For example, the company distributes its products through various channels, including catalogs and direct sales. As a result, there is a Channel dimension. Dimension values of Channel include Catalog and Direct. Other dimensions at Saturn Electronics are Product, Geography, Time, and Measure.

The arrays associated with each measure are variables. The variables can be raw data, like that contained in Units, or calculated data, like that in Units % Change Year Ago.

Some dimensions in the Saturn Electronics database have *hierarchies*.

A hierarchy level is logically connected to the levels that are above and below it; data values at lower levels aggregate into the data values at higher levels. In addition to providing aggregation, the hierarchical structure imposes a *family structure* on dimension values.

For a particular dimension value, a value at the next higher level is its *parent*, and values at the next lower level are its *children*. These familial relationships, which are hierarchies, allow analysts at Saturn Electronics to access data quickly.

For example, different levels and family structures of Geography enable Saturn Electronics to report on aggregate data from stores, districts, regions, or the total company. Figure 16 shows one hierarchy in the Geography dimension:

Level	Sample Geography Value
Total Geography	Total Company
Region	East
District	New York
Account	Circuit City-NY

Figure 16. Geography hierarchy at Saturn Electronics.

Another structural element of the Saturn Electronics database is *attribute*. An attribute is a shared characteristic of dimension values at the same level in a hierarchy. For example, users might want to access all accounts by the attribute Los Angeles, or by the attribute K-Mart. In a database of jams and jellies, users might access products that share the flavor attribute strawberry.

Elements of Multidimensional Displays

The OLAP application that runs on the Saturn Electronics database organizes dimensions, dimension values, hierarchies, and attributes into the pages of reports and graphs. Although the user interfaces for multidimensional software vary from vendor to vendor, most multidimensional tools offer capabilities to select and rotate data, and to drill aggregate data down to examine its details. Codd thinks that these features supply "the multidimensional conceptual view," which he identifies in *Computerworld* as the first rule to follow when designing OLAP applications:

"Sales, for instance, can be viewed not only by product but also by region, time period and so on...Users are able to manipulate such multidimensional data models more easily and intuitively than is the case with single dimensional models. For instance, users can slice and dice, pivot and rotate consolidation paths within a model."

The user interface at Saturn Electronics, however, uniquely exploits the structure of the multidimensional server to ease the data selection, layout, and analysis that define the discovery process. As you compare multidimensional tools, you will see different approaches to the user interface.

Dimension Tile Panels

In Figure 17, the Sales, Quota and Variance report at Saturn Electronics is reduced to display just its title, and its top and bottom *dimension tile panels*.

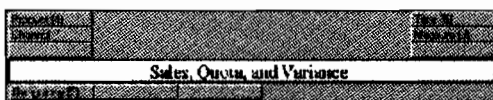


Figure 17. Dimension tile panels for five dimensions.

These panels contain the *dimension tiles* named Product, Channel, Time, Measure, and Geography. Figure 18 shows that Geography in the bottom dimension tile panel makes Geography values the rows of the report. Time and Measure tiles in the upper right of the tile panel make those dimension values the columns of the report. And Product and Channel tiles in the upper left of the tile panel create the pages of the report. To rotate a report or graph layout, you use the Windows features drag, drop, and swap to move dimension tiles within the panels.

Page Controls

For those dimensions that create the pages of the report or graph, there are *page controls*, one control for each dimension in the top left panel. For example, in the report in Figure 18, the page controls are black.

Sales, Quota, and Variance					
	January 1993			February 1993	
	Sales	Quota	Variance	Sales	Quota
Total Company	\$41,750,187	\$42,366,890	(\$618,299)	\$42,149,380	\$41,624,747
Channel	\$12,138,990	\$12,585,740	(\$446,750)	\$12,317,160	\$11,842,747

Figure 18. Page controls for Total Product and Total Channel.

At any given time, a page control contains the name of the page dimension value that is currently displayed in the graph or report. For example, the page in Figure 18 displays values for Total Product and Total Channel for Sales, Quota, and Variance in the months of 1993.

By using the page control to select a new product value, you change the report page to display the slice of data that is Audio Division (Figure 19).

Sales, Quota, and Variance					
	January 1993			February 1993	
	Sales	Quota	Variance	Sales	Quota
Total Company	\$21,354,760	\$21,708,790	(\$402,030)	\$21,551,440	\$22,448,747
Channel	\$6,443,193	\$6,718,082	(\$274,889)	\$6,538,910	\$6,643,747

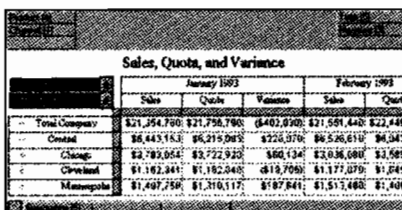
Figure 19. Product page control changed to Audio Division.

Drilling

When working with hierarchical data, you can drill down on aggregate data to see the levels of its detail. In reports, users drill on the row values.

When a dimension value is preceded by a plus sign, you can drill that value down to display its children. When a dimension value is preceded by a minus sign, its children are already displayed. Clicking on one of the two symbols drills values up or down.

For example, if you drill down on Total Company in the Sales, Quota, and Variance report, you see indented regions. If you drill down the Central region, you see its indented child values, which include Chicago, Cleveland, and Minneapolis (Figure 20):



	January 1993		February 1993		
	Sale	Quota	Variance	Sale	Quota
Total Company	\$21,354,700	\$21,736,790	(\$402,090)	\$21,551,440	\$22,448
Central	\$6,443,140	\$6,219,062	\$224,078	\$6,526,610	\$6,043
Chicago	\$2,783,054	\$2,722,920	\$60,134	\$2,936,080	\$3,569
Cleveland	\$1,162,841	\$1,162,040	-\$800	\$1,177,079	\$1,045
Minneapolis	\$1,497,245	\$1,310,117	\$187,128	\$1,513,451	\$1,409

Figure 20. Total Company and Central drilled down.

Selector Tool

Users at Saturn Electronics choose data to display in reports and graphs by using a *Selector* tool. Figure 21 shows the Selector dialog box for the Geography values that are displayed in Figure 20.

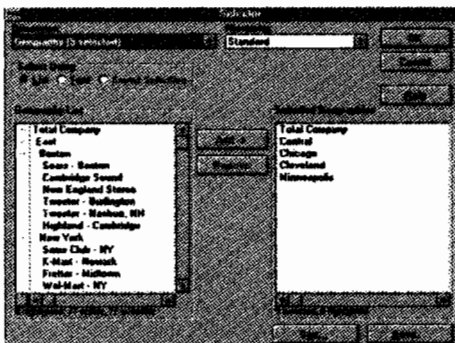


Figure 21. Selector dialog box.

The Geography List box, with its drill symbols next to indented dimension values, reflects the family structure of the dimension.

Evaluating Sales: A National Perspective

To begin your quarterly analysis, you want the big picture. In a multiple bar graph, you examine Sales of Total Product in all channels across all regions Year-to-Date, compared to one year ago. You are happy to see that Sales has increased for Saturn Electronics from the same period last year (Figure 22).

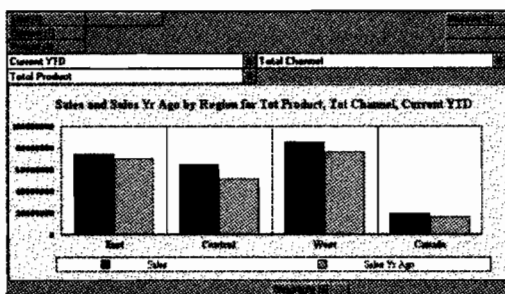


Figure 22. Examining Sales by Region.

Next, you turn to another page of the report. You wonder if the product divisions Audio and Video have contributed equally to this year's success. Unfortunately, the Audio Division in the East has declined (Figure 23).

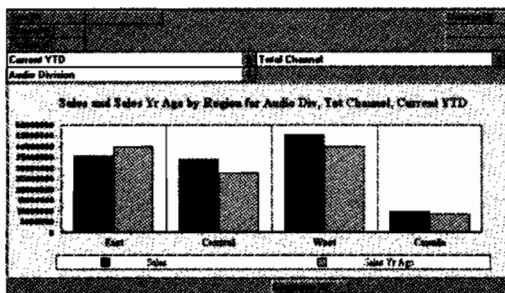


Figure 23. Eastern Audio Division decline in Sales.

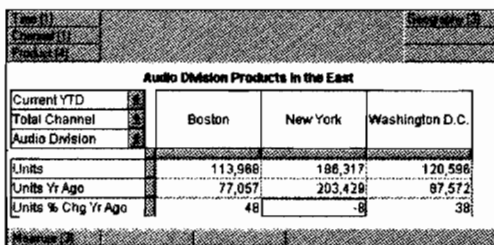
To see the specific numbers of the decline, you create a report from the above graph. It displays the actual values for Sales and Sales Year Ago (Figure 24).

Product Performance, Audio Div, Tot Channel, YTD				
Current YTD	East	Central	West	Canada
Total Channel				
Audio Division				
Sales	\$35,897,432	\$34,142,300	\$45,733,360	\$9,833,632
Sales Yr Ago	\$40,037,880	\$27,763,180	\$40,137,280	\$9,725,084

Figure 24. Report based on graph in Figure 23.

Evaluating Units in the East

Next, because you want to focus on the East, you display only the children of the Eastern region for your Geography values. To examine Sales in terms of the Units sold from various product lines, you display the measures Units, Units Year Ago, and Units % Change Year Ago. Units in New York has declined (Figure 25).



Audio Division Products in the East			
	Boston	New York	Washington D.C.
Current YTD			
Total Channel			
Audio Division			
Units	113,966	196,317	120,596
Units Yr Ago	77,057	203,429	87,572
Units % Chg Yr Ago	48	-8	38

Figure 25. Audio Division Units in child values of East Region.

You wonder what's happening to Audio Division products in New York.

Evaluating Products Sold in N.Y. Accounts

You decide to examine the performance of key accounts, such as Wal-Mart and K-Mart, that usually contribute most to your revenue. To focus on parent accounts, you use the Selector to study information along a different hierarchy path. You change the Geography hierarchy from Standard to that of Parent Account.

You also limit the time period to the first quarter, because you don't want April values as part of your analysis. And you decide to examine only the Direct Channel, since that is the major sales channel. IS, of course, doesn't have to code these selections for you.

From your parent accounts, you want to select only those in New York. You therefore use the Attribute tool of the Selector to select the parent accounts that share the Geography attribute New York. The application finds six, which are listed in Figure 26 under **Selected Geographies** in the Selector dialog box.

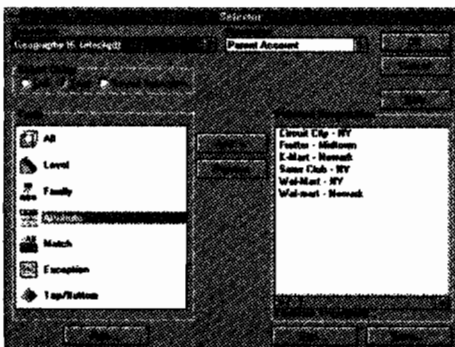


Figure 26. Parent accounts selected by attribute New York.

Finding Top and Bottom Performers

Of those six, you decide to narrow your focus to the two best and two worst performers. To do this, you use the Top/Bottom Selector tool, which is listed at the bottom of the Selector Tools box. The Top/Bottom tool is especially useful when you are looking for best and worst performers among hundreds of thousands of entries.

You fill out the Select Top/Bottom dialog box to keep, from the New York list of six accounts, the two best and worst performers based on the following: Units % Change Year Ago in the Audio Division for the Direct Channel during the first quarter of 1993. The application describes your selection at the bottom of the dialog box (Figure 27).

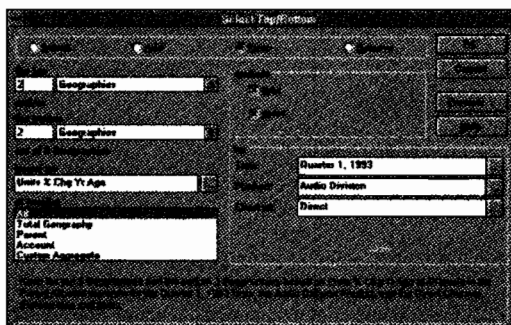


Figure 27. Top/Bottom selection dialog box.

Returning to the report, you swap the Geography and Measure tiles. The resulting display sorts the top two and bottom two Geography values among New York Parent Accounts. Fretter is doing best; Circuit City is worst (Figure 28).

Top/Bot NY Accts Based on Audio Div % Chg Yr Ago			
Quarter 1, 1993	Units	Units Yr Ago	Units % Chg Yr Ago
Direct			
Audio Division			
Fretter - Midtown	28,508	19,781	34
Wal-Mart - NY	10,281	9,053	13
K-Mart - Newark	8,336	9,388	-11
Circuit City - NY	45,452	63,842	-28

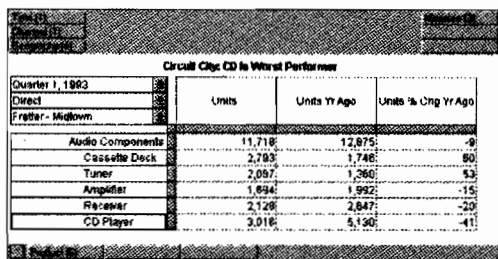
Figure 28. Top/Bot NY Audio Division performers; Geography rotated to rows; Measure rotated to columns.

Focusing on Account, Drilling on Product

Next, you want to know what products, if any, are particularly low sellers at Circuit City. By swapping the Product and Geography tiles, you lay out Product as the row dimension and add Geography to the two other page dimensions.

The immediate children of Audio Division are Audio Components and Accessories. You are more interested in Components because they are the larger ticket items. You therefore drill down Audio Division to its child value Audio Components, and then drill down Audio Components to create a report of the tuners, CD players, and receivers whose sales in Circuit City may have flagged.

Finally, based on Units % Change Year Ago, you sort, high to low, the five products that aggregate into Audio Components. CD Players are performing worst (Figure 29).



Circuit City: CD in Worst Performer			
Quarter 1, 1993	Units	Units Yr Ago	Units % Chg Yr Ago
Direct			
Fratter - Midtown			
Audio Components	11,718	12,875	-9
Cassette Deck	2,793	1,748	60
Tuner	2,097	1,360	53
Amplifier	1,894	1,922	-15
Receiver	2,128	2,847	-25
CD Player	3,016	5,130	-41

Figure 29. Products sorted high to low by Units % Change Year Ago.

And the Analysis Goes On

Before your analysis is done, you use similar reports, graphs, and multidimensional tools to track the performance of CD Players nationwide, and to study the performance of CD Players in other Circuit City accounts. You also examine the types of promotions for CD Players that other parent accounts have successfully run, and compare those promotions to what Circuit City in New York ran.

Since you find that Circuit City promoted its CD Players very differently from other New York accounts, you conduct a promotion simulation. It tells you how many more CD Players, at a cost either to Saturn Electronics or to Circuit City, might have been sold had a particular promotion run for a specified period at Circuit City in New York.

User Independence

Using these graphs, reports, and tools requires no personalized programs for the Saturn Electronics national sales manager. The application used by the manager is one of two products. It is either a pre-built application that is sold by a multidimensional software company, or it is a custom application built by IS personnel who used a multidimensional DML.

Either way, all Saturn Electronics analysts can use the same dimensions, customizable reports and graphs, and multidimensional tools. What analysts find during the initial stages of studying Saturn Electronics data leads them to what they examine in final stages; this is the nature of ad hoc analysis that multidimensional tools support.

How important is this type of ad hoc analysis to the future of business, and how many users are likely to need to perform it? In a Gartner Group *Conference Presentation*, Program Director Howard Dresner addressed these issues:

“As the next century approaches, business intelligence (BI) systems must become pervasive, leveraging data to a competitive advantage. Instead of a small number of analysts spending 100 percent of their time analyzing data, all managers and professionals will spend 10 percent of their time using BI systems.”

The father of relational technology agrees. Codd wrote in *Computerworld*:

“Ultimately, an enterprise's ability to compete successfully will be in direct correlation to the quality, efficiency and pervasiveness of its OLAP capability. It is, therefore, incumbent on information technology groups to prepare for OLAP and to provide rigorous support for it in their organizations.”

Further Reading

If you are interested in learning more about the advantages of multidimensional environments, several publications may interest you. For information about traditional database management software and the technologies behind it, we recommend *Understanding Database Management Systems* by Robert M. Mattison (McGraw-Hill, New York, 1993).

For reprints of the following articles that were referenced in this paper, call IRI Software at (800) 765-7227:

Codd, E. F. ; Codd, S. B.; Salley, C. T. "Beyond Decision Support." *Computerworld*. Vol. 27, No. 30. 26 July 1993.

Dresner, Howard. "Business Intelligence: Competing Against Time." *Office Information Systems Conference Presentation*. The Gartner Group. 5 May 1993.

Dresner, Howard. "Multidimensionality: Ready or Not, Here It Comes." *Office Information Systems Research Note*. The Gartner Group. File: Technology, T-MD-1137. 3 June 1993.

Hurwitz, Judith S. "Modeling Complex Multi-Dimensional Information." Hurwitz Consulting Group *Client/Server ToolWatch*. Vol. 1, No. 10. Dec. 1992.

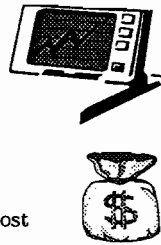
"IRI Software." Aberdeen Group *Profile*. March 1994.

Ricciuti, Mike. "Multidimensional Analysis: Winning the Competitive Game." *Datamation*. 15 Feb. 1994.

"The Trouble with SQL." *Office Information Systems*. The Gartner Group. File: Technology, T-240-759. 6 Aug. 1990.

What is a Mission Critical Computing Environment?

- Includes customer environments where computing operations are critical to business success
 - > financial transaction processing
 - > manufacturing control
 - > patient monitoring
- Often involves lost revenue or irretrievable cost



 HEWLETT
PACKARD

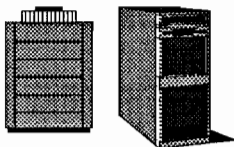
HP
TAK OH

As you know, rapid change is occurring in our business environment. To be competitive in today's worldwide markets, companies are showing greater reliance on the use of technology. Successful implementation and management of computer systems, networks, software applications and information databases are integral to business effectiveness. Companies increasingly depend on computers to run business operations, improve productivity and reduce costs.

Mission Critical computing environments involve configurations where computing operations are critical to business success. Representative examples include businesses such as telesales/order processing, financial transaction processing (i.e., trading, banking), health care applications such as patient monitoring and manufacturing control systems. Frequently these operations run 24 hours a day up to seven days per week. At the peak of operation, a down computer system can translate into lost or dissatisfied customers, lost revenue opportunities, increased costs and sometimes may involve monetary penalties and loss of life.

Industry Trends

- Migration from mainframe to client/server technology
- Greater dependence on information
- Role of information superhighway
- Complex multivendor environments
- Security concerns



 HEWLETT
PACKARD

INTEGRIS
164 D4

The trend to shift computing applications from mainframe-based systems to client/server, multivendor-based environments, presents an increasing need for management of critical computer centers. In addition, mission critical environments are evolving out of applications which have traditionally been implemented in business server computer environments. Providing this level of support in a distributed environment has a new set of challenges. This presentation describes how you can better address the mission critical needs of a client/server environment.

Planning Throughout All Phases of Operations

Planning for
System Operations

System Changes
& Upgrades



System
Installation
& Implementation

System
Operation

 HEWLETT
PACKARD

 DIGITAL
WORLD

One key aspect of successfully supporting a mission critical computing environment is developing appropriate plans to provide technical guidance for all phases of computer operations. How is successful operation defined? Typically, success is achieved if operational metrics are met. Often, this is defined in terms of system availability, number of preventable problems, number of hardware failures, number of software failures and similar measures.

In the client server environment managing operations is complicated by the multiple vendors found in distributed networks. Also there is less rigidity in system admin procedures and often times fewer tools available. All of these issues create additional challenges in providing mission critical support.

Mission Critical Planning Process



HEWLETT
PACKARD

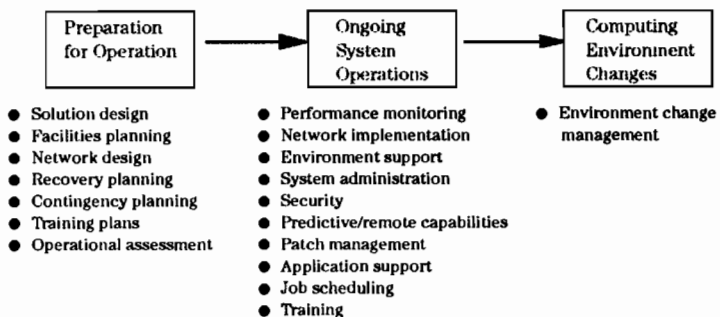
DIFFERENT
THOUGHT

Prior to a major computing purchasing decision, several areas of structured planning should be considered. Planning activities include a properly designed computer room, determination of system architecture, application loading, database structure, security procedures, network topography, user profiles and access, and system administration procedures. In addition, recovery and backup planning and processes should be defined. These activities are sometimes undertaken by the customer. In some situations the vendor or an independent consultant is engaged to provide the complete or a partial analysis. In addition to planning and design, appropriate training is crucial to success.

Often a computer vendor or consultant assisting in a customer's mission critical installation will perform an operational assessment to understand the customer's organization structure, vision, business plan and critical success factors as well as review the customer's operation plans as outlined above pertaining to recovery, administration, networking, security, database concerns, etc.

Plans should be developed for a smooth migration of operations at system installation and startup to ensure minimal unavailable computer resources. User training needs should be assessed and procedures for all areas of system operation should be defined.

Mission Critical Planning Process



 HEWLETT
PACKARD

HPD000000
10/01/00

During on-going data center management, well defined system operating procedures are necessary as well as appropriate security measures. While the goal of a mission critical support program is to prevent failures, in the event a failure does occur, defined, agreed upon escalation procedures should be established with the computer vendor(s). Also, throughout operations, regular review of system performance, capacity, network implementation, job scheduling and application loading is advised. On-going training needs should be periodically assessed. Most vendors have tools to help evaluate system performance and assist in system administration functions.

Computing Environment Change Management

- Situation: A high technology semiconductor chip manufacturing company is using Hewlett-Packard Unix business systems in an IT, mainframe, data center environment
- Hewlett-Packard supports this environment with high availability, personalized, technical support
- HP provides environment change management planning to ensure successful implementation of each scheduled change
- Includes hardware installs, software updates, upgrades and changes, release planning & testing



Changes in the computer environment require documented plans and guidance through the change cycle to ensure effective implementation. Typical changes include software update/release planning and implementation as well as implementation of hardware changes. Especially across a distributed client server environment, managing changes, upgrades, etc. becomes critical to ensuring network and system availability. HP can offer very personalized services to mission critical customers to help manage the change elements important to your environment.

Enhanced Support Delivery

- Immediate attention to critical customer problems
- Assigned problem manager
- Immediate escalation procedures
- Support for unique software applications
- Patch update process and tools
- Specialized call handling
- 24 hour-a-day, 7 day-a-week coverage
- Failure verification



 **HEWLETT
PACKARD**

**INTERIX
SUN ON**

Often, vendors provide enhanced support delivery capabilities, specifically targeted toward mission critical environments. These may include personalized support, an assigned problem manager, specialized call handling, failure verification and root cause analysis, support for unique software applications, predictive and remote support capabilities, special processes for installing patch updates and tools to maximize system availability.

In HP's case, for severe problems, we provide these immediate response deliverables to a growing customer base. These include immediate access to a system level problem manager, 24 hrs a day, 7 days a week. This problem manager drives problem resolution and has priority access to additional technical resources to solve problems more quickly and minimize downtime. Some of these customer agreements also provide enhanced coverage for unique software applications.

Flexible Support to Meet Your Needs

- Situation: Some customers desire assistance managing their data center operations
 - system administration functions
 - planning
 - change management
- Hewlett-Packard provides full time on-site system engineers at several Fortune 100 companies
 - a computer manufacturer
 - a consumer products company



OFFENSE
104 CM

On-site technical consultants and system engineers are common in these environments. Several situations involving on-site technical resources range from consulting and training to designing system administration procedures and operating the computer data center for the client. Generally, these on-site technical resources complement the capabilities and skill set of the customer's staff.

In many situations, HP can provide dedicated on-site assistance. These engineers become an integral part of the customer organization's team and are immediately able to deal with a loss or degradation in system availability.

Rely on Hewlett-Packard

- Commanding presence in client/server solutions
- Respected, market-leading computer vendor
- Worldwide
- High quality products
- Technology leader
- Customers rate HP the best in support



 HEWLETT
PACKARD

HYPERC
THE CH

In this challenging, fast paced, world of information management, it is important to carefully select the computer vendor based on reputation, technology, worldwide presence, common operating environment, ability to respond quickly to changing market forces and flexibility to engage in partnering activities. Because of HP's investment, commitment and reputation for customer support we can offer real value as your mission critical support partner.

Retrofitting Your HP 3000 Applications to the MS-WINDOWS Standard

PAPER NUMBER: 6006

**Ross Hopmans & Joe Grimm
Brant Technologies Inc. & MiniSoft Inc.
458 Elizabeth Street
Burlington, Ontario
CANADA L7R 2M2**

tel: (905) 639-7220

fax: (905) 639-7287

INTRODUCTION

The widespread use of MS-WINDOWS has brought a vast array of software applications that support advanced graphics, text processing and visually oriented features to the desk top. Many new representations of data such as imaging, sophisticated business graphics and desk top publishing are now commonplace on the standard PC platform. This paper explores the alternatives available to software developers for incorporating the flexibility and sophistication of these MS-WINDOWS applications into production software on the HP 3000.

Using what is generally known as client/server technology, we are attempting in one way or another to link the data and applications in an HP 3000 production environment with the users' desk top. Many users utilize "co-operative processing" by working with both HP 3000 and PC applications. Unfortunately, these users are often frustrated by the manual procedures they must undertake and the duplication of effort that is generally required today to integrate their HP data with their PC applications. The implementation of client/server techniques would allow the co-operating processes to be linked with each other so that

complementary software products make users more efficient and less vulnerable to errors and oversights.

BACKGROUND

Although some very sophisticated tools such as Forest and Trees, and Powerbuilder are available for the development of client/server systems, our focus is on the simpler solutions that are quick to develop, easy to learn and use, and take advantage of your existing tools and technology. Although we do not question the quality of solutions that can be developed using Forest and Trees or Powerbuilder, these tools require a major investment of time and money. Our emphasis here is on the benefits and efficiencies of simpler client/server solutions that you may be able to develop in one or two days. We seek to transform your Excel spreadsheets from standalone tools into clients, served by the HP 3000.

Products like Faces for Windows (or HP NewFace/3000 allow you to add a GUI interface to HP 3000 applications. Faces adds a layer on top of the terminal emulator and it is useful for giving all of your applications the same look and feel, although the applications remain host based. Our goal is to push the host into a server role to add power to the PC applications.

We should mention also that we feel the UNIX/SQL market is well served by many tools and thus we will restrict our discussions here to the HP 3000 with TurboImage data bases, KSAM and MPE files. Further, we assume an MS-WINDOWS environment on the PC with LAN connections; though not necessarily a PC Server or LAN operating system.

APPROACH

Our approaches range from "retrofitting" your HP 3000 applications in order to take advantage of available MS-WINDOWS features and functions, through to new development, and we will show why you would choose one approach over another. Although we imagine that many new applications will be written to client/server standards, it can be far more cost effective to update your tried-and-true HP 3000 production systems using our proposed techniques, rather than having to re-write them.

We will consider three alternatives available to the developer when considering the development, redevelopment or retrofitting of HP applications to a client/server architecture. These approaches are "screen-scrapers" for retrofitting; a PC toolkit with a customizable HP 3000 server program either for new development or redevelopment; and a high-level development environment for new development.

SCREEN-SCRAPER

The first alternative is what we term "screen-scrapers" technology. We would not recommend this as a universal approach but it may provide a solution in specific cases as it requires absolutely no additional purchases. A screen-scrapers system uses a terminal emulator, DDE and the macro capabilities of your PC application. Connectivity to the host along with access to host application programs and data are all provided through the terminal emulator.

By way of an example, let's use an actual environment. The user has a typical host-based HP 3000 application developed using PowerHouse, although the development tool is not germane to the example. The application is a medical system containing hospital records. In addition, they have a MS-WINDOWS based imaging system that contains the actual documents. The users locate documents in the PC imaging system and then log on to their HP to locate the header record, and add or update details from the document as appropriate.

Users were interested in linking the two systems to eliminate duplication of entries. Using MiniSoft's WIN92 MS-WINDOWS terminal emulator with its available command language and DDE facilities, the MIS staff have linked the HP 3000 application with the PC imaging system. The imaging system issues DDE commands to communicate with the terminal emulator, instructing the PowerHouse application to locate the relevant record. Further, the imaging system issues appropriate commands to the HP application to add or update the Image data base and it pulls data off the HP screen to add to or update its own information store. In this case, the PC acts as the server and the HP 3000 is its client through the terminal emulator.

This is a good example of the use of "screen scraper" technology to improve the user environment without altering our HP 3000 applications. Note that we are not adding new functionality. Rather, we are tying two separate applications together to improve both, allowing each to share the other's information. From the users' point of view, the two applications are co-operating to reduce duplication of entries and manual intervention. Data is transferred by "scraping" it off the screen.

MiniSoft's PerfectDesk is another good example of a product that uses screen-scraper technology to add value to an HP 3000 host application, in this case HP Desk. PerfectDesk allows the user to use WordPerfect or MS-Word as his word processor for creating messages. PerfectDesk makes use of MiniSoft's terminal emulator to transfer files, process script files and run the word processor with macros.

The investment required to implement screen scraper technology is minimal. However, it is not truly client/server. This approach is available to any PC applications that support DDE and provide a macro capability. In addition, your terminal emulator must also support DDE and provide a script language. The technology is available today at a low cost. It requires little development effort and no modifications to your HP 3000 applications with a potentially significant payback.

CLIENT/SERVER TOOLKIT

Our second approach is far more universal and uses true client/server technology with a client/server toolkit, such as MiniSoft's recently released *MiddleMan*. Let's separate the toolkit approach into its client and server components and then examine how they would be used.

A "stock" server program is supplied with *MiddleMan* to run on the HP 3000. The server's job is to access TurboImage data bases, KSAM and MPE files using instructions from its clients. For example, the client may want the current account balance from the Accounts Receivable data base for a particular customer. The client would send the request to the server and the server would carry it out and return the data or an error code.

At the lowest level you can imagine the client sending a TurboImage

intrinsic call to the server to implement. However, a server should have a higher level language and syntax to make it more convenient for passing messages. It should also have to have a "dictionary" for data security reasons.

A server is required on the host for access to the host data bases and files. A third-party toolkit provides a host server and a language to access it, in much the same way as SQL. Many third-party report writers employ a server on the HP 3000 to provide read access to data. Obviously the server program would have to allow for add, update and delete access as well, under the security restrictions contained in the data dictionary.

In most client/server environments, the server functions simply as a data repository and access engine. The logic for the software application resides on the PC. However, there are situations where it makes sense to associate some of the processing back to the server. For example, your client may be a spreadsheet that wants monthly totals for each product line, but the data resides in a sales transaction data set on the HP 3000. Rather than transfer all detail data down to the PC, it makes sense to do the totaling inside the server program and simply pass data back to the client in the format it is expecting. We strongly believe that the ability to customize the server is an important feature in a useful toolkit. Customization like this example would be easy to implement and leads to significant processing and data transfer efficiencies. However, it is important that separate clients can simultaneously access customized and non-customized servers.

One of the most important aspects of client/server computing is to provide robust connectivity between the PC and the HP 3000. We see two approaches available for connectivity. If you were developing your toolkit in-house you may choose a low level approach to explicitly access the WinSock interface on the PC and the NetIPC interface on the HP 3000. You would use this interface to send messages back and forth between the client and server.

However, developing this type of connectivity interface is not a trivial task due to the complexities of dealing with the network interface and timing issues, among others. As a result, developers will likely choose to work at a higher level through a third-party toolkit such as MiniSoft's MiddleMan. In

addition to providing file transfer and the server as mentioned previously, such a toolkit would provide the mechanism to accommodate client/server connectivity along with a simple means for the user to pass messages.

The PC acts as the client and all the processing logic resides on the PC, other than that associated with the customer server routines mentioned above. We see two ways that the HP 3000 community will use our client/server toolkit: called directly from PC applications such as MS-Word or Excel; or embedded within a PC application developed in house using a tool such as VisualBasic or Visual C++. The big advantage of accessing the client/server toolkit from your PC applications is that the integration can be achieved with minimal effort to greatly enhance the functionality while eliminating user intervention and duplication of entries. It can make a broad range of PC applications more useful by giving them access to HP 3000 data.

Let's say your user has an Excel spreadsheet that requires sales data from the HP 3000. You could provide the user with a macro that would automatically connect to the HP 3000, extract the appropriate data from the TurboImage sales data base, and populate the spreadsheet. In this case the macro would define the object slots by calling DDE-POKE to load values for the data base, data set name, data item list, etc. MiddleMan would then instruct the server to open the data base and find the data, then call DDE-REQUEST to transfer the data from the HP 3000 into the macro. In this case you have no need to develop a user interface, no programming and your users have access to a tool that is very familiar to them.

Any PC application with macro capabilities and a DDE interface can now become HP 3000/TurboImage enabled. What's more, customizing the server allows us access to specialized processing on the HP 3000 and we could even permit the user to update the TurboImage data from their spreadsheet or other PC applications if appropriate.

We believe that customized development on the PC will probably be done in a visual development tool such as Visual Basic or Visual C++. These programs allow the developer to communicate with the toolkit for server access, as well as other PC applications, through the DDE interface. This

enables access and updates to PC data as well as host data.

Visual Basic in particular is very easy to use and it can create useful windows applications with minimal development effort that combine PC and HP 3000 data. In fact, MiddleMan includes a Visual Basic custom control for simpler and more efficient access to the server.

The bottom line is that you now have a way to access your HP 3000 easily at a high level using a standard interface. Your users' PC applications now have customized access to the HP 3000 to greatly enhance their value.

HIGH LEVEL DEVELOPMENT

The third approach to client/server development is at a higher level than that offered through Visual Basic. We envision a third-party product which provides both the client and the server. This type of product would have similar capabilities to those offered in some 4th generation languages in that it would hold data about the data items themselves. To design a screen, the user would simply select which data items should appear and the product would use its information on the data to decide properties such as how the field should be represented (ie radio buttons, check box, pull-down menu, among others) and which field edits should be applied. The product would also provide pre-defined logic for certain actions which could be modified as appropriate, and automatic messaging with the server. It makes sense that the pre-defined logic could be modified as appropriate. We hope that by the time this presentation is delivered we will be able to discuss some of the design considerations in such a high-level tool, currently in the early stages of development at MiniSoft.

CONCLUSION

Our purpose is to add functionality to HP applications by taking advantage of available technology on the PC platform. These may include adding, for example, verbal instructions instead of help screens, or integrated graphics and imaging for enhanced presentation value. The secret to success in this area is through simple but effective "middle-ware" client/server tools.

We have outlined three approaches to accomplish this goal. Whichever

approach you choose to move towards client/server technology, you will require flexibility and simplicity in order to support the continued evolution inherent in this environment.

Our challenge is to transform your workhorse HP 3000 into an engine that provides services to a great variety of client applications, while preserving your investment, security and control. We think the approaches outlined in this paper can provide high-payback solutions for a wide variety of user needs.

*Ross Hopmans, Brant Technologies Inc.
Joe Grimm, MiniSoft Inc.*

June 30, 1994.

All trademarks mentioned are the property of their respective owners.

#6007

Tuning the HP-UX Series 800 File System

Glen Johnson
Computer Solutions, Inc.
120 East Marks St. #225
Orlando, Florida 32803
407-649-0123 or 512-343-6634
grjohn@bga.com

The HP-UX high performance file system (HFS) creates a lot of structure on your disks in an attempt to provide a flexible and fast data storage and retrieval environment. For many applications this structure may decrease instead of increase system performance. This presentation will provide a limited description of the HFS structure, concentrating on the areas we, as users, have some power to control.

Basics

Much of the following discussion will concern file sizes. The sizes are measured in bytes. A byte is just a character, such as a letter or digit. A kilobyte is a thousand bytes, but this is a computer thousand. A computer thousand is 1024, not 1000. This is because computers count in base 2 instead of base 10 and 2 to the 10th power, which is the first power of 2 that results in a number greater than decimal 1000, equals 1024. The letter K is used as shorthand for kilobyte. Similarly, a megabyte is a computer million bytes, 1024 times 1024, and a gigabyte is a computer billion bytes, 1024 times 1024 times 1024. The letters M and G are used as shorthand for megabyte and gigabyte.

Blocks

The HP-UX disk management system divides the disk into fixed length number of bytes called blocks. Historically UNIX block sizes have been 512 or 1024 bytes. The Series 800 HFS, however, permits the user to choose a block size of 4K or more. The rationale behind

the use of a smaller block size is that most files on computers are less than 10K bytes long. If a large block size is used, more disk space would be wasted. For example, if a file is 2000 bytes long, the smaller block size would only require 2048 bytes of disk space (two 1K blocks), while the 8K block size system would require 8192 bytes (one block).

The problem with the smaller block size systems is that even if 80 percent of the files on your system are less than 10K bytes long, the other 20 percent of the files are accessed 80 percent of the time. It is the large files that your programs continually use. As we will see later, the larger the file the longer it takes to access the data.

The term block is used throughout the UNIX environment without much concern for clarity. In the utilities `df` and `cpio`, values are given in blocks but these are 512 byte blocks. In some third party supplied file structures the file is logically divided into sections called blocks but these are 1024 byte blocks. In this paper, unless otherwise specified, a block will mean a file system block.

Files

From your exposure to other computer systems or even if you write COBOL programs, you may be familiar with the concept of relative record files. These are files that are broken into fixed length records that are accessed by the record number. All HP-UX files are essentially relative record files with a record size of 1 byte.

A file, to the file management system, is just a bunch of bytes. It doesn't care what the byte values are or if they are related to each other. The program just tells the file system it wants to write X number of bytes to location Y of the file. You can write one byte to location one million of a newly created file, and it will only have one block allocated for it. An `ls` command will show the file only contains one byte. The HFS does not add any structure to the internals of the file.

File System

HP-UX does add structure to the disk in order to

help manage your files. This structure is called a file system. A file system can be a portion of a disk, an entire disk or portions of multiple disks. The maximum file system size on HP-UX systems is four gigabytes. Most other UNIX systems only permit file systems to be two gigabytes. This is due to the basic design of the UNIX file system. Some of the structures associated with the file system have a field that contains the byte offset into the file system. Since this field is 32 bits in length and is sign sensitive, only 31 bits can be used to represent a byte location. Two to the 31st power is 2 gigabytes. Since a file system can only be 2G, the largest file on a UNIX system is only 2G. Even on the HP-UX system, the largest file is limited to 2G. This means that if your programs use 500 byte records, you can at most have 4 million records in the file. By using a data base subsystem that uses character compression, this limitation can be circumvented. I have encountered a situation where I wanted to convert a file from the TISAM file structure to C-ISAM. The problem was that the file contained 5 million 1000 bytes records. Using TISAM, which has compression, the data file only took up 800 megabytes. The C-ISAM data file would have required 5 gigabytes. We ended up staying with the TISAM structure and using some other method of solving our problem.

Inode

The structure that keeps track of all of the file information you see when you perform an `ls -al` command is called an inode. The inode contains file information, such as, access and modified times, size, access rights and who owns it. It also has fields for the blocks allocated to the file. Since the inodes are fixed length structures there are a limited number of fields available for pointing to blocks. Specifically, in HFS, there are 14 fields. Since files must be permitted to grow up to 2 gigabytes, some of the 14 fields are used to point to blocks that contain the block numbers allocated to the file.

On other UNIX systems that use 1K blocks, the first 10 of 13 fields contain the actual block numbers of the blocks that contain the first 10K bytes of data. The eleventh inode block field contains the block number of a block that contains the pointers to the actual data blocks.

Since a block is 1K long, and the pointers are 4 bytes long, 256 block numbers can be stored in a 1K block. Therefore, the eleventh inode block field identifies bytes 10K through 266K of the file. These bytes are accessed by what is known as a single indirect access. Byte 266K plus 1 of a file requires the twelfth inode block field.

This field contains a block number of a block that contains block numbers of blocks that contains block numbers of blocks that contain actual file data. This is a double indirect access and is used for all bytes in the file greater than 266K and less than 64M + 266k. Therefore, in order to read or write byte 266K + 1 of a file, three disk I/Os must be performed instead of just one.

Any byte greater than 64M + 266K requires four disk I/Os. The thirteenth inode block field is used for these bytes. This field contains a block number that contains pointers to more double indirect blocks. Thus, the thirteenth inode block field is a triple indirect pointer. The reason there are only 10 direct pointers is because when the designers of UNIX looked at their existing systems they determined that the large majority of files were less than 10K bytes long. Therefore, they wanted to make access to those files as fast as possible. Also, if the block size was larger than 1K, too much disk space would be wasted. Remember, these decisions were made in the mid 70's when disk space was a very big concern.

This design made access to the majority of the files on file systems fast at the expense of the few large files. But, it is the large files you use more than the small. Your customer master and inventory files, I assume, are much larger than 10K. For this reason, HFS permits you to define the file system block size. Series 800 HFS even forces you to pick a good size (4K or larger) so all of your files can be accessed via at most a double indirect pointer. This is a big advantage of using HFS over other UNIX file systems. To get around the problem of wasting up to 8K-1 bytes of disk space at the end of each small file, the last block allocated to a file, that is less than 13 blocks long, uses a user defined fragment of a block.

Fragment

A fragment may be 1K, 2K, 4K or 8K long. On a file system with an 8K block size, if a file is 8K+1 byte long, the first 8K bytes will be placed in a block and the last 1 byte will be placed in a fragment. If the fragment size is 1K (the default), then only 9K bytes of disk space will be used by the file instead of 16K. The other 7K of the block the fragment occupies will be used for fragments of other files. This algorithm permits you to have your cake and eat it too. You have faster data access due to the large block size and you don't have much allocated but unused disk space.

Only if a file requires less than 13 blocks will the last block be allocated as a fragment. The 13th block, when required, will be allocated as an entire block. Once a file gets this large, the percentage of unused disk space allocated to the file becomes small. The purpose of fragments is to keep disks that contain a lot of small files from becoming needlessly full.

Cylinder Group

The HFS has another feature other UNIX systems don't which is the concept of a cylinder group. On other files systems all of the inodes are kept at the beginning of the file system and all of the free blocks are kept in a global free list. When a file is opened, this structures forces the disk to go to beginning of the file system to read the inode and then go some place into the file system, possibly all the way to the end, to read the first block. The free list keeps track of all of the blocks that are available for use. This starts out as a nice orderly list but as files are added and deleted from the file system, the order of the blocks within the free list can become very disorganized. It is not unreasonable for the first block of a newly created file to be near the beginning of the file system and the next block near the end. The cylinder group structure corrects these problems.

In the HFS, every X number of disk cylinders is handled like a miniature file system. X is a user definable number between 1 and 32 inclusive. Each group has it's

own set of inodes and manages its free space. When a directory is created it is placed in the cylinder group that has the greatest number of free blocks and the fewest directories. When files are created, they are placed in the same cylinder group as their parent directory (if possible). This algorithm ensures locality of reference for small files. That is, the internal structures required to manage a file system and the files data are physically close to each other on the disk. This makes stand alone access to the files data fast.

The other problem that HFS corrects concerns the handling of free space. Instead of a global free block list, each cylinder group has its own free block bit map (an array of words in which each bit represents a fragment in the cylinder group). The use of a bit map ensures that when a file grows and requires another block, the "best" block can be allocated. The best may or may not be the physically next block depending on some user definable parameters and what is available. But, at least some intelligence will be used to make the choice instead of just using the next block on the free list.

Directories

Directories are special files that would rarely require the indirect pointers of its inode. On most UNIX systems that use a 1K block, directories contain 16 byte entries. Two bytes contain the inode number of the file and 14 bytes contain the actual file name. Each block can thus hold 64 entries. As files are added to the directory the next available entry is used. Directory management is a sequential process. To add the 577th $[(9 * 64) + 1]$ file to a directory requires the first 9 blocks to be read and searched for an empty entry before a new block is allocated for the directory and placed in the tenth inode block field. If there are 641 files in the directory, the last file would require an indirect block access to get the directory entry. Therefore, just to find the inode of the last file would require 14 disk reads (10 direct blocks, 1 indirect block, 1 pointed to by indirect block, 1 for the directory entry block and 1 to read the inode). If the first 638 files are deleted, the entries for `.` and `..` are always defined, the directory will stay the same size. It will still take 14 disk reads to get the inode of the only file in the

directory. Again HFS rides to our rescue to save us from this primitive environment.

Series 800 HP-UX directories come in two forms. Directories may have a maximum file name size of 14 characters or a maximum of 255. But, because of the large blocks used by HFS and because there is structure added to the directory entries, the problems that other UNIX's have are mostly overcome by HP-UX. While large full directories still require the file name to be found via a sequential search of the directory blocks, the larger blocks reduce the number of disk reads required. The internal directory entry structure permits HP-UX to only look at used entries instead of used and empty ones. This corrects the problem of large sparse directories.

With directories you obviously want to keep the number of files in them small, one or two blocks. If you do have a directory that contains hundreds of transient files, be sure to remove the files when they are no longer required. If all of the files in the transient directory do not get deleted and the directory requires more than one block, move the ones that remain. Either by hand or with a script, perform a mv operation on each file that remains after the massive purge. Move it to a different directory or to a different name in the same directory and then back to the original name. This will place all of the files in the first block of the directory. Just because HFS permits you to have 255 character file names doesn't mean you have to create them that large. System performance and personal sanity will be better served with reasonably sized file names. I would use the large file names option (newfs -L) when creating file systems. Not for the ability to have long names but because each directory entry only uses the number of bytes it requires. The short file name entries (newfs -S) are all 32 bytes long while the long entries only use the space required for the name plus overhead. Generally the long file name file systems will place more directory entries per directory block than the short file name systems.

Tuning

The commands I will be discussing are newfs and tuneufs. Newfs is a front end command to mkfs and so the mkfs options will also be discussed.

File System Sections

When creating a file system, the first parameter to be determined is where on the disk you want to place it. Series 800 disks are divided into sections. There are up to 16 sections on a disk depending on it's type. The disk type also determines which sections overlap each other and the maximum number of non-overlapping sections. The file /etc/disktab describes the disk sectioning. An example of a disktab entry is the following for disks greater than 350M bytes and less than 2G bytes. The numbers are the section numbers. This is the number that is used in the file system path name as the last number (following the s). /dev/dsk/c1d0s7 identifies section 7 of the disk. The table shows that sections 15 and 14 represent the same area of the disk as sections 0 and 1. And section 9 overlaps sections 4 and 5. Section 2 overlaps all of the other sections. Since the file systems on a disk can not overlap, if you decided that one of the file systems was going to be section 13 the other file systems would have to be sections 6 and 15. You could pick 6 and 0 but then you would be wasting disk space.

outer cylinder

6	6		6	6		
0	15		15		7	
1	14					
10	10	10		12		2
3	3		13			
4		8			11	
5	9					

inner cylinder

I find the figure from the "How HP-UX Works: Concepts for the System Administrator" book page 8-12 easier to visualize the disk sectioning concept.

For disks of this type, you could specify the entire disk to be used by one file system by specifying s2 as the section portion of the path name. Or, you could have 7 file systems on the disk by specify s6, s0, s1 s10, s3, s4 and s5 as the file system sections.

The following table values were taken from the /etc/disktab file. As you can see, if the disk was an hp2203 and you made file systems using sections 13, 6 and 0, as mentioned earlier, you would use

$$603,872 + 1,998 + 24,280 = 630150 \text{ 1K}$$

blocks.

Specifying 13, 15 and 6 would use

$$603,872 + 1,998 + 48,560 = 654430 \text{ 1K}$$

blocks.

Therefore, using section 0 instead of 15 would waste over 23 megabytes. Specifying just section 2 would use the entire disk, 654,948K blocks.

section	hp2203 # 1K blocks	hp7935 # 1K blocks
-----	-----	-----
0	24,280	24,280
1	48,560	48,560
2	654,948	394,979
3	29,298	29,298
4	107,426	107,426
5	313,236	53,520
6	1,998	1,998
7	75,484	75,348
8	450,192	190,462
9	420,812	161,160
10	129,024	129,924
11	579,464	319,630
12	652,688	392,886
13	603,872	344,148
14	24,280	24,280
15	48,560	48,560

newfs

Now that we understand the disk sectioning we are ready to create our file system(s) on the disk. This section will explain how to do this using the `newfs` command. Only the optional arguments that when changed from their default values may affect system performance will be discussed.

`newfs` requires a disk section path name and a disk type. The section path name must be of the form `/dev/rdisk/cxxd0syy` where `xx` is the logical unit number of the disk and `yy` is the desired section on the disk. The disk type is defined in the `/etc/disktab` file. To determine the disk type of a given disk, enter the `diskinfo` command and look at the returned product id number. For example on the system I am using

```
diskinfo /dev/rdisk/c1d0s7
```

returns a product id of 2203. Therefore, my disk type is `hp2203`. The `newfs` command that uses default values for the optional arguments would be:

```
newfs /dev/rdisk/c1d0s7 hp2203
```

The optional arguments I will be discussing are `b`, `f`, `c` and `m`.

The Series 800 HFS permits us to define the block size to be used by the file system. In order to guarantee that no file will require triple indirect addressing within its inode, the minimum block size is defined to be 4096, 4K. The only other size we will consider is 8192, 8k. Blocks can be made 64K, but that is not a realistic size for the environments we are discussing. A block is read or written when a program performs file I/O. The block is kept in memory until the space is required for other activity or the file is no longer being used. A larger block, for some types of files, will result in better system performance because there will be fewer disk accesses required to read the file's data. There are some cases that a larger block size hinders performance. This will be discussed later.

Fragment

As I mentioned earlier, the HFS places the last part of a file in a fraction of a block. This fraction is called a fragment. The concept of block

fragments is included in the HFS in order to prevent the last block of every file on the system from wasting an average of 4K bytes. As a file grows, the data that makes up the last block of the file is placed in a fragment of a block. If the defined fragment size is 1K, and a file is between 8K and 9K bytes in size, the first 8K bytes will be placed in a block and the bytes left over will be placed in 1K bytes of a block being used for fragments. Once the file grows beyond 9K, the fragment will require another 1K. If within the fragment block the 1K following the current fragment is free, it will be allocated for the file. If the next 1K fragment is not available, a 2K fragment must be found in a different fragment block. The current 1K fragment is moved to the new 2K fragment and the new data added to the newly allocated area. This process is continued until the fragment fills an entire block. Fragment blocks are only used for the direct access blocks (the first 12). Once data blocks have to be accessed via indirect blocks, only whole blocks are allocated.

A test I ran on a new file system that had 1K fragments and 8K blocks pointed out that fragments are moved even when the file system is empty. I then tried a file system with 8K fragments and 8K blocks. And found a slight degradation in performance. My tests showed a performance improvement when the block size was 8K and the fragment size was 4K. It is surprising that there is a measurable difference since only the first 12 blocks worry about fragments.

Cylinder Group

The number of cylinders that make up a cylinder group is also a configurable parameter. The default value is 16 cylinders per group and the maximum value is 32. There is a small amount of overhead associated with each cylinder group to manage the blocks and inodes. If the number of cylinders per group is too small, this percent of overhead blocks to usable blocks becomes high and a significant portion of the disk is not available for user data. Also, more processing power may be required to manage many small groups instead of a few larger groups.

Free space

The -m option permits you to specify how much of the

file system will not be available for normal use. The default is 10 percent. This means when your file system becomes 90 percent full, no one except the super user can allocate any more space to any file. This parameter is intended to prevent system performance from degrading. It reduces the time required to find free space when a file is expanding. By having the minimum free space at 10 percent, or larger, the system is suppose to run better because there is a greater locality of reference when accessing files. The disk doesn't have to search as far from where it currently is to find the next block the OS has requested.

tunefs

Once a file system is created, the command `tunefs` can be used to modify some more file system parameters. The two I will discuss are `d` and `e` (the percentage of free space, `-m`, may also be set by `tunefs`).

Rotational Delay

The `-d` argument specifies the time in milliseconds it takes to complete a data transfer and initiate another one. Using this information, the system can determine the optimal positioning of a file's blocks on the disk. If the blocks are placed correctly, then when they are read the data seek times will be minimal. As the disk platter is spinning, all of the blocks that have been placed on a given track would be read in one revolution of the platter instead of requiring about one revolution per block. This is a very sensitive parameter and should not be touched by the faint of heart.

Blocks per group

`-e` defines the maximum number of blocks per cylinder group that any one file is permitted to allocate. The default value is about 25 percent of the total blocks in the cylinder group. This is to prevent one large file from using all of the blocks in the group and forcing the other files within the same directory to be placed in other cylinder groups. This parameter tries to ensure locality of reference for between files and their inodes.

Tuning ATG (According To Glen)

If the files on your system are all small bin and text files and you use a data base that manages its own partition HFS is built for you. The default parameter settings were set with you in mind. Especially if your system is seldom used by more than a couple people at a time doing highly interactive work.

If, however, you have a heavily used system that has several large files. And, those files are a type of ISAM (index sequential access method) file, such as C-ISAM or TISAM. Or, if you use your own indexing scheme built into a flat file, read on.

One of the great philosophers of my generation has said many times;

"You can't always get what you want
But if you try some times
You just might find
You get what you need."

HFS is not what I want . Let's see if it is all I need.

File System Sections

The HFS does force us to pick 1 of up to 16 locations on the disk for our file systems. This isn't what I want . I would like to have more freedom. I hate external constraints. But, it is definitely all I need (especially if the Logical Volume Manager is used).

My goal in life is to keep things as simple as possible. To that end, if resources are not a limiting factor, I would have one file system per disk. This simplifies performance tuning since I would now only have to worry about load balancing disks and not file systems on disks. One of the main reasons the HFS has cylinder groups is to reduce disk seek times by increasing the likelihood that successive disk access are physically near each other. By having multiple file systems on a disk, we are forcing the possible accesses to be far apart.

Unfortunately we, or at least I , don't live in a perfect

world where the accountants give us all of the money we want for equipment. Therefore, in this imperfect world, we do have to place multiple file systems on some of our disks. This is because some files perform best when the file system is tuned one way and others when it is tuned another way. We obviously want to separate these files into different file systems. Bin files are generally accessed best on a file system that uses the default setting while large data files require different settings.

Or, if during the day time one set of files are primarily accessed and in the evening, when reports are run, a different set are accessed, it would make sense to separate them into two file systems. If they were kept in one file system, the blocks of both sets would be spread across the entire disk. If each set was in their own file system, then their blocks would be closer to each other.

If you have to use multiple file systems, try to put an infrequently accessed one (or two) on the same disk as one that is frequently accessed.

Directories

ATG the fewer directories on a file system the better. Each directory is placed in the cylinder group that has the fewest existing directories. That is, assume you have a newly created file system that contains about 40 cylinder groups (hp2203 with a -c value of 32). The first directory created is placed in the first cylinder group, the second directory in the second cylinder group and so on. If you put a file in the first directory, its data will be in the first cylinder group near the beginning of the disk. If you then put a file in the twentieth directory, its data will be placed near the middle of the disk. Alternating access to these files will require the disk head to move half the distance of the disk each time a block must be read. If the files were in the same directory, the disk seek time is greatly reduced. Having many directories on a HFS forces the

data to be highly fragmented across the disk. You may be only using one percent to the disk, but you are requiring the system to span the entire disk in order to access the

data.

I like some things

For files that are opened and then the whole thing, or large chunks, are sucked into memory is one read, the HFS is what I need and want. The large blocks, rotational spacing, and inode placement algorithms are beautiful. Every millisecond of performance is accounted for while still being realistic. Or, at least, it was when the algorithm was created. But now, with many disk controllers performing their own track caching, the availability of bigger cheaper disks and the active user count on the machines sky rocketing, many of the design decisions should be rethought. I like the algorithm but I also liked my 1964 Rambler as a teenager. Time has past both by.

In today's computing environment with its large data bases that are randomly accessed, tuning is required.

Tuning Parameters

-b

Large is a relative term. If your system has 100 megabytes of data that is accessed most often (80% of the time) and you can use 25 megabytes of memory for buffer cache, that is not large. 100M of data in this case isn't large because there is so much buffer cache that you can expect a high buffer cache hit to miss ratio and thus logically reduce the amount of data. In almost all cases like this it is best to use a 8K block size. If, however, your system only has 2 megabytes of buffer cache, very few of the disk buffers can be cached and the buffer cache hit ratio becomes small. In cases like this, a smaller buffer results in better performance.

This is because, when the block is read into memory, usually only a small portion of the data in the block is used immediately. The rest of the data may be accessed later by the calling program or another program but that depends on system activity. If there isn't a lot of buffer cache space, using 8K instead of 4K cuts in half the number of different buffers that can be in memory (it

also takes twice as long to perform the actual disk to memory transfer but this isn't a significant portion of the total buffer access time). For randomly accessed data (such as an ISAM file) the likelihood is small that the rest of the data in the block will be required before the block is thrown out of memory to make room for a different block. With smaller blocks, it is more likely that more frequently accessed blocks will stay in memory since there is room for more blocks.

- f

The fragment parameter only affects performance when files are being increased in size. It is possible with a 1K fragment and an 8K block to have to move the original 1K of data seven times before it is placed into an 8K block. This file data movement causes system overhead which degrades performance. Fragmentation is only used with the first 12 blocks of a file. After that, an entire block is allocated every time. I had expected this parameter to have little affect on performance on the building of a large data file. I was surprised. With an 8K block I had best performance when using a 4K fragment. With a 4K block a fragment of 1K worked better than a 4K fragment. These results for a large data base are more interesting than important. However, for a file system that has many small transient files I believe an 8K block size and 4K fragment would work best. I am not worried about wasting up to 4K per file on today's systems that have gigabytes of disk space. Most systems today are limited by performance not by disk space.

- c

For large files, cylinder groups just get in the way. By setting -c to the maximum value, 32, and making the recommend modifications to -d and -e file accesses are a little faster. It is not significant and leaving it set to 16 probably won't make a noticeable difference on your system. Be sure to make the -d and -e changes. Just changing the -c to 32 actually caused my tests to run a little longer. The amount either way is small, but every little bit helps.

- d

-d 0 will cause the blocks of a file to be placed contiguously on the disk, if possible (there is one quirk in the HFS allocation scheme that makes the last

statement slightly incorrect but for the most part it is true). For ISAM or ISAM like files, the blocks are not read sequentially. To perform a single record read, block 10 then 31024 then 392 then 51324 may have to be read. Using rotational delay to try to time the placement of block negatively affects performance because it increases the scattering of the file's blocks. -d 0 by itself, helps performance to some extent but by adding the -e modification, large files are helped a lot.

- e

This option specifies the maximum number of blocks a file can use within a cylinder group. Remember the purpose of cylinder groups is to increase the locality of reference between a file's inode and the first several blocks of the file. For large files we want locality of reference to be as small as possible between the blocks of a file. I don't care how far from the inode the data is because it won't be accessed very often compared to how often the data is accessed. Therefore, set -e to the blocks per group value. If you don't want to look for that value (by doing a tuneufs -v) just enter -e 9999. That is large enough. -e by itself will not keep the block of a file close to each other. You must also specify -d 0. If you do not, the system will permit the file to use the entire cylinder group for one file but since the allocation scheme is skipping X number of blocks between allocations, when it gets to the end of one cylinder, instead of using free blocks earlier in the group, it will move on to the next cylinder group.

- m

For file systems that are used for large data files and tuned as I have suggested above, I would set -m to 1. Leave a little space for the super user to maneuver. The purpose of leaving 10 percent free space is to aid performance. For file systems that contain mainly small transient files, this may help while allocating files. But for file systems with a few large files, all this accomplishes is to waste 10 percent of your disk. The free space is not kept equally spread among the cylinder groups. depending on how the files in the files system are created and expanded, the free space may be all in one cylinder group or spread around in a few. The free space is not forced to be spread among the cylinder

groups. It is not the amount of free space in a file system that affects performance but the distribution of the files. If we are suppose to not use 10% of our disks, then HP. should charge us 10% less for the disks.

The -b, -d and -e options affect performance the most. The others, -m, -c (16 or 32) and -f for large file systems have minimal affect.

A Bold Statement

File systems that are heavily used by multiple processes and are nearly full, generally do NOT adversely affect individual process performance or overall system performance if the files are overly fragmented. This is because the next block to be read is likely from a file other than the current one. Since the files are spread across the entire file system, the average seek distance would be half of the disk. If the file were not fragmented, the average seek distance would still be half of the disk. But, if

- 1) the files are spread across the entire file system and the system is less than two thirds full or
- 2) the file system is lightly used by multiple processes or
- 3) the file system is used by a single process such as a report generator

performance is adversely affected by file and file system fragmentation. All of these situations cause the average seek time for each disk I/O to be longer than if the files on the file system were contiguous.

One way to correct this situation is to create a new file system and copy all of the files, one at a time, from the fragmented file system to the new one. A less attractive way is to backup the file of a system to tape, recreate the file system and then restore the files. This is a scary method because the files must be deleted from the disk and only be on tape. If you use this method be sure to have multiple good backups (color me *paranoid*). Since the design of the HFS makes it difficult to contiguously allocate a single file's data let alone multiple files, a better way is to use a utility to unfragment the file system in place. The utility can bypass the HFS allocation algorithm.

Wish List

I would like to be able to specify a cylinder group of 64 or (gasp) 128 cylinders instead of a maximum of 32. Or, even better, would be to say that the entire file system was one cylinder group. This would reduce the system overhead of jumping between cylinder groups as well as give me back some disk space (I know, picky, picky). I would not use this feature for all file systems, but it may help in some cases.

I would like to be able to turn the fragment block feature off. If I say -f 0 while doing a newfs, let that mean to always allocate full blocks to the file. Never add the overhead of moving fragments just to save a little disk space. I am still researching the strange results I saw in my testing. When I have a block size of 8K, a 4K fragments is optimal. I had thought an 8K would be best. An 8K fragment actually gave me slightly worse performance. When I have a block size of 4K, a 1K fragment is best.

I would like to be able to set the block size to 1K or 2K. For ISAM files that perform I/O in 1K blocks, having larger file system block sizes adds needless overhead.

I would like to be able to turn off the directory placement algorithm. Put all of the directories in the middle of the file system instead of spread all over the place. Then allocate blocks from the middle out in both directions.

OPEN SYSTEMS AIR TRAFFIC CONTROL

ROBERT L. CHEW

EG&G DYNATREND

**VOLPE NATIONAL TRANSPORTATION SYSTEMS CENTER
55 BROADWAY, CAMBRIDGE, MA 02142 PH: 617-494-3637**

THE ENHANCED TRAFFIC MANAGEMENT SYSTEM

Today, via networked computer workstations, air traffic controllers easily access data on traffic and flying conditions over, into, and out of the U.S. air space using the Enhanced Traffic Management System (ETMS). Developed by the Volpe National Transportation Systems Center for the U.S. Federal Aviation Administration (FAA), the ETMS compiles real-time flight and weather data and presents it in an easy-to-understand format, alerting controllers of areas where congestion is likely to occur. This real-time data is updated at least every five minutes over satellite and terrestrial links nationwide. Using the ETMS, controllers are expected to save billions of dollars over the next decade. Those savings will benefit commercial airlines, passengers, and the U.S. government, which employs the air traffic controllers.

Every day 30,000 commercial and military flights carrying 1.5 million people are safely led through U.S. airspace. During peak travel times, as many as 4000 aircraft may be airborne simultaneously, creating a logistical challenge for the air traffic controllers who must choreograph their takeoffs and landings. Considering the cost of operating large aircraft – as much as \$12,000 per hour – a one-hour delay of 100 aircraft, not atypical at a major airport such as Chicago's O'Hare, could cost more than one million dollars. The goal of ETMS is to minimize such delays and significantly cut air transportation costs.

Since the ETMS was first deployed in 1987, air travelers have experienced fewer and shorter delays over airports. The system was installed initially at Los Angeles International Airport to handle traffic flow while a runway was repaired. The FAA's objective was to compensate for the temporary loss of one runway and allow for more efficient use of those still in operation. The system exceeded expectations and actually increased capacity, because it allowed controllers to evaluate the situation in advance.

Viewing the Aircraft Situation Display, air traffic controllers can sort information selectively, checking the incoming flights to a particular airport. For more detailed data, they also can zoom in on selected areas and see boundaries, routes, and navigational aids. In addition, the system allows controllers to focus on a particular flight, calling up information on its origin and destination, aircraft type, altitude, flight number, estimated time of arrival, and speed.

To help controllers foresee – and possibly prevent – problems, the system predicts the flow of air traffic up to eight hours in advance. ETMS also suggests possible solutions for resolving congestion problems and avoiding delays, allowing controllers to consider several options. And it provides weather information – data on precipitation, recent lightning strikes, and the velocity and location of the jet stream.

DISTRIBUTED COMPUTING EFFICIENCY AND RELIABILITY

The ETMS employs distributed computing technology, which allocates processing tasks to different computers on a network, and makes significant dollar savings possible. It allows processing that would have been done on expensive mainframe computers to be distributed over much less costly workstations. Additionally, the distributed architecture permits redundancy, so that a processing failure by any one of the workstations in the network causes no interruption of the information flow. Automatic detection and replacement of failed processors is built into the ETMS, to ensure a highly reliable system.

The ETMS was begun as a research program in the mid 1980s to test the concept of viewing live data representing all nationwide flights. The ETMS application has grown to approximately 400 thousand lines of Pascal code and comments and operates on hundreds of HP Apollo networked workstations using the proprietary Domain Aegis system.



TRANSITION TO AN OPEN SYSTEMS ENVIRONMENT

Initial planning for the open systems transition began in 1991, at which time the Volpe Center became a member of the Open Software Foundation (OSF). We had active participation as member of the End User Steering Committee, the Standards Coordinating Committee, and in the Special Interest Groups such as Distributed Computing Environment (DCE). OSF architectural standards and technologies, and the association with other end users and vendors through OSF have all been useful in helping guide our planning process for an open ETMS.

The ETMS graphical user interface (Aircraft Situation Display) has been completely re-designed with object-oriented concepts in C and C++, using X windows and OSF's Motif. We converted and demonstrated part of the ETMS using Motif and DCE for the OSF Challenge '93 event on the following initial platforms: HP700 UX, IBM 6000 AIX, DEC Alpha OSF/1.

During Spring 1994, the FAA authorized the Volpe Center to begin the conversion over a two year period, with the objectives that the ETMS be converted as an open system to continue operating on the current Apollo platforms, and also that the converted ETMS be tested on a variety of vendor open platforms in preparation for a major new procurement.

Thus, we are now in the midst of a major transition of the ETMS from the proprietary Domain Pascal language and platform software to an open, standards-based software architecture. This transition will conform to the U.S. government procurement guidelines, which specify open system technology standards. This will lower system

life-cycle costs through procurement of advanced technology products conforming to standards on a very competitive basis. It also will provide for interoperability among heterogeneous vendor products, portability of the application over different platforms, scalability over a wide range of hardware configurations, faster application development and reusable code, and a more consistent user interface.

OPEN SYSTEMS ARCHITECTURAL FRAMEWORK

The framework consists of a set of standards and specifications for integrating system and application software to meet the FAA requirements for ETMS. The target ETMS will adopt the standards and transition paths outlined below and permit procurement of any vendor platform and software tools whose products conform to those standards. The open systems standards selection is being guided by careful, pragmatic choices of de jure and de facto industry standards. The initial applicable standards areas (showing current ETMS implementation, target standards, and transition path) are:

OPERATING SYSTEM INTERFACES

CURRENT ETMS

- HP Domain/Aegis proprietary system calls

TARGET STANDARDS

- FIPS 151.2 POSIX 1003.1 System Calls, 1003.2 Shell and Utilities, 1003.4 draft Real-time and Threads extensions

TRANSITION

- HP Domain version SR10.4 POSIX compliant transition path

GRAPHICS AND USER INTERFACE

CURRENT ETMS

- Based on HP/GPR, and only available with HP Domain

TARGET STANDARDS

- FIPS 158 (X Windows) and MOTIF replaces GPR functions

TRANSITION

- Domain SR10.4 transition path, with X Windows and MOTIF

COMMUNICATIONS AND DISTRIBUTED COMPUTING

CURRENT ETMS

- HP proprietary Apollo Token Ring (ATR) network, and specialized interprocess communication calls provided by HP Domain with remote site communications using special built software

TARGET STANDARDS

- TCP/IP, X.25, NFS, and other full distributed computing functions in the OSF/DCE as standards evolve (including file transfer, time, security, directory, systems management)

TRANSITION

- HP Domain ATR and IPC to TCP/IP, X.25, and NFS.

APPLICATION DEVELOPMENT TOOLS AND PROGRAMMING LANGUAGE

CURRENT ETMS

- Approximately 400K lines of code with comments in HP Domain Pascal

TARGET STANDARDS

- ANSI C (FIPS 160) and selected use of C++

TRANSITION

- HP Domain Pascal to ANSI C

TRANSITION STRATEGY

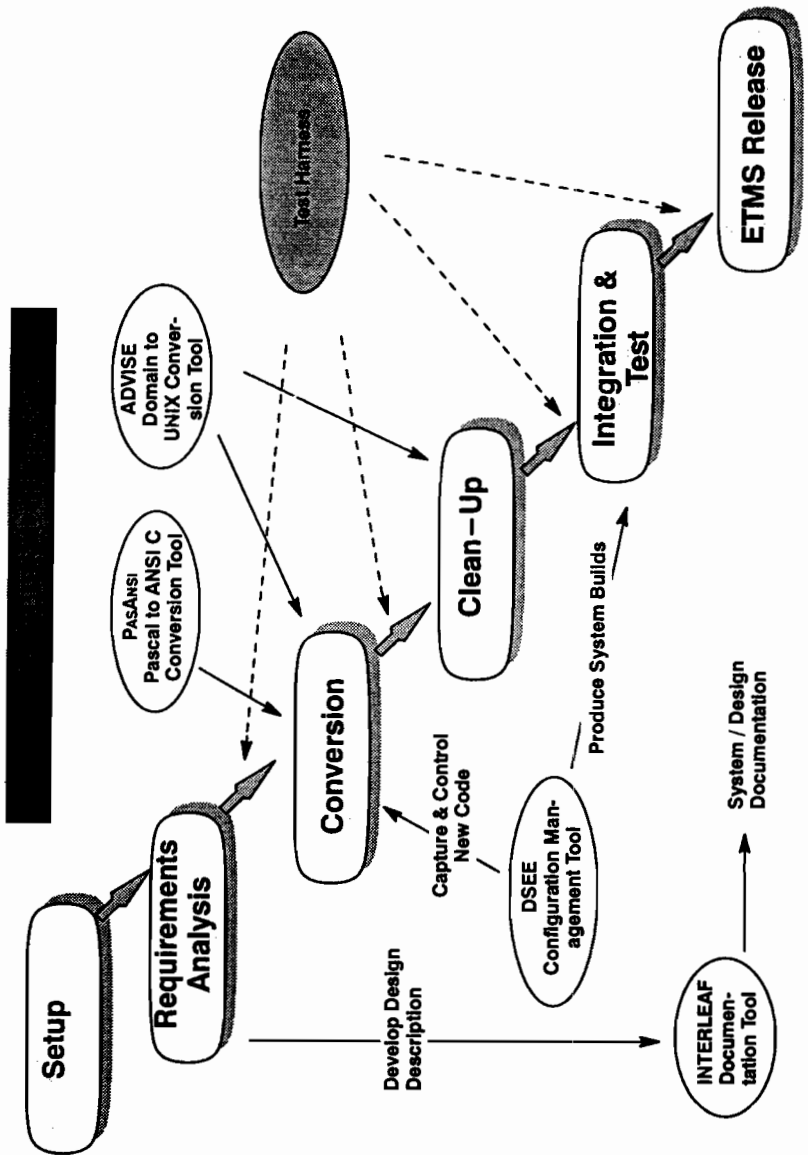
The strategy or approach is phased, evolutionary, with modular development, integrated with ETMS functional changes, and fielded over the course of the conversion process. Conversion will comply with open systems standards as identified in the previous framework section.

This phased, evolutionary approach is justified for sound engineering and management reasons. Some of the important arguments for the approach are:

- The phased approach avoids the major problem of integrating an evolving ETMS with a separate complete conversion of ETMS.
- Performance degradation can best be avoided by effective engineering, appropriate hardware sizing, and adequate testing of phased module conversions in an operational setting.
- Functional integrity can best be assured through building and running the software in an operational setting after each discrete set of changes.
- Impact to users may be minimized by introducing changes a little at a time. The operational functionality in the transition process will be transparent to the users except for the change in the user interface.
- Impact to the support personnel can be minimized by introducing changes a little at a time.

The conversion work is being done on the current HP/Apollo Domain platforms. Converted modules are tested on a variety of different vendor open system platforms in the Volpe Center's open systems lab, to evaluate standards compliance, portability, and scalability of the modules tested across the platforms. Performance analyses (user, module, communications, data access) are being conducted during the conversion process to assure that the completed system will meet the FAA's requirements.

The Open Systems Conversion Process and Software Tools Used



TECHNICAL ISSUES

Three key technical issues are discussed here: Using an API to become platform independent; Testing the converted ETMS; Parallel software development.

USING AN API TO BECOME PLATFORM INDEPENDENT

The goal of the conversion is to port the ETMS from a proprietary Pascal/Domain/GPR environment to a generic C/Unix/X/Motif environment. The language conversion (Pascal to C) is a much different job than the "system-calls" conversion (Domain to Unix, GPR to X/Motif). The conversion is being performed in a manner that makes the two jobs independent. Furthermore, the "system-calls" conversion can be done in a manner that allows the application to be built for either Domain or Unix at any time, and that facilitates porting the application between different implementations of Unix/X/Motif. The conversion of the system calls is done by developing an Application Program Interface (API). An API is a set of subroutines which correspond (generally one-to-one) to the system calls. The application programs (i.e., ETMS functional modules) call an API routine rather than a Domain routine or a Unix routine (other than calls that already exist in ANSI C).

TESTING THE CONVERTED ETMS

The ETMS presents an enormous problem for testing. It is a very large system with many complex processes. The reason it works as well as it does today is that it has been developed incrementally and used operationally over a number of years. While in-house tests can detect gross errors in behavior, there are thousands of subtleties throughout the ETMS processing which have been established and tested through years of operational evaluation and use.

The question then arises: how can a new version of this processing be created with minimal risk of loss of functionality, and how can the accuracy of the converted processing be established. One school of thought is to take this processing along with all other aspects of the ETMS processing, and re-engineer it into a totally new package. It is difficult to see how this new package can be brought to the same level of accuracy and functionality without going back and repeating years of the operational use, bug-fixing, and engineering improvement that ETMS has already gone through. The other school of thought is to preserve those years of effort by mimicing the current processing as closely as possible in the converted product, and by introducing the changes into the system in small pieces. The latter approach has been chosen as the least risky.

PARALLEL SOFTWARE DEVELOPMENT

Parallel development is a problem which many software engineers working in other environments do not encounter in a significant manner. However, it is one that we have had much experience with on ETMS. Tools exist, such as SCCS, DSEE, and Clear Case, which help handle the mechanics of parallel development. But the main issue is how parallel development is managed. The problem can be simply expressed: the longer parallel development goes on without merging the changes, the more difficult the merging. Furthermore, the difficulty increases exponentially over time.

There are two approaches to resolving this problem. One is to eliminate parallel development by only allowing one person to work on a module at a time. This solution is not practical in ETMS. The other solution is for people to work on copies of the module, periodically stop and merge their changes into a single version, and resume their development on copies of the newly merged version. By carefully controlling the amount of change that can be introduced between merges, one can ensure that the merges are always straight forward.

Our approach is to carefully manage the changes. Furthermore, by not re-engineering the system during conversion, the ease of merging changes can be assured even during the conversion of the programming language.

SUMMARY

The ETMS is undergoing a major transition to open systems standards compliancy to meet FAA operational requirements for future air traffic flow control. Based upon the planning and analysis done by the Volpe Center, conversion of the ETMS is well underway, nearing completion of the initial pilot project. The pilot represents a major learning experience of selecting an ETMS module, getting outside expertise to work with internal staff, using software tools to convert the code, converting Domain system calls to a POSIX compliant API, and training staff to maintain the converted open system.

At the end of this two year effort, we will have completed conversion and evolutionary integration of the standards-compliant ETMS into the FAA's operational ETMS, during which functional enhancements will continue to be made. The FAA will then be in the position to procure new vendor open systems platforms and also re-engineer parts of the ETMS as needed.

Acknowledgements must be given to all the technical staff at the Volpe Center who have created, implemented, and maintain this very successful system, with the close cooperation of the FAA controller organization nationwide.

Paper Number 6009
ODBC: Visual Basic Application Development
with HP/ALLBASE

Jeff Zalkind
Database Specialist
Microsoft
Building 6 One Microsoft Way
Redmond, WA 98052
206.936.2942

Handouts will be provided at time of presentation

Paper Number: 6010

Title: Client-Server and Rightsizing, Open Systems and Challenges

Author: Marlene Nesson

Company: Information Builders, Inc.

1250 Broadway

New York, New York, 10001

(212) 736-4433

General Overview

Client/server and rightsizing, along with open systems are constantly touted solutions as well as buzzwords in the industry today. But really what does this all mean? Everybody seems to think it is a good thing- even if they can't agree on a definition.

Client/server seems to be a new kind of parlor game for the industry. It has been described as a style of computing, a way of saving money, a collection of technologies, an architectural platform, an application development method, a systems integration solution, a re-engineering tool, a way to implement standards and even a paradigm shift. Client/server has many, many different flavors. It won't look the same everywhere you go as there is no textbook that defines client/server and even if there was a textbook definition the English language is open to different interpretations.

Rightsizing, originally called Downsizing (since the term inferred the moving of applications from a larger system down to a smaller system), is basically looked at moving applications and/or data from one system to another. It may mean changing the application or keeping the application the same. It may mean utilizing existent technologies or implementing existing technologies. Once again, since there is no textbook definition rightsizing will look different wherever you go.

Open systems has been referred as interoperability across heterogeneous systems, networks, management of heterogeneous database, integration of open and proprietary networks, and integration of standards. Open systems offers users both platform and vendor independence. Users have the flexibility to choose the hardware, the software, the network and operating environment according to what makes the best business sense.

A simple definition for all 3 of these technologies may be freedom of choice. Freedom to choose the hardware platform, databases, tools and user environments that are right for your business.

This paper will approach the subject of client/server, rightsizing and open systems through real business challenges and the solutions implemented that were the right solutions for each individual's business.

We'll begin by presenting an overview of client/server, rightsizing, and open

systems followed by factors companies considered in adopting and/or integrating these new technologies. Two in-depth studies of companies who have implemented client/server, rightsizing and/or open systems solutions will then be discussed.

Overview of Concepts

Client/Server

Client/server is an open architecture that distributes processing between a back-end processor called a "server" and a front-end processor called a "client".

The distribution of the workload between two or more processors would result in greater efficiency and lower network traffic.

The key components in a client/server environment are an

- Operating system(s)
- Database management system
- Development tools
- Graphical user interface and/or a character interface
- Client/server applications
- Connectivity protocols
- LAN (local area network)/WAN (wide area network).

Client/server computing grew from the network computing of the 1980s. Networking gave users the ability to share files and resources. Client/server computing, however, takes the networking technology a step further. It allows applications processing to be shared and allows the ability to operate on multiple platforms, systems and networks.

Client/server computing is experiencing overall growth. In fact the Meta Group, believes that "...by 1995, virtually all new applications will be client/server applications". The most dynamic area of expansion in client/server computing is software development: database management systems, development tools, and client/server applications. For example, Sybase and Informix, a database management system software vendor, more than doubled unit shipments between 1990 and 1992.

Client/server computing produces a number of benefits and advantages. *The most common benefit is cost savings realized by capping mainframe investment, increasing productivity, reduced training time, and buying less expensive workstations, desktop platforms and software.*

Rightsizing

Rightsizing encompasses the act of changing platforms, moving and/or changing applications and utilizing existent and/or implementing new technologies.

Changing platforms include: downsizing - moving systems to smaller, less expensive platforms; as well as upsizing - increasing processing power when the current systems reaches its capacity. Rightsizing also involves a wide spectrum of activities, the three most common being:

<i>Rehosting:</i>	Moving applications, unchanged, to a new platform
<i>Rearchitecting:</i>	Adding new technologies such as client/server graphical user interfaces or relational databases, to the existing applications
<i>Reengineering:</i>	Making significant changes to the underlying business processes

Originally, rightsizing, was called "downsizing", or more specifically, "system downsizing", and referred to the processing of moving information systems off the proprietary host, usually a mainframe or minicomputer, to a lower-cost (hence, smaller) system. The system might be a PC or UNIX workstation or server, which was clearly smaller than the host, or even a multiprocessing UNIX machine, that, quite possibly, rivaled or exceeded the proprietary mainframe in size and processing power. The smallness implied in downsizing really referred to the cost of the new system, which came with a smaller purchase price and a lower cost of ownership over time, regardless of its actual size.

Rightsizing, as a word or term, is better suited to describe the activity than downsizing as it refers to the process of moving information systems to the most appropriate system for a particular application whether it be larger or smaller. Rightsizing matches the needs of the application and the organization with the strengths of the various systems and often client/server is integrated in a rightsizing scenario. Thus, the "right" in rightsizing refers to picking the best system for the application based on the system's strengths, not its size.

Rightsizing scenarios typically move toward the same goal: distributed, networked systems that interoperate transparently as one system - freedom of choice.

Open Systems

An open systems environment supports interoperability across heterogeneous systems and networks, management of heterogeneous database and the integration of open and proprietary networks and the ability to run with industry

standards. Open systems offers users both platform and vendor independence. Users have the flexibility to choose the hardware, the software, the network and operating environment according to what makes the best business sense. Therefore, a major advantage that open systems have over closed architecture host-based systems is multiplatform and multivendor support.

The problem experienced by end-users with the closed nature of proprietary, host-based systems was the inflexibility of the closed environment. It was difficult to share information. Essentially locked into a platform or vendor, there was little or no flexibility to address users information needs as they grew or changed. End-users had to use the same hardware and software vendors even if that platform or software was not the best option to solve their business needs. Information might be available solely on one platform, yet difficult to access because the users of the information, scattered throughout the organization and network, might be on different platforms and software systems. Open systems solution offers the users the ability to get the information they need, no matter where it resides, and deploy applications on the hardware that makes the best sense - freedom of choice!

X/Open Definition of Open Systems :

"The three goals of an open systems environment are interoperability, portability and scalability of applications. Interoperability is the ability to share information in a heterogeneous environment of dissimilar workstations, servers, networks, and vendor applications. Portability allows for the migration or "porting" of the applications to other platforms. Scalability allows movement of an application from a PC or workstation to proprietary host mainframes and minicomputers."

X/Open is a consortium dedicated to developing an open systems standard, and is widely accepted in the industry. Another group, with similar goals for software application development, is the Open Software Foundation (OSF) and a newly formed group called Common Open Software Environment (COSE).

Note: COSE was announced at Uniform 1993 by Hewlett-Packard Company, IBM Corporation, The Santa Cruz Operation Incorporated, Sunsoft Incorporated, Univel, and UNIX System Laboratories Incorporated. Each company said it would adopt the most popular de facto standards in the UNIX business: the Motif user interface, Sun's Open Network Computing (ONC) networking, and the X Consortium's X Window system. Although not positioned as "another" standards committee or organization, the six companies stated they would announce a set of specifications that would result in a common set of programming and user

interfaces that should bridge many of the gaps between their different versions of UNIX. COSE is thought to be the best news for standardizing UNIX.

It should be noted that in the marketplace, open systems have been more or less synonymous with UNIX. Although as clearly described above, open systems embodies much more than UNIX and more appropriately an interoperable environment that can include UNIX as well as a host of heterogeneous systems, networks and databases - once again the freedom of choice.

Many believe that open systems provide the ability for applications to work with or be integrated with industry standards, whether they be protocols, application programming interfaces (APIs), structured query language (SQL), object technology, and so on. There is a standards committee or organization for almost every flavor of data processing technology in today's marketplace. The goal here is not to review all the "industry" standards, since standards, as such, are still emerging. However, it is worth pointing out that open systems does seem to inherently imply the ability for applications to be incorporated into and/or interoperate with standards as they evolve in the computing industry.

Given this brief overview of client/server, rightsizing and open systems let's look at some of the factors companies are considering in the adoption of these new technologies.

Factors - Adoption of New Technologies

Today's workplace whether in retail, manufacturing or service is virtually unrecognizable from the one of twenty years ago. Change in the workplace, however, is now moving at an exponential rather than a linear rate. As a result, the workplace five years from now will be so different from that of today that it will likely have a brand new name. For example, the "home office" was, until recently, a concept that engendered ridicule. Today, for many professionals, the home office is the preferred situation.

As companies strive to remain viable in an increasingly competitive market place, they are examining ways to become meaner and leaner. The main thrust is to produce more with less. For a company's Information Systems (IS) people, this means tightening up the technological processes of data management.

To do so, MIS is bringing the large mainframe applications down a few pegs to a more manageable and cost-effective size. It also means simplifying the complexities of data analysis which historically had been handled solely by

technically oriented IS experts and moving these procedures down to the workers' level where the need for stored information is most crucial.

Client/Server, Rightsizing and Open Systems- Related and Inter-related Technologies

So what are companies doing to cut costs as well as enable the workers to get to this critical information? Companies are rightsizing moving from mainframes down to less expensive, user friendly personal computers or workstations. However, the mainframe is not being eliminated but rather is being used in a more innovative and ambitious way in conjunction with other less costly computer equipment. One method of using the information and technology already in existence is client/server computing. Companies who have chosen to rightsize see client/server computing as one way to use their systems to stay competitive and profitable. Open systems, that is interoperability across heterogeneous systems and networks, management of heterogeneous database and the integration of open and proprietary networks, is a requirement to implement a rightsized, client/server environment.

The companies that are profiled in the next several pages, adopted a rightsizing and client/server environment and evaluated a number of factors during their decision making process (see Figure 1). In almost all cases, these customers integrated a rightsizing initiative with a client/server computing environment. This meant keeping a mainframe computer, moving certain applications and/or data from the host, and setting up a client/server environment to take advantage of lower-cost client environments. The common reoccurring goals of all these customers were to *decrease costs, maintain investment in both hardware and software as well as improve workers ability to get to information quickly and efficiently.*

Factors Companies Consider in Adoption Rightsizing and/or Client/Server Technologies

- Improved individual productivity
- Maintain investment
- Less expensive hardware
- Minimize database management
- Shorten development time
- Faster turnaround
- Reduce personnel costs
- Faster, better market response
- Create more competitive products
- Pushing decisions to lower levels, employee empowerment

Figure 1

User Profiles

The profiles that follow are in-depth studies of 2 companies who have implemented rightsizing, client/server solutions. The following outline will be utilized for each profile described:

- Goal and objectives
- History
- Implementation: Hardware, software, people resource, and costs
- Challenges Uncovered
- Results and Comments

It should be noted that these company profiles are based on experiences by companies who use Information Builders, Inc. software products. Clearly, companies who have invested in other software products that are portable, flexible and comprehensive in functionality could be substituted in the examples that follow.

Profile I - Municipal Court

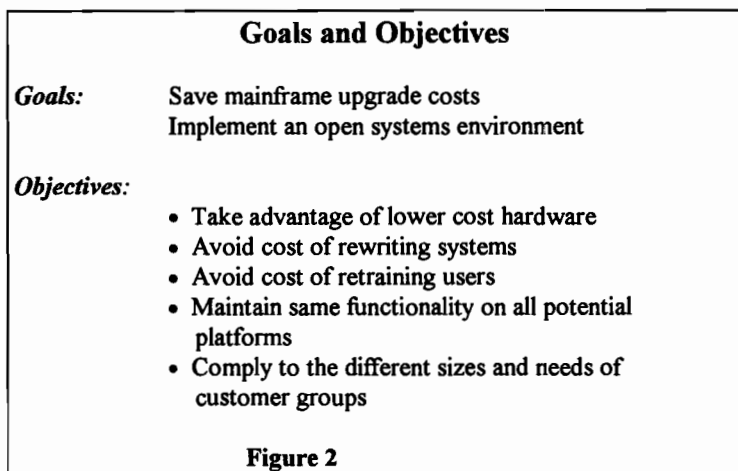
A municipal court from within California made a decision to rightsize their environment.

Their goals were to:

- 1) Save mainframe upgrade costs and

2) Implement an open systems environment

One of the primary objectives in this rightsizing effort was to maintain investment in the 4GL applications they had been running on their mainframe computer since 1988 so as to avoid the cost of rewriting systems and the retraining of users. Another objective was to maintain the same functionality on current as well as on all potential platforms they would integrate into their enterprise in the future. Still another objective was to implement a solution that would maintain the same functionality on the open system platform, UNIX, as well as other potential platforms that could be implemented into their enterprise in the future (see Figure 2).



History

The court was running 2 systems. Both of these systems were written in a 4GL called FOCUS from Information Builders, Inc. and were running on an IBM mainframe 3090 computer.

The users on the systems were approaching 300 and the mainframe capacity was being stretched to its limits. In order to maintain the users capability of real-time access, the court was required to upgrade their mainframe - a very costly endeavor. At the same, the court was beginning to investigate the feasibility of implementing an open systems and client/server environment. Since the decision to upgrade the mainframe would be politically a very unpopular decision (high level management said Open Systems and/or client/server was the strategic direction -

save lots of money), the MIS manager decided that an Open Systems environment, in this case UNIX was equated with Open Systems, would be implemented.

The court wanted to maintain flexibility as well as the investment in both software products and product training. Since the FOCUS code was portable across platforms (in this case from the mainframe to UNIX environment), the court was confident they would have the continued flexibility in choosing the platform of choice for deployment of applications as well as maintain investment in software knowledge and training.

The FOCUS product works with its own database (which was the database the court used) and was architected to work with over 50 databases, both relational and non-relational, across 35 platforms. The primary reasons the court chose to use FOCUS several years ago was: FOCUS code portability across all major platforms, comprehensive decision support and maintenance capabilities, and end-user tools (users can be productive without the requirement to know the FOCUS language):

Reasons for Choosing FOCUS

Code Portability, Decision Support, Maintenance

Portable code across 35 platforms
Access to over 50 heterogeneous data structures
Report Writer
Financial Reporting
Output Browser
Graphics and Statistics
User-Friendly Tools for reporting and maintenance
Universal Join (join heterogeneous data structures)

In addition, IBI provides a client/server (middleware) product that is supported across 35 platforms, which would enable the customer to implement a client/server environment as desired.

Implementation

The court set a timeframe of 6 months for the rightsizing project to be completed. This entailed 1) choosing and installing a UNIX machine 2) porting and testing the FOCUS applications and data on the UNIX machine and 3) deploying the new system. A future, secondary, goal would be the implementation of a two tier or

three tier client/server environment: PCs --->UX and/or PCs ---> UX ----> IBM mainframe computer.

Current Configuration

The court was running FOCUS applications with the FOCUS database on an IBM 3090 model 400J running the VM/CMS operating system. A Local Area Network (LAN) running Novell Network 3.0, had personal computers, IBM 486, running PC/FOCUS applications on the DOS and OS2 operating systems.

Configuration	
<i>Original</i>	<i>New</i>
<u>Hardware</u>	<u>Hardware</u>
IBM 3090 Model 400J	HP 9000 Model G50
IBM 327X terminals	IBM 327X terminals
PCs on LAN	PCs on LAN
	IBM 3090 Model 400J
<u>Software</u>	<u>Software</u>
FOCUS with FOCUS database	FOCUS with FOCUS db
	Terminal Emulator for IBM terminals (Haltek)

The FOCUS code on the IBM mainframe consisted of approximately 240 procedures with upwards of 4000 lines of code. There were 46 databases that consisted of approximately 3 gigabytes of data. The FOCUS procedures included online maintenance, reports as well as batch reports that were overnight.

New Configuration

After analysis of several UNIX computers, the court decided to purchase an HP-UX, G50 as their UNIX, Open Systems platform. The PC/FOCUS DOS and OS/2 applications were to be maintained as is; no changes required. The IBM terminals would be continued to be utilized and hooked into the HP-UX computer.

In order to meet the timeframes for implementation, the court decided to solicit the assistance of the IBI consulting organization. By incorporating consulting services

into their rightsizing project, the court was able to free up their own internal resource for other projects, eliminate the pressure and requirement to train the developers on the UNIX operating system immediately, and meet the goal of a 6th month implementation schedule.

Note: The court had a few very experienced FOCUS programmers. One of the programmers were dedicated to managing the rightsizing project. Utilizing outside resources enabled the court to minimize their internal resource on the rightsizing project and continue to provide timely support and service to their users.

The time from the inception to rightsize through competition was about 6 months. The major elements included:

- 1) Evaluate, choose and install UNIX (approximately 2 months)
- 2) Port software applications (approximately 3 months)
- 3) Deploy applications (approximately 1 month)

Challenges Uncovered

The porting of the FOCUS code from the mainframe to the HP-UX environment was relatively seamless as the code is portable from platform to platform. There was an expected learning curve involved with acquiring the required knowledge of the (new) UNIX operating system. This customer choose to train the current IBM knowledgeable system staff on UNIX rather than hiring an experienced UNIX systems person.

One challenge uncovered and that was maintaining the IBM terminals as the method to link into the UNIX computer. The court had the challenge of seeking out software that would enable these IBM 327X terminals the ability to emulate the UNIX environment (they did not want to purchase UNIX terminals or PCs for the users). The court chose a terminal emulator software solution called Haltek that was able to accommodate this requirement.

Note: There are several companies who provide similar UNIX terminal emulator for IBM 327X terminals. Included for your reference are a few suppliers of terminal emulator software:

Brixton Systems, Inc.	(617) 661-6262
Haltek	(410) 247-0886
NCR	(612) 638-7872
OpenConnect Systems	(214)888-0435
Systems Strategies Inc.	(212) 279-8400

Some of these solutions are software only solutions whereas others are a combination of software and hardware solutions.

Results and Comments

One particularly important element that proved to be instrumental to their success was the dedication of an internal resource as project leader. This project leader managed the scope, activities, progress and expectations of the project.

The court may decide to implement a client/server environment in the future although no timeframe has yet been determined. With the IBI products, the court is confident they can evolve into a client/server environment and maintain their software and people training investments into the future.

Today, there are approximately 300 users accessing the applications running on the HP-UX computer. They were able to meet all their goals and objectives and the court was pleased in their decision to rightsize. A realistic expectation of deliverables (6 month time period), internal expertise, and external consulting services enabled this rightsizing project to be completed successfully.

Profile 2 - Financial Service

This company provides financial services for more than 2 million investors with 66 billion dollars in assets. They supply a broad set of low-cost investment services that enable self-directed investors to customize investment programs to meet their financial goals. This company pioneered fast confirmation of market orders and provided an order entry service; therefore, delivering up-to-date, timely information to users were of primary importance to the companies continued success.

This company invested considerable capital in building its information technology infrastructure. The infrastructure not only includes computer hardware, software, and data network, but also a sizable professional systems staff. The applications running on that infrastructure touch virtually all phases of this financial systems operations. This company is heavily dependent upon information technology.

Their current information technology infrastructure was almost exclusively based upon large-scale mainframe computers. Mainframes were very expensive per unit of throughput. They recognized for some large workloads, mainframe are the only possible solution. However there were many workloads running on the mainframe that they believed could be processed on smaller and less expensive machines.

Use of these computer resources constitutes expense to the firm. It was not "funny money" as beginning in the 4th quarter of 1991, cost centers had been charged for their usage of various system resources. Because of Chargeback, users had a keen interest in any project that can potentially improve the cost effectiveness of their information processing.

This financial institution decided to implement a client/server, distributed system environment. The primary goals were to:

- 1) Reduce Costs
- 2) Rightsize and implement client/server architecture

The primary objectives in this rightsizing-client/server effort was to enable end-users continued productivity with the tools they were accustomed to using, provide users access to data regardless of where the data resides in the enterprise, rightsize current applications to a lower-cost, open systems platform, implement a bi-directional client/server architecture (UNIX <---> IBM Mainframe) (see Figure 3).

History

This financial company had been running with FOCUS applications for a number of years on an IBM computer. The number of users that were trained and using the applications were approaching 500. FOCUS was primarily used by end-users to create ad-hoc requests as well as individual applications to access, summarize and manipulate the data. Since FOCUS is an easy tool to learn and use, additional applications were being developed and the mainframe computer began to approach its upward capacity limit. In order for users to maintain fast response to their

Goals and Objectives

Goals: Reduce costs
Rightsize and Implement client/server architecture

- Objectives:**
- Take advantage of lower cost hardware
 - Avoid cost of rewriting systems
 - Avoid cost of retraining users
 - Provide access to current information (regardless of where the data resides in the enterprise)
 - Implement bi-directional client/server architecture

Figure 3

application requirements an upgrade of the mainframe computer was required. Since the upgrade cost to the mainframe was an expensive proposition and they continued to experience a steady increase in chargebacks, the company decided to rightsize to a lower-cost, open systems, environment. This lower cost solution would eliminate the requirement to upgrade the mainframe as well as reduce the cost centers "chargeback" for mainframe usage.

An open systems UNIX environment was decided upon. A majority of the mainframe applications would be ported to the UNIX, however, operational data as well some applications would continue to reside on the mainframe computer. Given this scenario, users would also require continued access to the mainframe. Therefore, a client/server environment between the UNIX and the mainframe computer was essential. This company added one additional twist to their scenario, they wanted users who required major access to the mainframe to use the mainframe as a client, but also have access to data residing on the UNIX machine. Or in other words, they were looking to implement a bi-directional client/server environment between the UNIX and the IBM computer (HP-UX <---> IBM).

Since over 500 users were productive with the FOCUS software, it was clear that a critical element of success for this project was portability of FOCUS code across platforms, that is, all the targeted rightsized applications could be ported to the UNIX computer. Also, since Information Builders offered a client/server, middleware product called Enterprise Data Access (EDA)/SQL that provides bi-directional client/server architecture, the company would be able to maintain their

investment in FOCUS and be able to implement the desired client/server environment as well.

One additional element that was important to this company's future planning, was adhering to, or integrating into, the Open Software Foundation's (OSF), Distributed Computing Environment (DCE) standards.

Implementation

The company set aside approximately 18 months for the rightsizing client/server project. This 18 months did not include the time spent defining the requirements (this took about 3 months) but rather the time spent on actually from beginning of implementation to competition.

They spent considerable time defining and refining requirements. Once they decided upon a UNIX platform - they choose HP-UX, they required the vendors, in this case, Information Builders, and Hewlett-Packard, to understand and assimilate their company's strategy. As mentioned above, DCE was a strategic direction for the company. Although DCE integration was not part of their initial plans, they were confident that their partners, Hewlett-Packard (ultimately chose HP-UX hardware) and Information Builders - (both members of OSF) would help the company achieve DCE compliance in the future.

They designed an elaborate project management plan for implementing a successful rightsized client/server architecture which included specific line items, or deliverables with specific dates and required timeframe for every milestone.

They assembled a project team consisting of HP, IBI, and specific individuals from the company. The company participants included: end-users (from Decision Support Group), UNIX system administrator (newly created position) and telecommunications (responsible for designing and implementing network solutions). This project team met weekly for the first 15 weeks, followed, then with bi-weekly meetings thereafter.

Two components of the project were considered to be new key technologies for the company: 1) UNIX and 2) TCP/IP. These "unknowns" were going to be watched carefully to ensure the company was comfortable with both the functionality and reliability of these new technologies.

In order to decide upon feasibility of the entire project, the first milestone was a port of the simplest FOCUS application and database to the HP-UX computer. The success or lack-there-of would indicate whether or not the project would proceed.

There were a number of milestones identified for this project (see Figure 4). The first milestone took approximately 3 months. This included choosing and installing a UNIX system, installing FOCUS and EDA/SQL on the UNIX machine, as well as porting the first FOCUS application with associated data. The first milestone was successful endeavor so the company choose to proceed with the project.

Once the decision to proceed with the project occurred, the next step was to port all the desired applications to the HP-UX machine. Once this was completed, the client/server element was implemented, that is, networking the IBM and HP box together, implementing the client EDA/SQL API calls in the UNIX FOCUS procedures to obtain the required mainframe data, and testing these client/server applications. Finally, the last component was implemented, that is, integrating the client EDA/SQL API calls in the mainframe FOCUS applications.

Milestones: Rightsizing - Client/Server

1. Choose UNIX hardware
2. Install UNIX (HP-UX) hardware
3. Install software (FOCUS)
4. Port targeted data and application to UNIX from Mainframe
5. Test application
6. Proceed with porting all targeted applications and data to UNIX platform
7. Install TCP/IP on IBM mainframe
8. Implement (one direction) client/server (EDA/SQL) APIs into FOCUS applications on the UNIX machine (UNIX ---> mainframe)
9. Test client/server application
10. Implement (reverse direction) client/server (EDA/SQL) APIs into FOCUS applications on the mainframe machine (mainframe --->UNIX)
11. Test client/server applications
12. Test entire client/server (bi-directional) applications

Figure 4

Note: It should be noted that EDA/SQL has been incorporated into over 150 products on the PC including LOTUS DataLens, Excel, Clear Access, HP's Information Access and many more. Therefore, if this company decides to integrate PCs into their architecture creating a 3 tier client/server architecture

(EDA/SQL supports 3 tier client/server architecture) they would be in an excellent position to enable users to continue to maintain their investment in their favorite PC tool.

Original Configuration

The company was running an IBM 3090 model 600J with FOCUS applications with the FOCUS database (80 supported and maintained FOCUS databases). Users utilized PCs, with terminal emulator software, to link into the IBM mainframe.

New Configuration

HP-UX 9000 Series I40 with 256MB RAM with a floating Co-Processor with disk arrays with fiberoptic interface, Ethernet LAN Interface, TCP/IP, and GlancePlus, an HP performance analysis tool, and HP 9000 Series 720 workstations.

Note: The company chose the I40 for this project. They anticipate an upgrade will be required to accommodate the large user base.

FOCUS and EDA/SQL were installed on the HP-UX machine IBM 3090 model 600J was maintained, and TCP/IP was installed on the IBM mainframe. In addition, EDA/SQL was installed on the mainframe computer.

Configuration	
<i>Original</i>	<i>New</i>
<u>Hardware</u> IBM 3090 Model 600J PCs linked to IBM	<u>Hardware</u> HP 9000 Model I40 with Floating co-processor Fiberoptic interface Ethernet LAN Card,TCP/IP IBM 3090 Model 600J w/TCP/IP
<u>Software</u> FOCUS with FOCUS database	<u>Software</u> HP-UX: FOCUS, EDA/SQL Mainframe:FOCUS, EDA/SQL

The time to rightsize did indeed turn out to be about 18 months.
The major elements included:

- 1) Evaluate, choose and install UNIX (approximately 3 months)
- 2) Port software application (approximately 2 months)
- 3) Port all targeted applications (approximately 3 months)
- 4) Test ported applications (2 months)
- 5) Implement client/server (EDA/SQL) API calls into FOCUS UNIX applications (approximately 3 months)
- 6) Implement client/server (EDA/SQL) API calls into FOCUS mainframe applications (approximately 3 months)
- 7) Test entire configuration (2 months)

Challenges Uncovered

There were essentially no obstacles identified during the rightsizing, client/server implementation. However, one challenge that had not been resolved during the process was system management. They will investigate system management tools when reliable, tested, distributed system management tools become available.

Note: As system management and system management tools are important elements for a true, production, distributed, client/server environment, below, is a brief discussion of products that are available in the marketplace today.

System Management Tools for UNIX and distributed systems

System management tools on the mainframe are reliable and many choices of tools are available. The limited system management tools available on UNIX are new or ported directly from the mainframe and do not take advantage of the distributed integration capabilities of UNIX as well as other operating environments.

But now, this is beginning to change. A number of traditional as well as new system management tool vendors (Tivoli Systems) are beginning to provide distributed system management tools.

For your information, the following is a list of vendors who provide system management software. This list may be instrumental for evaluating system management tools if you are considering implementing a client/server, distributed UNIX environment.

	Planning/Modeling	Performance Management	Fault Tolerance & Operations	Configuration		
				Management (Hard/Software)	Accounting Management	Security Management
Boole & Babbage Inc.		X	X			
Computer Associates International Inc.		X	X		X	X
Compuware Corp.	X	X	X	X	X	X
IBM	X	X	X	X	X	X
Hewlett-Packard Co.		X	X			X
Landmark Systems Corp.		X	X			
Legent Corp.		X		X		
OpenVision Technologies Inc.	X	X			X	X
Tivoli Systems		X	X			

Source: DATAMATION: December 15, 1993

Results and Comments

This company was successful. They were able to implement a client/server, open systems environment that decreased the mainframe costs that were being charged back to this firm.

The success of this company's rightsizing, client/server endeavor was primarily due to the excellent planning and process that was implemented. The comprehensiveness of the plan and the careful watchful eye of the project teams over the project management schedule (schedule was adjusted during the process as required) so there were no unanticipated surprise, were important elements for success.

Since, there were project meetings with the vendors every week for the first 15 weeks and then bi-weekly meetings thereafter, any potential problem was identified and rectified quickly and efficiently.

Today, there are over 500 users accessing the applications running on the HP-UX and/or mainframe computer. They were able to meet all their goals and objectives as they were able to implement a client/server, rightsized environment, avoid the cost of a mainframe upgrade, and ultimately reduce the high, chargeback costs, of the mainframe computer.

When and if they chose to implement an OSF DCE environment, they are comfortable that their choice of partners (Hewlett-Packard and Information Builders) will help them implement these new technological standards.

Conclusion

Successful client/server, rightsizing and open system solutions are not only possible but have proved to save money and increase efficiency of operations within many companies in today's world.

Clearly, a realistic expectation, meticulous planning, dedication of resource (both internally and externally), and hardware and software expertise are critical elements required for success.

The open systems standards (i.e., OSF, COSE, etc.) as well as tools (such as system management tools) are continuing to evolve and become available for both UNIX as well for the distributed environment. As these tools mature, they become viable options to implement for a true, production, client/server environment.

In the end, client/server, rightsizing and open systems really do mean the freedom of choice. The freedom to maintain investment in hardware, in software, in people training, and the freedom to implement new technologies and services as they emerge in the, ever changing, ever evolving, arena of information technology.

HP/UX System Administration in Distributed Computing Environments.

By Jeffery J. Hamilton.

Unix Systems Administrator.

CELLULAR ONE / Genesee Telephone Co.

1 Marine Midland Plaza, Suite 1300. Rochester, NY. 14604

Office: (716) 292-6300 ext. 4628

Mobile: (716) 747-2557

INTRODUCTION.

This paper is aimed at readers who are considering a move to the distributed computing model or have recently done so and are interested in seeing how others have weathered the storm.

I confronted the task of becoming the Unix Systems Administrator for a company that never formally had such a position previous to my hire. At the time this company had no idea of what the terms and acronyms Client / Server, TCP/IP, DCE, SNMP, Network Backup, RAID and NIS were. The company was about to embark on a journey that would involve and use every one of these (and more) and be one of the first in our industry to do so. Fortunately the company chose to go completely with one vendor and to accept, that all of the above came at a price.

At present I manage seven HP 9000 800 and six HP 9000 700 systems spread over a WAN that spans from Albany to Buffalo NY via T1, with links to Champaign IL and Pittsburgh PA via 56K leased lines.

Of the seven servers, two of them are H-70 and H-40 class and are the database engine server and its "fail-safe" server, which could replace the H-70 in the event of failure with a replicated copy of the database. These two servers are fiber linked to disk subsystems stored in a separate two meter tower which houses 20 Gigabytes of RAID-3. Additionally more than 24 Gigabytes of SCSI disk farms are used as the on-line copy of the database, located on the H70. These SCSI-II disks were replaced by an equivalent of Fast/Wide SCSI in May. The other five servers are the "Client" portion of the Client / Server equation. Each client machine is an 847 or 867 series server and is the front-end machine, where an average of 50 to 65 users per machine log on every day. This represents an average user count of more than 200 that log on daily. There are more than 400

accounts in total when all users, the I.S. Staff, application vendors, and systems support users are considered.

While all this may sound normal to some, it is further complicated by a company so mobile and dynamic that more than 25 % of the accounts are regional positions that require the ability to login at any server anytime and have it *feel* as though they are **\$HOME on the Range**.

At the time of install, no user had direct Unix knowledge or training and only a hand full of the users had an understanding of DOS. This meant that the *feel* in "\$HOME on the Range" had to be real good, or our Help Desk was in for long days (they already were because that was the early ages of client / server computing and our major application had it's fair share of bleeding edge to ride).

THE TASK (should you choose to accept it).

- (1) Create an environment where 300-500 people can login and work from any one of twelve locations in four cities across upstate New York.
- (2) The user must be able to connect with the database from an ASCII terminal or a Windows based PC running a terminal emulator.
- (3) The user environment has to work the same no matter what city or host the login happens in.
- (4) Provide for network printing so the user can print anything needed to anywhere the boss is at that moment. This includes printing from Windows, Novell, UNIX applications and environments.
- (5) Allow for a Unix Email installation that will be easy to manage, even if mostly unused by the majority or employees.
- (6) Back up all that data every night because they WILL need it and with your luck on a daily basis!
- (7) Have this environment safe for everyone involved -- computer host and user data.
- (8) Design a method to automate and simplify the maintenance of this newly created environment regardless of where you, the System Administrator may physically be on this network.
- (9) Determine a way to keep all this hardware running efficiently via performance analysis and be able to predict when and where changes or upgrades might be necessary.

These tasks took me almost a full year to realize and get to an operational state that was acceptable by everyone involved.

YOU CAN GET THERE FROM HERE.

The first step was to figure out a way in which I could easily allow 300+ users to login from anywhere with the same look and feel. I

determined that there must be some uniform way of dealing with all users in all cities.

I divided our company into groups as a first step. Each group was common to all cities or markets, so the naming scheme was kept similar. For example the Buffalo server has groups called "bacct" and "bsales", while the Albany server has groups called "aacct" and "asales" (Buffalo and Albany accounting and sales respectively). This scheme was duplicated for the Rochester and Latham offices too. Additional support groups were added such as "misops" and "misdev" for IS personnel.

The next phase was to determine where to begin the control of a typical login process. All non-IS accounts are treated this way.

I decided to eliminate all user \$HOME/.profile files as they would become a maintenance nightmare and allow for too many differences from user to user. I decided to make all environment decisions from the /etc/profile. The /etc/profile is read by everyone (not running /bin/csh) that logs into an hp-ux host, so it was the lowest common denominator for a starting place. All other dot files such as .profile, .vueprofile, and .kshrc are read after the /etc/profile provided that they exist.

Look at my example of an /etc/profile :

[PAGE 1. /etc/profile]

Notice that one of the first things I did in /etc/profile was assign GID (Group ID) and UID (User ID) variables. All branching decisions are based on the user's GID variable. Further sub branching can then be done on the user's UID variable. Read through the /etc/profile further and you will see that all decisions are based on exception. I did this so that if I were to create a new group in the future, I wouldn't have to go through each server's /etc/profile and add a new GID to trap at any particular point. Also, it is easier to manage a system if all users are essentially equal during the login process, making exceptions for only a few. As an example, unless the user's GID is equal to an I.S. or SYSTEM GID the user will be forced to run a menu shell I called "newmenu". Notice also, that all user environments (except I.S. and SYSTEM users) are set to a "nice" level of 15, which means that should a user create a runaway process it wouldn't be able to take priority over a system or I.S. process. If you "nice" the user's login shell ("newmenu" in my case) any subsequent processes the user starts are also "nice'd" to the same or higher level. Another thing evident here is that certain ID's do only one thing. Such is the case for the GID number 58 which is a "Training GID". This method allows our Education centers in all cities to do training on what appears to be our production systems but the data that these users see isn't the production data, rather a copy of it located on the "fail-safe" reporting

server. This file also checks to see what its host name is and then determines what the user is allowed to do on it. Certain servers will only let the user create and run SQL reports through "sqlmenu", while other servers will only let the user access the database application via "newmenu". One host even 'decides' what the user wants to do based on the kind of processes they may already be running.

This one file (/etc/profile) is perhaps the most critical file on my systems now. Review it further and notice other things that take place based on exception such as motd, news, filecleanup and a homegrown password aging routine for everyone except I.S. and maintenance users.

At this point, I've divided all users into groups and effected a method of decision-making based on GID's and host name. The next move is to create some kind of common user environment for everyone to end up in.

[PAGE 2: /USER_ENV/newmenu]

I wrote a script called "newmenu". This script is the shell which most users end up in, unless they are system or I.S. users. Some of the first things I do is trap the user's keyboard so they can't break out using the standard keys, go hunting for any existing processes which this user may have started in a previous session and kill them off unless they are specifically a report they started in the background, check for the client machine specifics and finally display a main menu.

The functionality of "newmenu" is straight forward and would obviously differ from company to company; however, the concept remains the same no matter what company you work for. This menu gives limited access and prevents users from hanging themselves out to dry in an operating system which would gladly let them do just that! It also assures the I.S. department and perhaps more importantly the Help Desk that users are following a relatively safe, and controlled method of accessing the applications and data that they need.

Notice the way in which I validate a user's right to use any particular part of our database. In order to simplify any "menu" system I could design, I wanted all program modules to appear to all users and be broken down by department, but only allow those users that appear in any of the "???PRIORITY.USERS" files to access the application. Everyone else is told they need to be given specific authority by their manager to use it. An additional file is reviewed to check to see if the application is available at all. This file is called "APPS-LOCKED" and it simply lists all the application types a user could use on our network. If the application is missing the "#" (pound or hash symbol), then the application is not available to any user. This allows our DBA to work on the database

associated with that application and rest assured no user is also modifying it.

You may wish to review how I deal with default printer devices. Our users have to be able to access any device in any city for any particular report they run. The list of printers is just a simple ASCII flat file but it was the easiest method for keeping a network wide list of devices. I'll demonstrate later how this file and all others are kept identical across the network on all client machines and what "network printing service" I chose to carry out the user's printing demands.

At this point, I have a basic method for allowing anyone to login at any server and be controlled basically by two files - /etc/profile and /USER_ENV/newmenu. This leaves some unanswered questions, however, and you've no doubt already asked them in your mind. Where is the user's \$HOME directory, that they could access it from any city on any server? How does the System Administrator keep all the copies of /etc/profile and /USER_ENV/newmenu the same on all these different host machines? To achieve this I make use of Network File Systems (NFS) for user \$HOME directories and an NIS-like routine for keeping /etc/profile and /USER_ENV/* the same on all client machines.

NFS \$HOME locations.

There are five basic locations across our network for a user \$HOME directory to be named "/users/{market}". Each of these file systems is located on each of the client machines. All of these file systems are also NFS mounted to every other client machine. This meant that each client machine had to have a directory called "/users/{market}" on it's "/" (root) file system to which the client's local users file system and all other remote users file systems could be mounted. Look at the example of a client's /etc/checklist file:

[PAGE 3: /etc/checklist]

The local client machine mounts it's own "/users/{market}" file system and then further down the list mounts all the other client's "/users/{market}" file systems. This provides for a simple way of allowing any user to login from any client and be able to access their \$HOME. It also ASSUMES that our WAN is never down, so there are some bugs in the system yet. The easiest fix for this is a redundant link for each major network span, and this plan is currently being reviewed. You could alternatively, use "automounter" to alleviate all the NFS mounts being on all machines all the time; however, in our case they would be anyway, due to the fact that operations are 24 hours, 7 days a week.

I am currently, working on a method that would log the user in on the least utilized server, based on the number of logins. This additional

process is needed as some of our markets have grown at faster rates than others, and the result is some client machines are almost bottlenecked while others are relatively unused. I will now be able to have each user attach to the local or closest and least utilized system at the moment they login. This process also occurs in the /etc/profile and is really quite simple. Look for an example of this routine (loginhost) later.

WHAT TYPE OF HARD(ware).

The type of hardware that any user can login from includes "dumb" terminals or a Novell networked Windows PC running a terminal emulator. The dumb terminals are all connected directly to terminal servers such as an HP DTC, located in each market, which in turn connect them to a Unix host of choice. The PC's are all connected to our network via 10baseT ethernet to a local Novell server, where they have access to various groupware applications, Email and terminal emulation software. The desktop Windows environment is strictly controlled; it is run from the Novell server at the time the user logs in and cannot be modified by the user. This gives the Novell Administrator the ability to control what default group and application icons are available to any user and how they will operate should the user select them. The user still has modification control to the basics of Windows, such as their screen saver, desktop layout and other similar things.

STAYING IN SYNC.

I have created a directory on each client machine called "/USER_ENV". Each directory is actually a copy of the one located on the Unix Administration Server (UAS). If you are familiar with Network Information Service (NIS) then you're already aware of how I keep my copies of all data in the "/USER_ENV" directory the same on all clients. The principle is the same as NIS, with one location to do maintenance and propagation of all the changes to all other hosts. Look at the /USER_ENV/ypuserenv file, which does the actual propagation of the directory structure:

[PAGE 4: /USER_ENV/ypuserenv]

The UAS is one of the machines which has an entry in all other server's root .rhosts file. This means that the user "root" on the UAS can write to any other client via the remote utilities "rcp" and "remsh". Since I make all changes only on the UAS I wrote the script /USER_ENV/ypuserenv to rcp (remote copy) all files contained in this directory to all other client machines. This script first moves some special files that only the UAS machine needs out of the /USER_ENV directory to root's \$HOME and then systematically copies the other files to the appropriate place on the

other hosts. The benefit is central management and distributed use of exactly the same files.

This "ypuserenv" routine is applied to every file in the directory "/USER_ENV", which has a few implications. All files or scripts located in this directory must NOT be server specific, and every script that could run from here must determine which client it is running on so it can make the appropriate changes to function correctly on any destination host. This means that writing any script is more difficult but it's worth it for the results. If you can apply the idea I've seen on bumper stickers, "Think Globally. Act Locally.", to your script writing you'll be far better off.

NETWORK PRINTING.

RING ... "Hi Lynn. It's Richard. I need a print out of that analysis we did for the Latham market yesterday. How soon can I have it?"

Does this scenario sound familiar? Sure.

Does this sound like a difficult task? No.

Now consider this. Lynn is at her desk in downtown Buffalo, where she and Richard created this report yesterday. Richard, however, is now a few hundred miles away in the Latham sales office, in a meeting with all the local big wigs and needs this report bad.

Does this scenario still sound familiar? Probably.

Does this sound like a difficult task? Depending on your installation and level of user sophistication. Yes. Why? If you are running a normal printing environment, the chances of a user logged in to a server in Buffalo printing to a printer 300 miles away in Latham are very unlikely. You may be lucky and have already created a remote lp device on the Buffalo server which is pointing to a printer hanging off the Latham server. If you are real lucky it will even be the correct device type (laserjet, deskjet or impact) for this particular job.

The odds are not in your favor as the System Administrator that this will be the case. As time goes by you are more likely to run in to a situation where you'll either have no remote devices or you'll have so many that it will become a full time job dealing with them. In any case, there are other scenarios which bring up equally frustrating problems. What if this same user wanted to print her Word for Windows document to a color, high resolution deskjet two floors below her which is connected to the network via Jet Direct interface?

The real solution is to utilize a *network* printing scheme. This isn't always easy and I can honestly say I spent a majority of my time resolving this one issue.

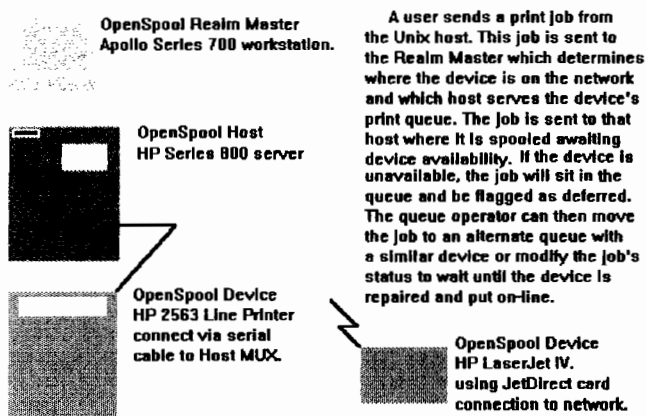
After much research on network printing services, at the time. I had only three choices (There aren't many more at the present). Of the three I

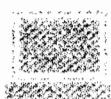
looked at, I picked a package called OpenSpool. The primary reasons for choosing this package were (1) that it was created for HP machines with allowances for SUN gear and (2) the interface to OpenSpool is very much like HP's SAM interface is. With the addition of two programs (LPR_GWY.NLM on Novell server and lpr on Unix) to create a Novell to Unix, or Unix to Novell gateway, I am now able to handle print jobs from both operating systems and service them with the same printer devices.

OpenSpool offers a central point of management for every device on the "realm". This means I do all configuration, modifications and daily maintenance on one machine (the OpenSpool Realm Master), then propagate these changes to all other servers in the "realm". At any time I can log in to the OpenSpool Realm Master and run the interface called "npui" or "npuix" and watch the entire network of printers.

Now, if a user wants to print their Windows Excel spreadsheet to a color deskjet next to them or in another city, they can. The user doesn't have to worry about host name, network path or any other technical detail. Furthermore, if that device happens to be down at the time and the user isn't aware of that, the job will wait in the queue. The Administrator can redirect this print job to a similar device anywhere else on the network if it's needed. In addition the person performing the daily maintenance (Help Desk at my installation) doesn't need to be the root user.

Implementation of this network printing system can vary from site to site. My site is installed as diagrammed below:

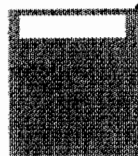




OpenSpool Realm Master
Apollo Series 700 workstation.



OpenSpool Host
HP Series 800 server

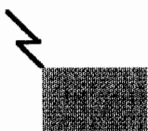


OpenSpool Device
HP 2563 Line Printer
connect via serial cable to Host MUX.



Novell Server
HP Vectra 486/66

A user sends a print job from a PC which is delivered to the Novell host. The Novell host passes this job to the lpr gateway via Flex/IP from it's local spooler. The lpr daemon sends the job to the OpenSpool Realm Master. The Realm Master determines which Unix host serves the device queue and sends it there. All the same controls can be applied to this job as in the Unix to Unix job.



OpenSpool Device
HP LaserJet IV.
using JetDirect card
connection to network.

As you can see by the diagrams above, the system can be as simple or complex as you need. In all cases, I have found it to be very reliable. The only caveat is in regard to reconfiguring the Realm. If you decide to add another printer or queue, you will stop all printing services network wide during the propagation phase. This is because OpenSpool shuts down all printing and takes a "snapshot" of how the "Realm" looked. It then modifies the appropriate host and queues/devices as you requested and propagates the changes to every host on the network. Finally, it restores all devices and their associated queue bindings. Any jobs which were in the printing phase during this would be restarted from the beginning unless you specified otherwise (i.e.: you could have any jobs restarted from the page they were working on at the point of "npact"). As an example, at my installation with over fifty printers on the network, the *npact* phase takes about 35 minutes from start to finish if run during non-peak use time.

Another feature is the syntax used by OpenSpool. It uses a command structure similar to lp but calls it np. If your users were used to the lp command style, you can enable "lp emulation" in OpenSpool. Alternatively, you can disable the lp emulation part and run lp as before.

The difference here is that you would have to modify the lp model scripts to redirect the job to OpenSpool using the np syntax. This must be done anyway if you wish to make use of the Novell to Unix gateway, since this gateway only speaks "lp".

The last thing I can recommend to you regarding network printing is to make use of the newer technologies available. One good example of this is network interface cards for printers. These little gizmos can be inserted directly into the printer, as in the JetDirect card for any HP LaserJet model III up, or attached to the side of the unit, as would be required by any ordinary impact printer or older LaserJet. With this device your printer becomes a unique address on your network, and thereby liberates any machines from being the printer's directly connected host. This saves anyone from having to "share" their PC's horsepower while a print job is serviced, and it means you can locate the printer anywhere in your company that you can string a twisted-pair or coaxial cable to. Another wonderful thing about newer technologies is printer speed. The newer laserjets like the HP LaserJet IIISi and IVsi are capable of pages per minute output equal to or even greater than most of the impact printers we are running (this included the HP 2563 line printers).

I now replace all impact devices with laserjets when they break down or expire. The cost for a laserjet printer is no longer the prohibitive factor that it used to be and therefore much easier to do cost justification for when the time comes. At any rate, what good is an impact printer, if no one uses it?

ELECTRONIC MAIL

Email by itself can be a real nightmare just to use, never mind manage. This was the case at my site. I had five different 800 series servers and six 700 series workstations all supporting users directly.

What I did to solve this wasn't elegant but certainly worked. I am running sendmail on all of the client machines and workstations. I picked one machine to be the network Email host and then NFS mounted its "/usr/mail" directory to every other host on our WAN (see the /etc/checklist file listing from one of my hosts above). Now the exact same mail directory is used by everyone and thanks to NIS, all user account names can be resolved locally.

BACKUP NOT DOWN

Backup is probably the most discussed issue amongst System Administrators. How many times have you heard (or even told) some of the horror stories about backups?

We are all familiar with the standard Unix backup utilities, such as tar and fbackup, which are supplied with hp-ux; there are all kinds of homegrown in-betweens too. The problem with most of these solutions is the reliability factor. Most of them, do not verify the media; they simply read the data off the disk and write it out to the media. If the media had a bad block, your data is written over it; so much for your backup. All of the supplied methods only backup local file systems to local backup devices; fine for most, but when you get into Client / Server and start working in a Distributed Computing Environment, this is no longer good enough.

I have servers remote to me and I can't always count on someone at these sites to diligently change the media daily. All of our 800 series servers have the 2 or 8 gigabyte DAT drives internally mounted. I also have additional 8 gigabyte DAT drives on the H-70 and H-40 machines for backing up the database engine.

I chose a backup utility called OmniBack to do all my backups. OmniBack is made by HP and I selected this software because it gave me the capability of backing up any local or remote file systems to any local or any remote backup devices. An added bonus is that OmniBack has its own user -"omniadm", which is capable of performing the backups even though "omniadm" isn't a root user.

At present, if a remote host fails to backup because the media wasn't swapped that day, the night operator in my department will get an Email message telling him this. He can then backup the system remotely from the central location in Rochester.

Another desirable feature of this software is concurrent sessions. Our on-line database is currently stored on 24 gigabytes of Fast/Wide SCSI disks with the redundant copy stored on 20 gigabytes of RAID-3. The backup scheme supplied with the database utilized tar to perform it's backup. The original scheme required up to seven hours to complete and used thirteen DATs (DATs cost about \$20.00 per unit x 7 days x 4 weeks + 1 set each week off site = beaucoup dollars)! I wrote my own script (see /etc/omniback/checkpoint script) to do this backup using OmniBack, and reduced the media usage from thirteen to two or three DATs and from up to seven hours to barely three. I was able to do this by using OmniBack's concurrent sessions and local and remote backup devices functionality. Now, when the night operator runs a backup of any database he loads the two local 8 gigabyte DAT drives and/or one remote 8 gigabyte DAT drive and then starts the script. Within minutes every DAT drive is humming along at the same time. This alone was worth the purchase price of this software. The fact that OmniBack keeps impeccable logging of everything it does is an added bonus. If anything goes wrong anywhere, it logs it and

I have my backup routine Email the operator the summary results. If he sees any indication of problems, he can easily review the detail logging to determine what happened and why, taking any necessary steps to correct the situation. One more benefit of this software is that it verifies everything it writes to the media, by default which is probably the most important thing any backup software could do.

If OmniBack has any pitfalls, it is the command line syntax. It's very unforgiving and fairly difficult to understand, not to mention long. But, all in all, I'd rather have long, cryptic, unforgiving command line options than have to face the Big Guy with some sad story about bad blocks and lost data.

The following is an example script I wrote to perform the /users backup of each client machine. The neat thing here, once again, is that it's the same script running on each machine. The script has the routines to figure out which server it's on and then backs up the appropriate file system.

[PAGE 5. /USER_ENV/users.backup]

[PAGE 6. /etc/omniback/checkpoint]

PRACTICE SAFE COMPUTING.

Having reviewed how I set up the typical user environment, you can see that not much could be done by the average user to lose data or cause the system harm. At this point the only people that could do damage are I.S. staff. To help prevent this from happening I decided that a few maintenance login shells were needed.

In the early days of our installation, it became apparent that our application had a nasty habit of creating runaway and zombie processes when a user powered off their terminal or abnormally exited their active terminal emulation window instead of logging out properly. This meant I spent at least an hour every day killing runaway and zombie processes. While anyone with root privileges can do this with little difficulty, I ran into problems with other staff in my department while I was on the road. The problems were created when an IS person would use the X environment capability of "cut and paste" with the mouse. Typically, the IS person would not cut the whole P.I.D. from the listing of processes for a runaway. This had the effect of killing an incorrect P.I.D., resulting in the loss of another (usually system-type) process, not the one they had identified as the culprit.

My solution was to create "maintenance login shells". I created a user called "torch" that has a user I.D. of "0" (root). This user has a login shell "/USER_ENV/poochem" instead of "/bin/c|k|sh" (the script "poochem" is shown below). The shell permissions are set to 750 so only a "root" user could execute it. It also traps the user keyboard so they can't break out of

the shell and drop to the command line. By doing this, I was able to give Help Desk, and anyone else in our department who is privilege to the password, the ability to kill off specific users or runaways. I later added some other functionality to assist in a few other routines that were "root" privilege required.

Another "maintenance shell" I created following the same scheme was "diagman" which has "/USER_ENV/diagmenu" as it's login shell. Similar precautions were taken with this shell, in that it was only executable by a root user and the keyboard was trapped. This shell allowed Help Desk to test network connectivity and use some utilities that only work for a root user. The ability to test and verify site to site or system to system connectivity becomes extremely important when you get into Distributed Computing / Client Server because all the user will tell you is "my terminal is slow" or " {application_name} has stopped". An example of "diagmenu" is listed below too.

[PAGE 7. /USER_ENV/poochem]

[PAGE 8. /USER_ENV/diagmenu]

There are a few other precautions I took on the machines to help prevent malicious use. I have set up all the servers so that the user root's ".rhosts" file is only one way. For example: all servers "/root/.rhosts" file have the UAS name and root user listed in them for propagation of the "/USER_ENV" directory structure; as does the NIS Master machine, however, the reverse is not true. The /etc/hosts.equiv file was similarly worked over. This means the user root on any other server or workstation is not equal to the user root on any other machine. This concept helps to prevent potential break-ins from any other server's root user. You may also have noticed that when I specified "/root/.rhosts" file above I didn't type "/.rhosts". The user root has it's own \$HOME directory which is not the "/" (root) directory. This is an easy and good practice to follow. The fact is that almost 80% of all system errors are made by someone logged in as root. By making the user root have a separate \$HOME directory from the operating system's "/", you can prevent "hp-ux" or "SYSBCKUP" from being deleted by some well intentioned but careless root user trying to clean up. Don't laugh at this one, I've witnessed it.

There are some other practices I'll mention, although I'm sure that you are already aware of them.

- Never have "." (dot) in root's \$PATH.

- Check your systems login files (/etc/wtmp | btmp | utmp) frequently. By doing it often the amount of data to sift through is less daunting. The /etc/wtmp file contains all the valid logins, while /etc/btmp file contains all the failed attempts. To "read" these binary files you type "/usr/lib/acct/fwtmp </etc/btmp | grep root | more". This will display a

listing of all the failed root user login attempts by ttyport, date, time and from where the attempt was made. You would obviously be skeptical of any date or time which doesn't conform to normal business or on-call hours and from any locations which aren't familiar to yourself. Verify that the port I.D. of your modem isn't appearing at any unusual hours too. The information displayed is about 100 columns wide so you may wish to resize your terminal window before running this command to make it more readable. I've included another script I wrote to help make the maintenance of these files easier. This script "/USER_ENV/accheck" will read in the files mentioned above, extract the last ~36 hours of data and then reinsert it. This will reduce the size of these files which grow without bound to about 4500 bytes. The script also places a copy of the entire extracted file into the root user's \$HOME directory for later review and backup.

[PAGE 9. /USER_ENV/accheck]

- Use more than six alpha **AND** numeric characters for root's password and change them often.

- If you have dialup capability, make sure you use the dialup password facility (see the man pages on "d_passwd"). and change this often too. HP usually supplies one "support" modem with each machine you put under support agreement. I decided that I would not put a modem on every machine. Instead I connected one 2400 bps modem with dialup password for HP support and another 9600 bps modem with dialup password **AND** dial-back on it to the same machine for I.S. on-call support use. Adding dial-back is a very good idea, and having at least the dialup password functionality enabled will prevent the majority of abuse. Most "intruders" will not bother trying to crack so many passwords because "there are too many other fish in the sea" that would be easier to compromise.

- Force your users to change their passwords at a reasonable frequency. This is not as easy as it seems. If you run stand alone machines then you can make use of "password aging" by modifying the password field in the /etc/passwd file. Unfortunately, this method didn't work with NIS. There are patches you can get from HP to work around the limitations but I created a script to force a user to change their password on a given number of days since they last changed it. The idea I had with "/USER_ENV/agepasswd" was to warn the user 10 days prior to their password expiring that it would do so. This gives the user a chance to start thinking about a new password.

[PAGE 10. /USER_ENV/agepasswd]

Another process to use with reasonable frequency is "pwck and grpck". These utilities check your password file and group file for common problems such as bad account names and inconsistent \$HOME directories. To use these you simply log in to the NIS master (or any local host) and type: "pwck {password file} [ENTER]". These utilities report all problems to standard error so using "|" more" isn't easy if your password file is larger than 24 lines. The following works for me: "pwck /etc/passwd >password.check 2>&1 [ENTER]". This will redirect the output from pwck to a file called password.check, which you can then cat or more to your screen or print for review.

The last issue I'll discuss about "safe computing" is the one we probably all hate the most - documenting. I was like many of you, in that I never documented things very well. I figured I'd always remember "this" when I next needed to use it. Bad plan!

If you were to suffer a bad disk crash (just to make it fun let's say it was on your server's root disk) would you be able to rebuild the new one to just the same state as the crashed one before it went south? It is a very good idea to keep a hard copy of all of your file systems set up and layout including disk types, a copy of the "ioscan" report and the /etc/checklist and mnttab files. To facilitate getting and keeping a copy of all this data, I wrote a little script called "/USER_ENV/getdiskinfo". This version of "getdiskinfo" works on both standard and lvm file systems, however, the lvm reported file systems require a little work to match physical disk information to file system name and mount point.

[PAGE 11. /USER_ENV/getdiskinfo]

You would also do well to make a hard copy of every file that is involved in your boot and shutdown streams. These files include: /etc/*rc*, /etc/conf/S800 - or what ever file you use to generate your kernel and all files in your /etc/shutdown.d directory. All this sounds pretty useless now, but remember this: "There are only two kinds of System Administrators. Those who survived a severe system crash and those who didn't.". You can bet those that did survive, either had a hard copy of all the files mentioned above or a functioning backup of them on tape that they were able to restore without errors (see the section above on BACKUP NOT DOWN).

AUTOMATIC AND SIMPLE MAINTENANCE.

To make maintenance simple and easy is the one goal all System Administrators share. There are probably as many different methods as there are SysAdmins and in the end anything in this direction is better than nothing.

For my installation I had a few concerns that I wanted to automate and simplify. These are:

User account creation, modification and maintenance. File system maintenance. Printer Utilization. NIS maintenance. Application availability. System performance (see next section: SYSTEM PERFORMANCE). Company specific goals - in my case cellular switch accessibility.

To deal with most of these meant writing shell scripts which I could then apply to cron for timely checking and notification. The one exception to this is the user account part.

User account maintenance can be static, in which case you make user X's account and that's the end of it. The company I work for, however, experienced a 15 to 20 user growth rate per week, with another 5 to 10 changed or modified weekly. These rates of creation and modification are primarily due to the industry I'm in, but you may have similar or even greater demands in this area. The way to get this part of your job under control is to control the process by which account duties are initiated.

In the beginning, I had numerous managers and supervisors calling me directly every day with a request for another account. Typically, they "needed" this account yesterday. I did this for a while and then realized that it was a huge waste of time going through the procedures to create or modify accounts every other hour of every day. I then modified the request process to stop this situation from growing even further. Now when a new hire is accepted and they come in for the final interview, they are given a "Login Request Form". It is a "fill in the blanks" style of form and basically makes the new hire's manager answer all the questions I'll need answered to create the account. It also includes a line which states that the user will not share their password or compromise system security by allowing others to use their account with a space for the new hire to sign their name stating that they understand these conditions. There is a similar form which managers must fill out to have an existing employee's job function/description changed. Making the managers take responsibility for this takes the error rate out of the "wrong group or spelling of user name" phone calls you'll get later if you follow the "phone-in" method. Next, I have these forms submitted to the Help Desk rather than to myself directly. This step means that other people in my department can be made aware of the impending changes in user status. The Novell Administrator can make sure there is a PC ordered or moved to the new location of this employee and that a Novell account is in place. The DBA can make sure the user is granted appropriate access level to the database(s) in question.

Each of the IS staff has a small check box on the same form to verify and date that they have taken the necessary steps.

The final change to this process is "when". At present, I do all account maintenance only once a week. All account modification or new account requests must be submitted to the Help Desk weekly, no later than Tuesday afternoon. If this stipulation is met, then the user is guaranteed of being able to login the very next morning (our new users go through a two day initiation class, thus the Tuesday deadline).

The end result? I spend one afternoon creating and modifying accounts in an almost mechanical work stream fashion. Doing it this way means I can open all the required windows on my workstation to run all the necessary processes and plug in all the new or modified data one after another. It saves a great deal of time and lessens the chance of making a mistake or missing a certain procedure when compared to the ad hoc method of before.

File system maintenance is automated, as much as it can be. Typical maintenance functions are:

The /tmp and or /usr/tmp directory clean up. Removing core files created by users. User data file clean up. File system status checks (how full / empty).

I wrote a script called "/USER_ENV/dumptmp", which runs each night prior to backups to list files in the /tmp and /usr/tmp directories and then delete the files which are not needed by any other system functions.

[PAGE 12. /USER_ENV/dumptmp]

The removal of stray core files is dealt with in a similar manner to "dumptmp". The script is called "/USER_ENV/dumpcoredump". It runs prior to backups too and simply uses the utility "find" to locate any core files on any local file systems and deletes them. Any core files and the directory path to them that are found are accumulated in a file which is Emailed so that I can determine if anyone in particular is creating an unusual amount of core files. If this is the case, the assumption is the user probably needs more training with the application they are using or they are experiencing some kind of technical problem which must be resolved.

User data files are one of the most difficult clean up routines to deal with. How am I supposed to know if this two and half year old file is needed by the user or not? The answer was to put "housekeeping" in the hands of the users. I wrote a script called "/USER_ENV/filecleanup" which is invoked every time a user logs onto a client machine. This script simply checks every file found in the user's \$HOME directory to see if it is older than 29 days. It compiles those files which match this criteria and then systematically forces the user to accept the deletion of them one by one. The user can choose not to delete the file(s) if they wish.

[PAGE 13. /USER_ENV/filecleanup]

Another thing I wanted to automate was checking file systems for disk space availability. Again, another script came to the rescue. It's simple but effective and I always know if or when a file system is going to fill up beforehand, so I can take the necessary steps to prevent it. This script can be passed a variable, which is the number associated with the amount of capacity to reach before beginning the warning Email to me. Some of my file systems are static but consistently maintain a 90% capacity used. These file systems have the parameter set to "95". Other file systems such as the /users file systems never reach 75% and if they should hit 80% I want to know.

[PAGE 14. /USER_ENV/isdiskfull]

Printer utilization is not really a daily maintenance task like file system maintenance, but I wanted to know more about what kind of printer utilization was taking place. This allows me to analyze usage levels and evaluate requests for new equipment. I made a two line modification to the lp device model /usr/spool/lp/interface/{printer_model_name} and OpenSpool's equivalent of the same file. This two line modification echoes to a flat file in the /tmp directory the name of the user's print job, the printer name and date. Another script reads each night from all client machines (prior to the backups running) the daily collected data and compiles it into one comma delimited file on the OpenSpool Realm Master machine. This file grows each night adding the date and the number of print jobs each device has serviced for the day. The completion of this script is the ftp'ing of the file to my Novell \$HOME directory. I then import the data into Excel and graph out the printer usage for our whole network. I run this graph weekly and keep on file the monthly statistics. From reviewing the historical graphing I was able to determine that six different printers are not being used anymore. These particular printers turned out to be the oldest impact printers our company owns. The net result of my findings was the removal of these unused printers and queues to alleviate some of the load associated with the systems's monitoring of them.

NIS maintenance is not really an issue, since all work is done at the NIS server and propagated only when changes to things like /etc/group, /etc/passwd, et cetera are made. I did run into one problem, however, which warranted a quick solution. Our WAN runs over T1 from Albany to Buffalo with Rochester as the mid-point. The T1 span from Rochester to Buffalo has a portion which is carried via microwave. This portion has experienced problems related to extreme weather conditions and cell site maintenance. If this span goes down, the Buffalo server loses it's NIS connection and can no longer answer any NIS related questions, such as

verifying that a user has correctly supplied a password to their login. After about five minutes the server, which is classed as an NIS_SLAVE_SERVER and NIS_CLIENT in "/etc/netnfsrc" will reset it's "NIS_MASTER" to itself and will remain this way until manual intervention. Since the time span of the T1 link being down has never been more than approximately 15 minutes I decided to write a script which checks each client machine's "NIS_MASTER" setting every half hour. The script, invoked by cron, runs the command "/usr/etc/yp/ypwhich", and verifies the response to be correct. If it should find the answer to be incorrect it executes the "/usr/etc/yp/ypset {NIS_MASTER_name}" and rechecks to see if the client has correctly reset itself. If the reset fails again the script sleeps for five minutes and runs again. This routine will retry up to six more times to reset, which will take just over half an hour - long enough normally to get the link back on-line. By running this script from cron, I never have to worry about resetting any clients manually. This comes in very handy especially on those nights where our cell site technicians are burning the midnight oil to do site maintenance.

Application maintenance is really not maintenance of the application, but rather a method of checking to see if any particular application is *available* to our users. In the past, our DBA or application support specialist would simply "chmod" the major binary for any particular application from mode 555 to mode 500 so that any user who tried to access the application couldn't execute it. This system is OK for stand alone servers, but the client / server model becomes much more work, since the person trying to put a lock on any application would have to login to every client machine first. The other problem with this system is its lack of elegance. It gives no explanation as to what is going on. The Help Desk, of course, starts ringing off the hook. To circumvent this method, I created an account called "applock" which is a root user that can login only to the UAS and has a shell (/USER_ENV/lockapp) which displays a list of all applications available on our Unix network. Each application can be "locked out" by entering the application name into the shell's prompt. This shell then modifies a flat file also located in /USER_ENV called "LOCKED.APPS" and propagates this file to all other client machines. This flat file is a simple list of the application names. If the application name is preceded by a "#" (pound or hash) character then the application is available. If it is not, the application is "locked". The /USER_ENV/LOCKED.APPS file is looked at by my "newmenu" shell when a user selects any application choice and will either let the user run the application or will tell them that the application is currently unavailable and to call the Help Desk. This system of

managing user access makes maintenance of the application much easier, since the person doing the application maintenance can easily and quickly lock or unlock the application's access by users and the users themselves aren't left to blank or flashing screens when they access an unavailable selection.

Company specific goals would be anything that your company does that equates to maintenance work for you. In my case, one company goal was switch access by users. In the cellular phone business, we have hardware called "switches" which make the transition of cellular communications via microwave to land line connections. These switches record every activity that takes place from the time a customer presses (SND) until they press (END). The switch is frequently accessed both automatically by our application and by users directly to modify the switch for what it knows about our customers. The problem was that our users frequently "exited" the switch incorrectly or were maybe victim to a network drop, leaving the ports into the switch in a locked state. My goal was to design a method for monitoring the switch and determining whether or not it was locked with no reason to be (i.e.: in a locked state, with a user no longer connected). This was reasonably easy to determine manually because the physical MUX port through which a user accesses the switch from the client machine was identified by a locking file (i.e.: LCK..ttyp05). If this file exists, then that port to the switch was presumably active, with a user. This lock file ownership would be set to the user that was accessing the switch from that port. If the lock file existed then I would list all the processes running and grep for the UID of the lock file. If no process was returned then I knew the switch was locked up. To alleviate the situation, I wrote a shell script and submitted it to cron on each client machine. This script runs every 15 minutes. It simply does what I would have done manually. It checks for the presence of the switch lock files and the related user processes which should also be present. If it finds a lock file and no corresponding user process then it removes the "LCK...ttyp##" file and verifies the switch interface process is running. If it finds the switch interface process wasn't running either then it restarts it. This sort of automation is cheap and extremely desirable, reducing on-call support levels.

SYSTEM PERFORMANCE.

When my company first committed to the "Client / Server" model of computing I had no idea of the difficulty involved in performance monitoring. Help Desk would typically get calls from users in a particular market saying "{the application} was very slow or seems to be hung". When this happened, the DBA would generally login to the database

server and verify the database was running properly, with no table locks capable of causing such a problem. If she found this to be true, she would inform me that the database wasn't the problem. Then I would spend minutes or hours logging in to the various client and server machines to see if I could find something wrong that would cause this kind of performance. Most everyone is familiar with the standard hp-ux commands available for performance monitoring. These utilities include: vmstat, iostat, netstat, uptime, sar, ps and so on. It got to the point where this sort of diagnosis could use up a vast majority of my day and didn't always reveal anything in particular that looked out of the ordinary - what ever ordinary is, when you are running in a distributed environment.

After going through these procedures for many weeks with generally limited success, I happened upon a product announcement from HP about their latest release of "Glance" (later, Glance Plus). I called HP's 1-800 number and ordered a copy of their "demo performance Pak". It arrived and I installed it on the client machines and on the database server. Within hours of installing this package, I realized this was the way to go.

Glance was a breath of fresh air. Glance is designed to show you performance statistics in reasonably organized bar graphs or screens which you can review real time or "at a glance" and see what is going on with anything from CPU, Memory, Disk, Swap, File System, Network and even kernel parameter utilization. This package is worth its weight in gold and very much worth the purchase price, given the hours I had been spending to find the same information.

I got the approval to order Glance for every client and each server we had on the network with little or no difficulty. Fortunately for me, the Director of my department had spent many hours doing System Administration in earlier days and was equally impressed with this new tool. Since then, the good folks at HP have substantially improved the Glance product.

Upon installation of this package, it took no more than a week to find all the real problems we were experiencing with our distributed computing environment and come up with the fixes to stop them. This package actually allowed me to prove to our application vendor that their coding wasn't extremely clean and was actually allowing zombie and runaway processes to take over the client machines, causing horrendous performance. Suddenly, the Help Desk was receiving far less calls about performance concerns.

About three months after I purchased Glance, I discovered that HP had developed another package called "Laser RX/UX" that enabled the client and server machines running a daemon called "midaemon" to record the performance data in binary files. These files could be downloaded to a PC

where I could do historical analysis, reviewing as much history as I wanted and allowing me to "zoom" in on particular points of interest, right down to a five minute detail. The concept of this package was very good, but I found it very perplexing, having numerous bugs associated with the PC side. The newer release is called "Perf / RX" and runs on my workstation. Additionally, HP modified Glance and Perf / RX to be much leaner in their separate approach to collecting the data. This meant that these utilities shared the same data collector daemon to lessen the loading they put on the machine. The redesigning of these utilities also combined them into a much more logical concept whereby they share the same directory structure for statistical recording. These packages can be purchased individually, or as a "Pak" which is probably the smarter way to go if you are going to consider any of them. They can be integrated with HP's "OpenView" software and are, in my opinion, still far better than any other offering made by any other vendors out there. Of course, this should come as no surprise to most of us. Who should know better than H.P., how to best analyze H.P. hardware or hp-ux?

The data collection made by the "midaemon" can be configured by the System Administrator to pay particular attention to any process or user the System Administrator wants through its configuration file. This file (shown below) is read by the midaemon upon initialization and records its findings under the groupings specified. The new "X" version of Glance Plus called "gpm" or Glance Performance Monitor will even display these same "groups" and their associated consumption of the system real time. Thus, I can now look at gpm and see what the group "Customer" is doing at that moment or look at historical data for say the past week of the same group in Perf / RX.

System Administration has become MUCH more convenient and far less tiresome with these tools. Another important thing that Perf/RX has afforded me is the ability to see when any hardware is going to be outgrown before it does. This fact came true this year, but I knew last fall it was going to happen. I had a short technical meeting with the Big Guy, showed him my findings and instantly, an entry into the 94 budget, first quarter - "Upgrade Database Server" and "Disk Subsystem Upgrades" was made.

[PAGE 15. /usr/perf/asrv01/parm]

[PAGE 16. gpm screens - MAIN, CPU, SYSTEM TABLES, DISK]

The only thing left to do was automate how I could collect all the historical data into logical formats for review and archiving. To do this I wrote a script called "/getperfdata" which runs weekly on my workstation. This script goes out to each host and extracts the past week's performance

data and then downloads it to the UAS. It stores the data in time-dated directories, creating the file name based on the date and server name.

[PAGE 17. /getperfdata]

So now you know how I keep an eye on my machines. What I haven't discussed is the kind of performance issues you should be aware of when you get into distributed computing.

First and foremost is **(1)bandwidth of the network**, which will experience considerable use. **(2)Your disk** access via the host bus adapter may have to be upgraded and your current disk utilization redesigned. **(3)Your network interface card** on the "server" machines may require the addition of redundant units. **(4)You'll probably have to add more memory** on some of your machines. These were just some of the upgrades my machines had to undergo in order to get to satisfactory performance in client / server.

The **network bandwidth**⁽¹⁾ between the clients and the servers were 56K leased lines. This would have been sufficient for just TCP/IP traffic and just the number of users we had when we started; however, if you wish to combine Novell's IPX (a real bandwidth pig) or allow for user growth, forget it! We currently keep T1's between the markets running at the 60% utilized range with spikes up to 100%. This is going to become more consumed as we start phasing in our retail stores and add more "clients" to the four levels of client / server relationships now. You may also have to consider network interface redundancy too. Once you start moving the clients out farther and farther away from your server, the possibility of a network link dropping out for even a few minutes seems to grow on a scale closer to exponential rather than linear.

Disk storage⁽²⁾. The original design of the database engine was to make use of fiber linked RAID-3 disks. Good idea, but wrong implementation as RAID-3 is reasonably slow when not used in sequential access methods. It's great for backup or reporting copies of the database, where speed is not a concern. It is not good for on-line transaction processing, where you can have multiple persons banging against the same table for information. You can expect anywhere from 10% to 25% slower response at level 3 RAID than compared to plain SCSI. The DBA and I were able to boost our performance with the RAID's only 10% (with the help of Perf/RX), but that was still unsatisfactory and took almost one month to achieve. These RAID's will soon be taken off-line and changed to a non-RAID configuration making use of four separate HP/FL channels - one for each subsystem. They will then be used as reporting and redundancy for the on-line copy of the database which is now running on Fast / Wide SCSI. We had to go to F/W SCSI just to keep up with the newer server's ability to handle more IO's. You may even go with distributing your

database over your network, having the portions that mean most, closest physically, to those that make use of it. Replication also comes to mind. If you wish to do this, then you can bet you'll want to be looking into better disk performance and better database layout. If you spend the time now, you can save yourself lots of head aches later. One of the best performance gains came from the redesign of the database's physical layout. By distributing the database in a logically thought out manner over as many channels and mech's as possible, we've eliminated "hot platters" and "choked interfaces".

Network cards(3) may become insufficient. The original design has one interface per machine. The current consideration is two interfaces per machine. One interface will be responsible for all the "local" communication between the client machine and the desktop (another client / server relationship). The other interface will be responsible for all the "remote" communication between the client and the server where the database resides. Isolating the traffic this way alleviates the waiting some users experience when the network card has delays and retries with the WAN routers that it talks to.

Memory(4) upgrades will be a part of your future, without question. Originally, the server machine was ordered with 256 Megabytes of RAM and the client machines were ordered with 64 Megabytes of RAM. Our assumption was that the database engine would be doing all the number crunching and table manipulations and the client machines would only be doing the screen pops and local SQL query designing. This assumption was not only wrong, it was almost exactly backwards. The current implementation has all machines running with 192 Megabytes of memory and almost every client could still use more. Some client machines will need to be upgraded to 256 Megabytes of memory by year end. The database engine rarely breaks a sweat and would do just fine with 160 Megabytes of memory; however, I wanted the extra memory left there so it would never have to page out or go to swap. Currently it never touches swap for use, only to allocate it for space.

All of the performance issues above are just hardware problems and hardware solutions. What about the software side of the coin? There are pitfalls there too. As you are probably painfully aware, the road to DCE and Client / Server Computing is riddled with pot holes and some of them are big enough to drive your whole installation into without ever coming back out.

Some of the big hitters for me were simple fixes but took a while to realize there were potential problems brewing. Watch out for the (1)"**under utilized equals over loaded**" client scenario. (2)**Operating system kernel** modifications will be needed and possibly often. Be careful

of any **(3) upgrades to anything from application to operating system.** Be careful of **(4)"X"**. It marks the spot - where many installations died. Observe the uncontrolled transient user load.

(1)We originally designed a "fail-safe" unit into our installation. After installation, it became apparent that this server was wasted CPU were it to just wait for it's moment to become the "pinch hitter". I started tossing smaller tasks at it, like being the OpenSpool Realm Master (the Realm as it was in the beginning wasn't much to deal with). Since the server was only doing OpenSpool why not add all NIS duties to this otherwise totally unused resource? That was a fateful move. I didn't think globally when I acted locally. Obviously, any server capable of being your "fail-safe" is going to have cycles to spare. Tossing it the odd bone to chew on isn't a problem, but, relying on it for a daily task or multiple interdependent tasks is not a good idea. In this case that one server going down wiped out not only all printing services, network wide, but also prevented anyone from doing anything else because NIS was suddenly gone. The worst part of it was that others in the department were eyeing this same machine for development purposes "in it's spare time". This led to assured and frequent shutdowns for disk additions and other system modifications. I learned the hard way about over loading an under utilized server. Now each function is handled by a separate (smaller) engine. Some will say multiple points means multiple points of failure, but in practice (so far) this hasn't proven true.

(2)Pay very close attention to the operating system kernel parameters. Especially, if you've upgraded from older releases of hp-ux. The newer versions since 8.0 have changed in their relationships and primary settings. Yet if you've faithfully upgraded instead of reinstalled, your kernel may not be setup optimally for the current version. Additionally, you may find yourself making kernel changes which you later change back because the installation may switch which is client and which is server! If you have the chance and the tools, I've found starting with a "vanilla" kernel and working up is better than trying to reverse engineer parameters currently running - possibly for unknown reasons to you personally. An example here was when I found the kernel on some machines had a huge number specified for "nfile" and "nflocks" parms, only weeks after our installation was initially brought on-line. Someone in the department (who shouldn't of had root access) had made the modification after hours because he had read somewhere that the database would need large numbers for these. What that "story" didn't tell him was there is an interdependency between "nfile" and other parameters not mentioned like "ninodes". Later versions of HP's "sam" won't let you make changes to ONLY one of these, if you are going in a positive or

increasing value direction. Further, the increase in nflocks turned out to be completely unwarranted, but I didn't know that until I got Glance running on them.

(3)When upgrading software in the form of applications and operating system, try to spread the upgrade over as much time as possible. I cannot tell you how many times I ran into problems with our application vendor sending us a new "improved" release of some module, only to have it crash and burn on some of our clients because of "an unknown bug having to do with hp-ux compatibility of". By giving yourself time between each machine's upgrade you build in crash recovery too. In the case of our switch interface upgrade that crashed and burned instead, I had luckily, not upgraded one of the three clients it was to be applied too yet, before the vendor even had a clue as to how to fix it, I was able to copy back into place the only existing, running version on the last client to the other clients. - BUG=0, SysAdm=1. Another example of upgrade over time was the move from hp-ux 8.01 to hp-ux 9.0. Did any of you run into a "memory leak bug"? I did. Unfortunately, the problem didn't show up right away. It could take anywhere from hours to days depending on the server's load. Because I chose to upgrade the relatively unused "fail-safe" unit first as the test, I didn't find it until almost the same time as every other machine started displaying the same problems which ended with the machine locking up. Where was I, when this particular bug surfaced? On route to Latham about a three and half hour drive away to upgrade their client machine. BUG=6, SysAdm=0.

(4)In the pioneering phase, we implemented several desk top strategies as tests to see what would be the best over all method. One of the choices was to deploy Xterminals. After a few months every Xterminal on our network except for the ones used by my department were pulled. Why? Mostly, because the level of user comfort with "X" was so low it was measured in a negative number and created far too many calls for the Help Desk. Another more important reason to me, was because of the loading that "X" puts on a system. It becomes not just a question of what the actual environment requires but what that environment will let the user do too. People with little experience on a standard dumb ASCII terminal will quickly figure out that clicking on that neat little icon over there will make a window on their screen which they can access the application through. So they'll do it once for every time they step away and return to their seat because they forgot they already had another window opened doing the same thing. Have you ever had your database tell you too many people were in it and many of them had the same login name? Have you ever met a user that thought "leaving the application" meant sliding the mouse pointer out of the window onto the desktop's wallpaper? Do you really

want this well intentioned user running "X"? To answer these I'd say "Yes. Yes. No."

As a result of the "regional" nature of many users and my solution to their constant roaming - making use of NFS mounted \$HOME directories on all client machines. I have discovered that users can inadvertently over load the client machines with sheer volume of logins. When I first got this concept up and running I had an average of 25 - 35 users on each front end. However, with the growth we experienced over the past two years I am now at a point where some client machines reach 60 + logins daily. These are just the "local" users for that particular client machine. If any other market users happen to be in town that day they can drive the login count so high that the server goes into "memory debt" and begins swapping and paging and everything starts going down hill pretty fast and ugly. At first I was upgrading memory and adding more swap to keep up with it, but there got to be a point where even this wouldn't suffice. I had to come up with an alternative that would prevent the client machine from choking itself to death by allowing too many logins to happen. This is when I came up with "/etc/loginhost". It is still (late May 1994) under development but I'll show you what I have so far. The script is to be called from /etc/profile or maybe just incorporated into /etc/profile as a subroutine. However I decide to actually implement, it will be the exact same file running on all client or server machines. It's function is to check what host the login is occurring on and based on the host's name it will have two other "alternate" hosts supplied based on their proximity to the "local" host. Once this routine has completed, the script will know who the local and two closest machines are. It then checks each of the three hosts to see which is least utilized. Once it determines which host is least utilized it will redirect the rest of login to occur there. In order to make this all work with any reasonable accuracy, other supporting routines were needed. For example, the local host had to be able to determine what the other closest host loadings were. Since the process is run by the user at login time and no user has remote capabilities, I had to make sure there was a way for the login host to have this information available locally. I achieved this by way of another cron job running on the one client machine that has a file system NFS mounted to every other client and server machine. This machine is the NIS and Mail server. It's local "/usr/mail" is NFS mounted to all machines. Since this machine is the NIS server too, the root user here has permission to perform remote commands to every other machine. This is how I got the information I needed - by having a cron job run as root that checks the current user total for each machine on the network every five minutes and then recording these findings in a flat file called "/usr/mail/REMOTE-HOSTS". The last

trick was to have "loginhost" be able to tell whether or not it has already been run during the read of /etc/profile or I'd end up with users in a vicious loop bouncing from one machine to another and never completing the login. To achieve this, I simply create a "check file" which the routine looks for at start up. If it finds this "check file" in the user's \$HOME directory it knows it is on the server where the login should complete by removing the "check file" and skipping running itself again.

[PAGE 18. /etc/loginhost]

That's it. A tour of Distributed Computing and what it can do **TO** you, or **FOR** you. As System Administrators we can make or break an installation in this environment, however, with any luck and a lot of reading and preparation you will be successful.

One last caveat, about Client / Server.

If you suddenly find yourself staring a large problem in the face. Don't kid yourself into thinking "There must be someone else that has had this happen to them?" and wait to resolve it while you wait for other "more DCE experienced people to call back". Just count on yourself and follow what makes most sense. I have had times where all the phone calls in the world (literally) ranging from Atlanta, GA. to Australia to Britain and back to Boston, couldn't resolve my situation and it took 36 hours straight to find out. During those times, I either buckled down and went into "cafein cooled debug mode from hell" or just plain aborted the mission and redesigned a system that would work. The truth of the matter is: "sometimes your the Louisville Slugger baby, sometimes your the ball." - Batter Up!

#6012
**Meeting ANSI Standards for
Software Configuration Management**
James Lofink
Operations Control Systems
560 San Antonio Road, Suite 106
Palo Alto, California 94306
(415) 493-4122

No company assigns a single person to design and build a product. Instead, there are teams of engineers and technicians, each responsible for a part. In order for a car, for example, to successfully come together, it is essential for the different teams to communicate their changes to the each other. If the groups isolate themselves, a car could be built with wheels that do not fit, or with only the ability to drive in circles. Once the car is successfully built, it is critical to know all of the parts that make up that model in order for it to be reproduced. Putting an older part into the car, forgetting a part, or using a defective part can result in the customer receiving a car that needs work or, worse, must be recalled.

Communicating and “controlling the changes” that are introduced into new models and insuring the appropriate parts (configuration management) go into the production of a car are critical factors in the production of a quality vehicle. Whether you’re building a nut, washer, and bolt assembly or a jet airplane, the same type of change control and configuration management factors apply whenever more than one person is working on the project.

Change control and configuration management are well accepted practices in the manufacturing industry. Early acceptance in the manufacturing industry can be seen through standards put together by the American Society of Mechanical Engineers (ASME) and military standards initially designed by the Air Force. These same principles apply to the development, maintenance, and distribution of software. The main difference is that software is easier to change. It is much easier for a developer to accidentally compile the wrong source file into an application than it is for someone to put the wrong tire on a car. As a result, in order to insure the quality of the application, the controls on software development are even more critical than in manufacturing.

Your company relies on the applications built and/or maintained by your development team to stay in business. These applications are “mission critical” and controls to safeguard them are extremely important. Safeguards are also important for existing applications because seventy percent (70%) of production

updates are changes or maintenance updates to existing applications. Many of these updates will contain errors that remain unnoticed until after users are impacted. The bottom line is that an organization loses time, productivity, and money when an application fails in a production environment.

Effective change management helps you manage the process of change while maintaining the integrity of your application development and production environments.

ANSI Standards for Configuration Management

In his book Software Configuration Management, H. Ronald Berlack describes Software Configuration Management as “a discipline that governs the identification, control, status accounting, and auditing of a given entity, such as a software program or system, and the components that make up that entity.” At its core, Software Configuration Management (SCM) insures that the right code is in the right place at the right time. Configuration Management and Change Control are used to some degree whenever more than one person makes changes to the same application. These controls can be as simple as hand written documents that detail what files make up a release and track program changes, or as comprehensive as an automated tool that enforces configuration management, change control and testing disciplines, and documents and maintains an audit trail of all changes. The advantage to an automated tool is that it enforces the processes and controls and is less likely to be circumvented.

The Institute of Electrical and Electronics Engineers, Inc. published the ANSI/IEEE Std 828-1983, IEEE Standard for Software Configuration Management Plans. Many standards, however, are overlooked or not adopted because companies are focused on increasing profits and they view standards as an added cost. It is important to recognize that the quality of a company's production software can directly affect the bottom line. Many organizations will measure lost revenue due to production application downtime in terms of thousands of dollars per minute. This is in addition to the costs for a developer to debug and fix the problems. An isolated change may seem harmless, but the ripple effect on interrelated systems can often be devastating. As the volume of code in a shop becomes greater, the effects of the changes can become staggering. Therefore, configuration management and change control procedures are critical to insuring the quality and integrity of software assets.

ANSI/IEEE Std 828-1983 provides a framework for successfully implementing Software Configuration Management disciplines within an organization and gaining some insight into the issues of Configuration Management. While some of the points in this standard may not apply to your organization, it provides a

starting point for implementing configuration management. The degree of commitment an organization has to configuration management directly correlates to the effectiveness of the configuration management processes and, thereby, the integrity and quality of their production applications.

Steps Towards Implementing Controls

Implementing software configuration management is centered on addressing these six disciplines:

- Configuration Identification
- Baseline Management
- Change and Library Control
- Status Accounting
- Reviews and Audits
- Release Processing

The method and degree to which each one of these disciplines is implemented depends on the organization's environment and the type of application that is managed.

Configuration Identification - The first step is to determine (at the highest organizational level) to which applications to apply the configuration management disciplines (i.e. payroll, manufacturing, order entry). In order to apply these disciplines to an application, the configuration management procedures must identify and track the parts that make up the application. These "parts" can include:

- Source Code
- Executable Code
- User Documentation
- Requirements and Design Specifications
- Management Plans
- Test Data and Test Generation Procedures
- Databases
- Maintenance Documentation

Baseline Management - Once the pieces of an application have been identified, one important goal of configuration management is to be able to reproduce the application at any given point in time. A baseline represents the application at a particular point in time. An application manager establishes a baseline by assigning a unique identifier to each of the parts which make up the application at that point in time. This marks and protects all of the parts in the application,

so that the application or any part of the application can be restored to that baseline any time in the future. The configuration management process uses the baseline concept to group related parts of the application (source, executable, documentation), distinguish between different releases of the software, and track changes to the application.

Change and Library Control - A "Library" is a controlled collection of the parts (source, executable, documentation, etc.) of an application. They provide the means for identifying and labeling the baselines (releases) of an application and for controlling the introduction of changes into an application. The library provides a central point of control for changes to the production application. There are several types of libraries: the dynamic (development) library which is used by developers to do their work; the controlled (master) library which is used for managing the current baseline of the application and contains the production copy of each part (entry into the master library is a controlled process); and the static library (software repository) which contains various versions of the application that have been released for general use. The number and type of libraries is dependent on the amount of control that is placed on an application.

Change control is the methodology that governs how, when, where, and by whom changes are made to the baseline of an application. The methodology describes the cycle (process) a part must go through to introduce change into the application. The methodology details the movement of the part between the different libraries. For example, a cycle typically begins with a developer checking out a file from the master library to the development library. After the developer has made his changes, managerial approval may be required before the file is moved to a test library (a form of a dynamic library). Most cycles end by moving the changed file back into the master library.

Status Accounting - Status accounting refers to the ability to determine and communicate the state of the application or any of its parts at any given point in time. In any project, communication between the team members is essential to the success of the project. Status accounting provides this communication through reports and inquiries that allow someone to review status information such as who last checked out a file, when it was last checked out, where it currently resides, the number of change cycles since the last baseline, and the last process performed on the file.

Reviews and Audits - Changes to applications are a normal part of daily data processing activities. While these changes are necessary, so is the need to adhere to good EDP audit standards. Auditing allows for a methodical examination and review of the steps used in introducing changes into production code. The key is to track the progress of work, including enhancements, problem corrections, and

file movements, and create a history of changes and a detailed audit trail. This ensures that every program change is authorized and controlled, making it possible to quickly restore systems when necessary. It allows management to review changes as they take place and verify that the changes are properly authorized.

Release Processing - The first five disciplines are used in introducing changes into an application. Once the changes have been made and a formal baseline is created, the final step is to distribute that application (make it available for production/users). Release processing guidelines manage and control the process of delivering the product to the production environment/user.

Automated Configuration Management Tools

As mentioned before, the greater the commitment and focus applied to the disciplines of configuration management, the higher the quality and integrity of your production applications. While it is possible to implement these disciplines with manual controls, it will take valuable time away from the development staff. If these disciplines are viewed as added work, the probability of success will be limited. There are a number of third-party configuration management tools on the market that can help your team develop and maintain higher quality code in less time.

What do we have for the MPE environment? Nothing comes standard with the operating system, but there are third-party products available that handle check-out/check-in and sophisticated move-to-production procedures.

In the UNIX world, engineers have two standard utilities packages, SCCS and RCS, for source code management (notice the word "source"). With SCCS, the `admin` function is used to set up a library. Engineers access files with the `get` and `delta` commands. Audit trails are maintained with the `prs` function. If the engineers prefer RCS, the `rcs` command is used to define the master library. Check-out and check-in functions are executed with the `co` and `ci` commands. The audit trail is maintained with the `rlog` command.

SCCS and RCS only address the change control and auditing disciplines within configuration management and are, thus, an incomplete solution for production environments. The nature of UNIX production environments dictates a more powerful, comprehensive, multi-system solution. First of all, the administration setup must be easy to use. We must be able to associate more than one file to a class category and then use that category to access a collection of related files. Enforcing procedures must be transparent to the engineer because, as we all know, they have better things to do than spend time learning new commands in

order to gain access to their source code. In addition, generating audit trails must be automatic and comprehensive.

Operations management needs an automatic method to distribute files across the network. Automation ensures that object code is matched to the source code and that the tested version of object code is sent to the correct production platform. There are number of third party tools in the UNIX environment that build on top of the change control offered by SCCS and RCS to offer a more comprehensive configuration management solution.

Benefits

Your company is currently devoting key resources to developing and/or maintaining software applications which will improve performance and productivity throughout the entire corporation. This software is a key component in the infrastructure needed to support your company's day-to-day operations. Applying software configuration management to these software assets will allow you to:

- Increase developer productivity
- Reduce the "bugs" in production code (Decrease application downtime)
- Streamline distribution of the application to remote sites
- Protect against unauthorized changes to critical system functions
- Decrease problems related to mismanagement of code
- Ensure compliance with audit requirements
- Implement an archiving process

Achieving these goals through manual or automated configuration management procedures will increase the quality and integrity of your production environment. A stable production environment enables your organization to maximize its revenue and profit potential.

Additional Information

The ANSI/IEEE Std 1042-1987 is an excellent guide to implementing software configuration management practices that are compatible with ANSI/IEEE Std 828-1983 within your organization. These documents can be obtained from:

The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street
New York, NY 10017

Additional information can also be found in:

**Software Configuration Management by H. Ronald Berlack
(Copyright 1992 by John Wiley & Sons, Inc.)**



Paper No. 6013

The New Computer Graphics Metafile Standard
A general-purpose, international standard
for structured graphics

John C. Gebhardt
InterCAP Graphics Systems, Inc.
116 Defense Highway
Annapolis MD 21401
410-224-2926

Lofton R. Henderson
Henderson Software, Inc.
P.O. Box 4036
Boulder CO 80306
303-442-6572

Computer-generated graphics are being increasingly called on to efficiently convey much of the complex information which fuels today's fast-paced global economy. The new Computer Graphics Metafile (CGM) standard promises to deliver a high level of performance for storage, management and exchange for today's and tomorrow's strategic graphics applications.

INTRODUCTION

Well-designed graphics are powerful communications tools. Computer-generated graphics are being increasingly called on to efficiently transmit much of the complex information which fuels today's fast-paced global economy. Until now, manufacturers, customers, strategic partners, vendors and suppliers have had to rely on multiple international, proprietary, and *de facto* industry standards to store and exchange today's wide spectrum of graphics data.

All indications are that CGM will prove to be the "industrial-strength", open graphics standard which will enable complex structured picture exchange on a global basis. The CGM standard, together with the

The New Computer Graphics Metafile Standard
6013-1

Standard Generalized Markup Language (ISO 8879, SGML) standard, will allow truly robust, standards-based implementations of systems for authoring, management and publishing of complex compound documents for the first time. Such truly open systems can be virtually vendor-neutral with respect to both hardware and software. These new standards-based systems will also be able to support the fast-growing requirement for authoring, delivery, and maintenance of complex, interactive, electronic documents using only standardized data structures.

A BRIEF HISTORY OF CGM

A standard for a general purpose CGM standard was originally published in 1986 by the American National Standards Institute (ANSI) as ANSI X3.122-1986. The same standard was published a year later by the International Standards Organization (ISO) as ISO 8632/1987. To make it easier to manage changes, ANSI X3.122-1986 was replaced by ANSI/ISO 8632:1987 in 1991. This standard defined a graphical interchange format for two dimensional pictures. Even before ISO 8632 was published, work was underway on a series of amendments designed to harmonize CGM with other graphics, application, and electronic technical publishing standards, and to enhance CGM's graphical expressive power. The amendments were driven in large part by the Computer-aided Acquisition and Logistic Support (CALS) program's requirements and the need to provide a standard for exchanging the electronic pictures produced by a new generation of illustrating applications. CALS representatives participated directly on the ANSI and ISO committees responsible for developing the CGM standard.

Instead of publishing a series of extensive amendments, the amendments were incorporated into a new CGM standard and published in September 1992, as ISO/IEC 8632:1992. CGM:1987 was canceled and is no longer the computer graphics metafile standard, although the Version 1 metafile defined in CGM:1992 is virtually the same as an old CGM:1987 metafile. The new standard is being adopted by ANSI, as ANSI/ISO 8632:1992.

CGM IN THE COMMERCIAL AEROSPACE INDUSTRY

Technical information in the commercial aerospace industry has evolved over many years, is growing at an accelerating pace, and must be shared globally up and down a complex "food chain" of strategic partners, vendors, customers and government agencies. Much of this technical

information consists of complex graphics such as the exploded view of a turbine engine shown in the Figure.

The Air Transport Association (ATA) has been developing standards for the digital exchange of technical manuals for the past several years. The Graphics Working Group published Version 1 of their Graphics Exchange Specification (GREXCHANGE V1) as part of the ATA Specification 100 Revision 30. GREXCHANGE references the Tagged Image File Format (TIFF) with CCITT Group 4 (Consultative Committee on International Telegraphy and Telephony) compression for binary raster images and ISO/IEC 8632:1987 (the "old" CGM standard) for vector graphics. An ATA-specific file header is used to exchange customer, document, and revision data related to the illustrations.

Version 2 of GREXCHANGE is now complete and is being validated by ATA member organizations beginning this past summer. It references TIFF and ISO/IEC 8632:1992 but encourages the use of the new CGM standard for both raster and vector data. ATA hopes that, over time, CGM will be used to exchange all graphics and that the use of TIFF can be largely phased out. The CALS CGM Application Profile (MIL-D-28003A) is also referenced by GREXCHANGE V2. In general, the CALS and ATA constraints imposed on the use of CGM are very similar. However, the ATA relaxes some constraints they feel are too severe and imposes others to align the use of CGM with their current use of TIFF. For example, external symbol libraries are not allowed in CALS metafiles but are allowed in ATA metafiles. GREXCHANGE V2 specifies that external symbol libraries are to be metafiles themselves, with a symbol corresponding to a picture in the metafile. For compatibility with TIFF files, the compression algorithm for bitonal tiles is restricted to CCITT Group 4 and the CALS-imposed tile size restrictions have been removed.

The ATA Graphics Task Group is also developing a specification for exchanging "intelligent graphics" and they intend to use the new CGM standard. Intelligent graphics, as defined by the ATA, are graphics which contain application-specific information in addition to the information necessary to render a visual image of the graphic. For example, an intelligent version of the picture of the turbine shown in the Figure might be logically structured into "graphical objects" such as the compressor, diffuser, etc. Information about each object and links to other objects could then be exchanged together with the pictures of the objects. The information stored about the various graphical objects from which exploded views, decision and wiring diagrams, system schematics, and navigation charts are constructed could be used directly by "smart" applications to navigate between and within illustrations and associated

textual documents or to extract data from a data base for digital display and further analysis.

The ATA will use these standardized intelligent graphics structures to implement interactive electronic technical manual applications, such as intelligent portable maintenance aids which seamlessly integrate technical information in the form of text and graphics supplied by organizations up and down the industry's "food chain".

The wiring diagrams which describe the myriad of electrical interconnections on a modern commercial aircraft are excellent examples of the type of graphics-intensive documentation which is tailor-made for CGM:1992 with the new Application Structuring extensions.

If wiring diagrams were exchanged using CGM:1992, all interconnections, wire properties, and component attributes could potentially be managed together with the diagrams themselves, and a common symbol library could be shared among all senders and receivers. Files would be very compact and the diagrams for components such as engines could be merged electronically with the airframe diagrams which could be merged electronically with the airline-specific diagrams. Such a system could save considerable time and effort now spent redrawing diagrams and revising interconnection lists at various levels of the supplier food chain. But the major benefit would be the increased aircraft availability which should result from faster and more reliable maintenance when mechanics have access to diagnostic tools driven by intelligent diagrams in their portable maintenance aids.

CGM IN THE CALS PROGRAM

The CALS initiative has long recognized CGM as the preferred format for delivery and interchange of technical illustrations. MIL-HDBK-CALS and MIL-STD-1840 identified CGM as the preferred format for such content in electronic delivery of technical documents. MIL-D-28003, completed in 1988, specified how CGM:1987 was to be used to deliver CALS electronic documentation deliverables.

MIL-D-28003 specified the subset of the standard to be used in CALS-compliant metafiles, and provided some conformance requirements for generators and interpreters of CALS metafiles. Every conforming MIL-D-28003 metafile was a legal metafile according to the CGM:1987 standard but the converse was not necessarily true.

In parallel with the ANSI and ISO work to write a new CGM standard, MIL-D-28003 was also revised. In November 1991, when the work on CGM:1992 was technically complete but still nearly a year from publication, CALS announced the first release of MIL-D-28003A. It incorporated many of the new capabilities in CGM:1992, and more precise conformance specifications. MIL-D-28003A is a profile of CGM:1992. In general, metafiles which were CALS-compliant under MIL-D-28003 can easily be made compliant with MIL-D-28003A by identifying the metafile as a Version 1 metafile in the Metafile Version element and by updating the Metafile Description as specified in MIL-D-28003A.

The changes between MIL-D-28003A and MIL-D-28003 include the following:

- All references to "draft quality" have been eliminated.
- All metafile versions (1,2,3) are supported with certain restrictions.
- Three types (monochrome, gray scale, and color) are defined
- Color metafiles are now fully specified.
- The metafile description references MIL-D-28003A.
- Additional fonts may be used.

NEW CAPABILITIES OF CGM:1992

Versions

The new CGM standard defines three versions of graphic metafiles. The versions provide implementors the opportunity to build CGM exchange facilities which conform to the standard but do not support all of the rich functionality if it is not appropriate to their application area. Versions also allow implementations to be upward compatible with the new standard, albeit only at the Version 1 level. The capabilities of the three current (and one proposed) versions of CGM-1992 are described in the Table.

The Version 1 metafile is a basic drawing and picture interchange format, usable across many application areas, and not particularly tuned to the application requirements of any single area. It has been widely implemented (a recent report to by Bruce Garner, CALS Test Network analyst claimed on the order of 350 software products using CGM).

A Version 1 metafile has a simple structure. It consists of a Metafile Descriptor (header), followed by one or more logically independent and randomly accessible pictures. The Version 1 metafile definition includes about 90 elements (an element is an individual function or entity, such as BEGIN METAFILE or POLYLINE).

The graphical content of Version 1 metafiles is defined by 19 "Graphical Primitive" elements described in the Table.

Version 1 metafiles are essentially the same as the metafiles of the old standard. In the 5 years since CGM:1987 was published, there have been over 100 requests for clarification and reports of errors. Most of these were trivial and editorial in nature. A smaller number clarified ambiguities or corrected mistakes in the document, which could potentially affect an implementation. All of these editorial errors, clarifications, and inconsistencies are fixed in the new 1992 standard.

Fortunately, there were few significant corrections. Some examples follow (for more information, see Appendix E of The CGM Handbook described in the Afterword):

Clarifications:

- Effect of zero-width lines
- Which C0 (Control) characters may appear in text strings
- Storage order of colors in Cell Array
- Partition alignment in binary metafiles

Changes:

- Rules for alternate character sets in non-graphical text were added.
- 1-point polylines, 2-point polygons, etc., were made illegal.
- Structured Data Record (SDR) coding technique was recommended for all type D parameters.

Version 2 metafiles

Version 2 metafiles may contain some 30 additional elements. (all Version 1 metafile elements are allowed in Version 2 metafiles). In addition, Version 2 metafiles relax some of the Version 1 metafile rules on the location of certain elements.

The most significant new capability of Version 2 is the graphical segment. In Version 1 metafiles, the picture definition is simply a sequential stream of individual elements. A segment is a group of primitives which is saved once and named, and then may be repeatedly instanced into the picture with different scaling, attributes, etc. Segments are a very useful efficiency mechanism that could be used effectively in schematic applications. A common library of symbols could be stored once in the metafile and used repeatedly throughout the various pictures in the metafile. Other useful features of Version 2 metafiles are listed in the Table.

Version 3 metafiles

Version 3 metafiles represent a major increase in graphical expressive power over Version 1 and Version 2 metafiles. Version 3 metafiles are a high-fidelity exchange format for 2D graphics in virtually any field including technical illustrating, graphic arts, engineering and architectural drawing, cartography, and many others.

Version 3 metafiles contain some 40 new elements in addition to those allowed for Version 2 metafiles. These new capabilities will enable the CGM to be used much more effectively to store and exchange structured graphics data. A few of the more important new capabilities are:

- The capability to represent compressed tiled images. The CGM can directly embed tiled raster content in a picture, fully integrated with vector graphics. All of the individual raster image capabilities of MIL-R-28002, ODA part 7 Tiling, and TIFF, are supported; JPEG support is pending.
- The capability to define external symbol libraries and reference these symbols from within a metafile are now available. Symbols can be transformed when they are instanced.

Much of the Version 3 metafile functionality was driven by the requirements of electronic publishing applications in CALS and CALS-like environments. In addition, the new capabilities were designed to maximize the interoperability and compatibility of Version 3 metafiles with existing standards.

For example, the compressed tiled raster facility was taken from and harmonized with the Tiled Raster Image Format (TRIF), the CALS standard for raster data, MIL-R-28002, and the Open Document Architecture (ODA) part 7 tiling standard. The JPEG support is being included through the ISO Graphical Registration process now that it is close to becoming an ISO standard. The new color models and color calibration capability are taken directly from the ODA Colour Addendum. Many of the new graphics rendering capabilities were harmonized with standard and proprietary page description language formats. The curve definitions were designed to be directly compatible with industry standard and proprietary formats for engineering, graphic arts, and page description. Finally, the text font and glyph mapping capabilities are based directly on the ISO 9541 font standard.

THE FUTURE OF CGM

All indications are that CGM:1992 will prove to be the "industrial-strength", open graphics standard which has been sorely needed to enable complex structured picture exchange on a global basis. Government and Industry groups throughout the world such as the ATA and CALS are viewing CGM:1992 as the ideal companion standard to SGML for integrated text and graphics authoring, management and publishing systems. Software vendors are hard at work building the infrastructure required to support these new initiatives. A wave of new commercial interface tool kits and end-user applications for creating, editing, exchanging, and managing CGM:1992 graphics data are appearing for a wide variety of computer platforms.

Meanwhile, ISO is currently processing two amendments to the new standard and is considering a third which promise to accelerate the acceptance of CGM.

Rules for Profiles

ISO recognized that uncontrolled proliferation of CGM profiles such as MIL-D-28003A could pose a situation nearly as bad as the lack of a graphical interchange standard. Without some guidance, it was feared that competing and confusing "flavors" of CGM would arise which could not interoperate. It was also recognized that in many cases two communities might be able to use identical or nearly identical profiles. Furthermore, five years of experience with "open interchange", and with the writing and evaluation of profiles, had led to considerable practical

experience within the CGM community as to what a profile minimally must specify in order to assure interoperability for its target community.

Accordingly the first amendment to CGM:1992 has been initiated to specify how profiles should be written. Amendment 1 adds no new graphical functionality to CGM. It specifies:

- Rules for Profiles of CGM, for those considering writing a profile,
- Definition of conforming generators and conforming interpreters, in terms of profiles, and
- A Model Profile. The Model Profile is a template or pattern for those writing profiles. It also a complete and valid profile. When Amendment 1 is approved, CGM:1992 will include a complete and testable specification for certifying generators and interpreters.

Earlier this year The National Institute of Science and Technology (NIST) started a certification service for CGM generators and metafiles. This program (described in another article in this journal) will be a very important factor in the success of electronic document interchange using CGM, especially within the CALS community.

Application Structures

Compound electronic document applications are moving in the direction of graphics and illustrations which are closely integrated with text as well as other content types (text, review/markup, live data from data bases, video, audio, other graphics, etc.). It has become obvious that a reader of an electronic document (for example, a parts catalog) should be able to point at a picture of a part and obtain other information about it. The requirement to "touch" an arbitrarily small part of a picture is common to a variety of applications including electronic review, configuration control, interactive training, electronic interactive maintenance aids, electronic documentation navigation, etc.

The functional requirement for a second amendment to CGM:1992 to address this need for application-related structuring of metafiles was completed in mid-1992. A work project for an amendment to add such elements to CGM was accepted by ISO in October 1992. Completion is scheduled for mid-1994.

The requirements of the CALS initiative for electronic review of graphics in compound documents and the requirements for "intelligent graphics"

as defined by the ATA were instrumental in creating a sense of urgency for this new capability.

3D METAFILES

There has been interest in a 3D metafile format within ISO for many years. One project was initiated in 1986, then killed for lack of well defined direction. As proprietary systems for 3D animation have proliferated and solids modeling technology for engineering design has become more widespread, new interest in 3D metafiles has emerged from the technical documentation community. A 3D metafile project is currently in the planning phase.

CONCLUSION

Using CGM:1992, organizations can store, manage, and interchange more sophisticated and complex graphics, at higher quality and fidelity levels, more efficiently, and with the option of fully integrated vector and raster data in a single file, without having to rely on proprietary file formats.

The profiles and conformance amendment, together with operational testing and certification, will provide the possibility for consumers of CGM technology to obtain certified products from their software vendors.

The application structuring amendment will enable CALS Electronic Review of CGM graphics to be a reality within two years, and will open up a wide range of state-of-the art , standards-based, electronic document applications in the CALS and commercial marketplaces.

Afterword

For those interested in more detail, there are three books available on CGM.

CGM and CGI, D.B. Arnold and P.R. Bono, Springer-Verlag, 1988, 279 pp, presents the two named standards in a more readable style than the standards themselves.

Computer Graphics Metafile, L.R. Henderson and A.M. Mumford, Butterworths, 1990, 409 pp, explains technical details, context, case studies, guidelines, and pitfalls, etc. of CGM:1987.

The CGM Handbook, L.R. Henderson and A.M. Mumford, Academic Press, June 1993, approx. 425 pp, is about CGM:1992 topics and the differences between the new versions of the standard.

About the Authors:

John C. Gebhardt is Executive Vice President, Chief Technical Officer, and Founder of InterCAP Graphics Systems, Inc. and has worked in the field of computer graphics for 30 years. InterCAP develops software products for intelligent graphics authoring, exchange, and publishing. Dr. Gebhardt is currently the Vice Chair of the ANSI CGM task group, a member of the US delegation to the ISO working group on CGM and is the document editor for the recently approved Application Structuring amendment. He serves as an advisor to the Air Transport Association's Graphics Task Group, helping them develop a profile for the use of CGM:1992 for exchanging intelligent graphics.

Lofton Henderson is founder and president of Henderson Software, Inc., a company specializing in CGM technology, tool kits, engineering, and training. He has worked on the ANSI and ISO committees responsible for CGM for over 10 years. He was the document editor of the CGM:1987 standard; leader of the ANSI CGM task group and head of the US delegations to ISO CGM working group meetings; and document editor of CGM:1992. Mr. Henderson has made substantial contributions to the definition of CALS MIL-D-28003 and MIL-D-28003A, has worked with other application communities on similar profile definitions, and continues to participate in the CALS Electronic Review initiative.

Descriptive Capabilities of CGM:1992 Versions	Version			
	1**	2	3	4*
Straight lines and segmented straight lines	X	X	X	X
Filled polygons and polygon sets	X	X	X	X
Rectangles and circles	X	X	X	X
Markers of several predefined styles	X	X	X	X
Text strings	X	X	X	X
Circular arcs	X	X	X	X
Elliptical arcs	X	X	X	X
Circular and elliptical pie and chord sectors	X	X	X	X
Bitmap and RLC raster images	X	X	X	X
Color attributes (RGB and indexed color)	X	X	X	X
Dash style and line width for lines and edges	X	X	X	X
Area fills, pre- and user-defined patterns	X	X	X	X
Text font, character set, placement, alignment, extent, orientation, etc.	X	X	X	X
Segments which can be transformed and instantiated		X	X	X
Closed figures consisting of multiple elements, end-to-end		X	X	X
Precise clipping and cropping		X	X	X
Definable attribute bundles		X	X	X
Save/restore graphical context		X	X	X
Tiled raster images: variety of compression schemes include MIL-R-28002			X	X
Advanced curve definitions including cubic Bezier curves, conic arcs, and B-splines			X	X
External symbol libraries			X	X
Text along arbitrary paths			X	X
New text control attributes			X	X
Definable cross hatching			X	X
Advanced areas fills including geometric, wallpaper, and interpolated			X	X
New color models including CMYK, CIELUV, and CIELAB			X	X
New line attributes including line caps, joins, miter limit, definable dash patterns			X	X
*Application structures				X
*Application structure attributes				X
*Picture directory				X
*Application structure directory				X

*Amendment 2, Approved. Published by 12/94

** Equivalent to previous (1987) CGM standard

HP-UX 9.X Operating System Update

Paul Perlmutter

Hewlett-Packard Company
Software Services and Technology Division
Fort Collins, CO
(303) 229-2549

The focus of this paper is on the key changes in HP-UX 9.0 running on Series 300, 400, and 700 workstations. We will also look at the incremental changes in the 9.0X releases since 9.0. As you probably know, HP-UX also runs on our Series 800 commercial server systems and we will look at how the 800's fit into the big picture, but primarily our focus will be workstations. We have aimed this paper somewhat broadly across all of HP-UX. It is a "10,000 foot" view of the workstation business.

HP-UX 9.0 OVERVIEW

There are 3 key contributions in HP-UX 9.0 for our customers:

- Performance
- Ease of use
- An open systems environment

In all three areas HP-UX/9.0 has made significant enhancements.

HP-UX 9.0 together with the PA-RISC 7100 CPU delivers a level of performance unmatched in the industry. In addition to supporting the PA-RISC 7100 CPU, the operating system itself has undergone major tuning with HP-UX 9.0.

HP-UX 9.0 continues to build on its "Award-Winning Ease of Use" position with major productivity enhancements for both the computer novice and the expert, the user and the system administrator.

HP has been the leader in open systems and open systems standards for over 10 years, long before it became the popular force that it is today. HP helped create and chair a number of open systems standards bodies. This open systems position insures a truly open systems environment for our customers worldwide, both today and into the future.

HP-UX provides leadership in all three areas.

The latest system software releases of HP-UX are: 9.03 on the Series 300/400, 9.05 on the Series 700 and 9.04 on the series 800. We will refer to these releases more generally as HP-UX/9.0 or just Release 9.0.

As the next major release following HP-UX 8.0, Release 9 supports a number of additional hardware platforms and includes a number of bug fixes. It also brings together incremental features from HP-UX releases, notably 8.07 on the Series 700 and makes those features available on all HP-UX platforms.

As of 9.0, the following features became available on the Series 300/400 and the Series 800 in addition to the Series 700:

The NFS automounter allows dynamic mounting of NFS mounts based upon user accesses. It allows you to more easily administer a network of systems to have a common view of NFS file space.

SIGWINCH support has been added to the kernel and shells as well as to the HP-UX commands: elm, more, stty, vi, rlogin, telnet and on. This supports dynamic resizing of windows.

A filter has been provided in the postscript LP model for printing ASCII files on a postscript printer.

The swapinfo command is now supported to report information about device and file system swap utilization.

Two new X clients, Imageview and Vuepad have been added. Vuepad is a text editor built from the Motif text widget. It supports standard cut and paste, word wrap, drag-and-drop and printing. With 9.0 and VUE 3.0, Vuepad is capable of handling 16-bit languages.

Imageview is a bit-mapped image file viewer. For 9.0 it has been enhanced to have these new features:

- ◆ Access to a wide range of bit-mapped image files
- ◆ An Enhanced user interface
- ◆ New hypertext help windows with sample images, and
- ◆ New sample image files

With the advent of release 9, Series 300/400 and Series 700 machines can now perform a cold network install from a netdist server onto a disk on the new machine.

HP-UX 9.0 SYSTEM SOFTWARE CHALLENGES

The key system software challenges are performance, ease of use and truly open systems while preserving customer's investments.

Total system performance is critical for our customers. From this perspective, the system must deliver superior hardware AND software performance. With HP-UX 9.0 significant software performance enhancements are achieved in areas such as the human interface, the I/O subsystem and the Virtual Memory (VM) system, and new compiler optimizations.

Another key challenge in the UNIX market place is making the UNIX operating system easy to use for everyone. HP-UX 9.0 provides ease of use enhancements in both the user interface and the system administration interface.

Open Systems? Here the challenge is to give customers the flexibility to choose the right solutions to run their business effectively and competitively. To accomplish this, HP's operating system provides customers with portability, interoperability and scalability.

Finally, HP-UX 9.0 must establish itself as the market leader in these areas *while* preserving the customer's investment. This means that customers are both protected and can also take advantage of the best software technology available.

INVESTMENT PROTECTION

As usual, HP has taken steps to make sure that customers can take advantage of this new software technology without losing their existing investment. Investment protection is a broad topic that includes compatibility, interoperability, migration paths, support lifetimes, and more. Investment protection means providing lasting value for your existing hardware and software.

HP-UX 9.0 is binary compatible with previous releases on the same platform. Our goal is that executables built on prior releases will continue to run under release 9.

- ♦ For the Series 300/400, 9.0 is binary compatible with HP-UX 8.0.
- ♦ For the Series 700, 9.0 is binary compatible with release 8.05 and 8.07.
- ♦ For the Series 800, 9.0 is compatible with 8.0, 8.02, and 8.06.

HP-UX 9.0 is binary compatibility across PA-RISC platforms. By this we mean applications that use features common to both the Series 700 and Series 800 can be built or run on either 9.03 on the Series 700 or 9.04 on the Series 800. This allows many VABs or users to build one set of executables that run across the PA-RISC line. The Series 800 includes some machines that implement PA-RISC version 1.0 as well as others that implement PA-RISC 1.1. All 700 machines use PA-RISC 1.1. Code that is compiled for the 1.0 instruction set can be run on 700s or all 800s, while code that is compiled for 1.1 can run on 700s or some 800s. HP-UX supports compiling 1.0 or 1.1 code on either 700s or 800s, with the default being 1.0 on 800s and 1.1 on 700s.

Our Series 300/400 systems are based on the Motorola M68XXX processor family and is architecturally different from our PA-RISC architecture, and hence the instruction sets are different. Thus, we don't have binary compatibility.

However we have worked hard in getting close to full source code compatibility. (We're proud of this effort, although we know that in best of all worlds we would want to go even further and develop an architectural neutral distribution and execution format.)

Besides compatibility, another form of investment protection is the support for upgrades. HP-UX 9.0 includes support for PA-RISC hardware upgrades for the 425s and 425t. These offer a path for 68K-based Domain or HP-UX customers to upgrade their CPUs to PA-RISC running HP-UX 9.0.

Finally, support for Series 300/400 HP-UX will continue through the year 2000. This includes bug fixes and selected new peripherals, customer driven enhancements, performance tuning, and support for interoperability.

HP-UX 9.0 SOFTWARE PERFORMANCE

Exciting news in software performance is that HP-UX 9.0 enhances the performance of many existing applications. Even greater performance boosts are realized when applications are recompiled.

In the kernel there are four major areas of performance enhancements for the Series 700 and 300/400 plus a few other goodies.

First, in release 9 the virtual memory system has been tuned. Much of this tuning reflects the changing application loads and increased speeds of modern systems. Heuristics that worked fine on VAX 11/750s running small programs that could fit into 64K segments need updating. CPU speeds increase a hundred fold and applications grow to 10 megabyte working sets. Processors have gotten faster, memories are larger, and the ratios between CPU and I/O speeds have changed. The virtual memory system in HP-UX 9.0 has also been tuned to improve performance under stress and to favor interactive processes over nice'd jobs in memory as well as CPU allocation.

Second, file system and I/O performance have also been tuned in 9.0. More aggressive file system read ahead and multi-stream I/O tuning help to keep faster CPUs fed with data. At 9.01 SCSI request merging was added to the Series 700 SCSI subsystem providing even more I/O performance gains and software disk striping was ported from the Series 700 to the Series 300/400.

The third major enhancement is the introduction of a dynamic buffer cache in HP-UX 9.0 on the Series 700 and Series 300/400. The dynamic buffer cache allows memory to be moved back and forth between use by the VM system and use by the buffer cache. HP-UX manages the size of the dynamic buffer cache based upon the relative demand for I/O buffers and VM pages. The dynamic buffer cache improves performance in two ways:

1. By eliminating fixed limits on the size of the buffer cache it improves I/O throughput when VM contention is light and improves VM performance when I/O needs are low.
2. It reduces the need for hand tuning of the buffer cache size and makes easier to get optimal performance. This is also a significant ease-of-use advantage for administrators.

The dynamic buffer cache is enabled by default when 9.0 is installed or when you update to 9.0. The dynamic can be overridden by explicitly setting either the NBUF or BUFPAGES parameters, or both, in the dfile.

The fourth major performance enhancement in 9.0 is the introduction of memory mapped files, or MMFs. MMFs and the msem_lock synchronization primitives are available as of 9.0 on the Series 700 and the Series 300/400. Memory mapped files reduce the system call overhead associated with making read and write system calls and avoid the copying of file system data between user space and the buffer cache.

Supporting the MMAP interface allows applications to be restructured to reference file data directly in memory and handle concurrence using the msem_lock routines. This can result in improved application efficiency. Support for MMFs also allows the importation of code that depends on these interfaces. For many customers that may be the biggest advantage of MMFs.

HP-UX has a long history of performance improvements in the filesystem. Many, but not all of these apply when accessing files using memory mapping. If your application makes heavy use of an enhancement such as aggressive read ahead that does not yet apply to MMFs, you may well see faster performance through the file system. If your application makes multiple passes through its data and you have enough RAM so that

the file stays in memory or if the application makes random accesses to small items in the file you will likely find MMFs to be faster.

Likewise if your system is CPU bound instead of I/O bound, MMFs will likely improve performance. Of course, whenever we speed up file system I/O, if that enhancement is not applicable to MMFs the relative speed of the file system goes up. As a long-term trend, we expect memory mapped files to improve performance for more and more applications. We are currently investigating how to make more of the file system optimizations apply to memory mapped files.

In addition to this, HP-UX 9.0 includes supporting EXEC_MAGIC programs of up to 1.9Gb on the 700, fast symlinks where short paths are stored directly in the inode, tuning of the curses library and tty line discipline code, and supporting the getpriority and setpriority system calls along with the renice command. The 9.03 release on the Series 300/400 sports a leaner, meaner kernel that uses less wired down memory and some special enhancements to the SAM product that really enhance performance.

Despite what some might like you to believe, not all the performance enhancements in 9.0 are due to hard working O/S engineers. The significant news here is that the new HP-UX 9.0 compilers and linker can show application performance improvements of 10 to 30% on the S700 workstations for CPU-intensive applications. Furthermore, the Series 700 linker itself has been enhanced to speed up link times.

EASE OF USE

HP-UX 9.0 focuses on two key Ease of Use areas: the user environment and system administration. In the USER ENVIRONMENT the key contributions to Ease of Use are HP VUE 3.0, and the new HP Help Manager.

HP VUE 3.0 is based upon X11R5 and Motif 1.2. VUE 3.0 has been designed to have increased usability, easier configuration, and a new visual design. The front panel includes multi-colored icons, slide-up subpanels, and animated controls. The VUE 3.0 file manager can now use the root window as a desktop. It sports an easier to configure the front panel, user selectable colors and enhanced System Administration Manager (SAM) integration with front panel access to SAM printer administration functions.

The new HP help manager is used for VUE help, but is independent of VUE. It can and is used by other applications. The HP help manager provides context sensitive help and hypertext links. It is based upon SGML and is localizable. VUE 3.0 and the HP Help Manager have been selected as key components of the common desktop portion of the Common Open Software Environment.

With HP-UX 9.0, HP has made significant enhancements in administration to ease the burden of maintaining UNIX-based systems. The System Administration Manager, or SAM, now features a Motif-based GUI for easy interaction and a new task-oriented design so users can easily step through system administration functions.

The distributed update and install facility provides for pulling software products from a central system administration node. As of 9.0, the Series 300/400 as well as the Series 700 can perform a cold network install. In 9.0, the fpkg command is now supported so that administrators and ISVs can package up software for use with update and install. The final component of Ease of Use is Instant Ignition. Instant Ignition is the option where HP preloads system disks with HP-UX and other software. This saves administrators from having to install software and offers a preconfigured system to get you started. With HP-UX 9.0 more layered software products such as the MPower

multimedia package can be ordered preinstalled. Instant Ignition in 9.0 also includes a new boot checklist to give a simplified visual indication of progress and status when HP-UX is booted. This cuts down on confusion and clutter for users.

INTERNATIONALIZATION

The key to being a successful market leader worldwide is to make sure that the specific needs of our international customers are also being addressed. HP has long been a leader in internationalization and localization. Many of these features are not new in 9.0, but they are worth mentioning because of the importance of internationalization and localization for customers worldwide. True ease of use can only be realized when the user deals with the system in the user's native language.

HP-UX supports internationalization using the locale and message catalog mechanisms standardized by X/Open, POSIX, and ANSI C. With the advent of release 9.0, HP-UX supports the XPG4 Worldwide Portability Interface for wide characters. This provides portable support for multibyte character sets. HP-UX 9.0 includes the locale and localedef utilities specified by POSIX.2 for managing locales. With the inclusion of X11R5 comes internationalization support for X programs.

Using these internationalization capabilities, HP has localized HP-UX 9.0 into six different languages: Japanese, German, French, Korean, Traditional Chinese, and Simplified Chinese. This localization support includes HP-UX message catalogs and VUE messages.

VUE help has been localized for Japanese, German, and Korean. The latest release includes a full set of manpages localized for Japanese.

HP-UX 9.0 / OPEN SYSTEMS LEADERSHIP

The primary benefits that customers are looking for in a true Open Systems environment are portability, interoperability, and scalability.

Portability based upon standards means that applications developed on HP-UX systems are easily portable, not only to other HP-UX platforms, but also to competitors' platforms. HP-UX 9.0 continues HP's strong commitment to standards based open systems.

Interoperability allows HP-UX-based systems and a wide variety of other systems to communicate and work together to meet customer needs. Customers are not locked into any vendor's solution. They can take advantage of the best technology available and can take advantage of different platforms' capabilities and tie them together into a heterogeneous network.

Scalability is not always thought of when talking about open systems. We believe it should because it provides the same thing for customers--*choice*. Scalability ensures that applications can run on the broad range of HP-UX systems available, allowing customers to choose the one that is best for their own business.

Let's take a look at the new standards supported under HP-UX 9.0:

The X/Open Soft XPG4 specification;

- POSIX 1003.2, which standardizes the POSIX shells and command set across UNIX platforms to aid in portability among POSIX-compliant platforms;
- The OSF/AES (Application Environment Specification) for increased portability among OSF platforms;
- The X/Open WPI (Worldwide Portability Interface), which provides a standardized interface for writing international programs as specified by XPG4;
- X11R5, which allows for up to 25% interactive performance increase over X11R4, as well as better Internationalization;
- Motif 1.2, which features a drag-and-drop mechanism for moving objects and exchanging data between applications;
- PEX, PHIGS Extensions to X;
- NFS 4.1 features to improve interoperability and which contain the auto-mounter for easier administration, controlled root access over NFS for increased flexibility, and directory and file export for increased security.

Products optionally layered on 9.0 include DCE, PEX, Fortran 90 features and C++ 3.0. DCE is OSF's (Distributed Computing Environment) for distributed homogeneous client-server computing and PEX is the PHIGS Extensions to X. The new Fortran 90 features simplify downsizing from mainframes or minicomputers to client-server platforms like PA-RISC- based workstations and servers. Fortran 90, the new ISO Fortran standard, is a superset of the existing ANSI 77 Fortran standard. C++ 3.0 is based on USL C++ 3.0 and includes support for templates to promote software reuse along with exception handling to provide a convenient mechanism for handling error conditions.

OPEN SCSI

For many of you, the terms "open systems" and interoperability go further than mixing and matching systems. One of the significant advantages of open systems is that it allows customers to mix and match peripherals from multiple vendors. HP-UX 9.0 on the Series 700 makes it easier to mix and match SCSI peripherals. There are three parts of the Series 700 Open SCSI strategy.

First, in HP-UX 9.0 we have improved the supplied drivers. The HP-UX supplied drivers are as robust, general, as possible. The SCSI disk and tape drivers supports a wide variety of peripherals.

Second, if the HP-UX SCSI drivers are not sufficient for a device or your special needs, HP-UX 9.0 introduces a new mechanism, `scsi_ctl`. The `scsi_ctl` mechanism gives user level drivers direct control of devices on a shared SCSI bus. This interface is documented in the `scsi_ctl` manual pages. Using `scsi_ctl` makes it easier to develop and debug drivers in user space. For those that require kernel level drivers there are a set of common SCSI services available to all S700 SCSI kernel device drivers.

Finally, we have made our SCSI drivers fully SCSI II compliant.

SCSI SUBSYSTEM ENHANCEMENTS FOR THE SERIES 700

Numerous enhancements have been made to the Series 700 HP-UX 9.0 SCSI subsystem in order to improve performance, better support our peripherals and give our customers a more robust family of drivers. A quick summary of the enhancements are:

In release 9.03 I/O request merging has been added to the SCSI driver. If the driver detects 2 or more small request to read or write contiguous blocks it will merge these into a single request. This improves performance by reducing per request driver overhead and reducing seek time and rotational delays.

In 9.01 we have added support for Fast/Wide 16-bit differential SCSI interfaces such as those found on the 735 and 755.

Multi-LUN support has been added for devices like some disk arrays that have multiple logical units at the same SCSI target address.

Command queuing has been added to support devices that can make use of command queuing to improve performance.

For our Domain customers who are migrating over to HP-UX, there is now SCSI tape support for QIC 24 devices.

In general the documentation has been improved with better manpages for the SCSI subsystem.

For SoftPC there is now a separate SCSI floppy driver that has been tuned for that use.

Throughout the SCSI subsystem a common mechanism has been used to log SCSI events.

In addition to the S700 SCSI support, HP-UX 9.0 includes support for some other important new hardware capabilities.

HP-UX 9.0 HARDWARE SUPPORT - SERIES 700

Supporting hardware platforms and peripherals have always been a fundamental part of the operating system's job.

First, HP-UX 9.01 supports the PA-RISC 7100 processor and its industry leading performance. The PA-RISC 7100 chip includes a hardware TLB miss handler. HP-UX 9.0 has been modified to take advantage of this feature when it is present without sacrificing performance on other PA-RISC implementations.

Second, the 7100 processor includes facilities to reduce memory traffic in uses like the kernel's copy routine. HP-UX 9.0 takes advantage of these cache hits to speed up the kernel copy routines.

The third PA-RISC 7100 related improvement is enhancing the utilization of block TLB entries. Some previous PA-RISC implementations, notably the chip used in the original 700s included a limited number of block TLBs. These are used to cover large contiguous translations such as the kernel itself and framebuffers. The PA-RISC 7100 processor has more available block TLB entries, and each entry covers both instructions

and data. HP-UX 9.0 has a generalized block TLB allocation mechanism that takes advantage of whatever facilities are present and maximizes the coverage of the block TLB entries.

In addition to supporting faster processors, release 9 on the Series 700 includes hardware support for a number of I/O devices. 9.0 supports CD quality audio hardware, Fast/Wide SCSI as mentioned before, the new CRX48Z graphics subsystem, and for PA-RISC industrial workstations equipped with VME busses, HP-UX 9.0 includes VME services for drivers. Release 9.03 also provides support for the new Series 712 low-end personal workstation.

HP-UX 9.0 HARDWARE SUPPORT - SERIES 300/400

For our Series 300 or Series 400 customers, there have been hardware support changes as well. The PC-101 or AT style keyboard is now an option for the Series 300/400. This option improves usability for customers who are used to the PC keyboard layout. Release 9.03 also includes support for the LaserJet IV and PaintJet XL300 printers as well as new 1GB, 2GB, and 525MB SCSI disk drives.

HP-UX 9.0 NETWORKING

Some of the most common questions and requests we hear from our customers have been "How do I allow root access across NFS on HP-UX" and "Why do I get INVALID CLIENT CREDENTIALS when I export a file system from my HP to another system." Fortunately we can now say: "Upgrade to 9.0" in addition to the previous workarounds.

HP-UX 9.0 implements NFS 4.1 features that have become defacto industry standards. With HP-UX 9.0 you can now export root privilege over NFS to specific clients. It is now possible to export directories in addition to entire file systems and you can do a read-only export in addition to a read-only mount. The INVALID CLIENT CREDENTIAL problem should be solved by the fact that users can now belong to up to 16 groups instead of the previous limit of 8.

As mentioned earlier, the NFS automounter is now available on all HP-UX 9.0 platforms.

In 9.0, there has been a major effort to tune the performance of NFS. This is reflected in the results for the SPEC SFS 1.0 benchmark where both the Series 700 and Series 800 do very well.

The latest release includes an optional product that implements Streams. This is based upon the OSF/1 streams implementation and implements a large subset of V.4 streams.

Along with defect fixes and upgrading services such as gated, it is now possible to dedicate a specific pty to a port on a distributed terminal controller. This makes it much easier to do things like attach serial printers off of DTCs.

HP-UX 9.0 LANGUAGES

As mentioned earlier, the HP-UX 9.0 compilers and linker can give CPU intensive applications improvements of up to 10-30% when recompiled.

A new feature for 9.0 is the introduction of a directory search path for shared libraries. This can solve problems that occur when shared libraries are moved. The search path can either be specified when the application is built or be determined at runtime from

the SHLIB_PATH environment variable. For PA-RISC platforms, the chatr command may be used to modify a program's use of the directory search path facilities.

Also new for 9.0 is profile based optimization. With profile based optimization an instrumented executable is run using sample or actual data sets. The profiling information is then fed back into the process of building a more optimized executable.

A relatively simple enhancement but one that will be of interest to many of you, is the introduction of extended ANSI mode in the ANSI C compiler. Extended ANSI mode is selected with the -Ae option. Extended ANSI mode gives you function prototypes and the rest of ANSI C but does not remove all the HP-UX symbols from the namespace. cc -Ae is the same as cc -Aa -D_HPUX_SOURCE plus it permits dollar signs (\$) to appear in identifier names, and enum declarations can include integral type specifiers. Additional extensions may be added to this option in the future.

HP-UX 9.0 on PA-RISC includes new PA-RISC 1.1 specific math libraries. These libraries have been tuned to take advantage of some of the 1.1 specific instructions. They are both faster and more precise than the 1.0 versions. Included in the 1.1 math library are a number of additional functions, many of which came from the DN10000 math library. The new functions include float versions of many mathematical functions, degree-valued trig functions, some functions recommended by the IEEE standard, and some BSD4.3 functions. Using the PA-RISC 1.1 math libraries has many advantages, but if it is important to get the same results for all operations as previous releases or if your application needs to run on a 1.0 based 800, you should use the PA-RISC 1.0 math library.

In the area of debugging, xdb has been enhanced to support symbolic debugging of shared libraries and to better support debugging in xterm windows.

HP-UX 9.0 CLIENT/SERVER LEADERSHIP

Put HP-UX 9.0 together with PA-RISC and you have one of the key contributions to HP's leadership in the client/server business, scalability.

HP's commitment to standards and open systems along with the power of PA-RISC allows us to offer HP-UX across the broadest compatible UNIX family in the industry. X-terminals that communicate using industry standard protocols and can run compatible clients locally provide graphics capabilities and simplified administration at an entry level price. A common application programming interface gives source code compatibility across the series 300/400, Series 700, and Series 800. A common application binary interface provides binary compatibility across the full spectrum of PA-RISC systems, both 700s and 800s. Interactions in performance and ease of use enhance HP-UX 9.0 across all platforms. Standards-based networking lets HP-UX systems interoperate with each other and other vendors' systems as clients and as servers.

This desktop-to-datacenter advantage allows customers to choose the system that best fits their needs without having to worry about maintaining multiple OS environments. This is a significant benefit to customers that really sets HP-UX apart from the other OS's in the industry.

As you can see, a LOT of work has gone into HP-UX 9.0 to make it the most Powerful, Easy, and Open system in the market today. And remember that HP allows customers to take advantage of all this new software functionality while providing UNMATCHED investment protection for its customers.

7000

Taking the Mystery Out of RTE-A System Generations

Bill Chu
Hewlett-Packard
11000 Wolfe Road
MS 42UN
Cupertino, CA 95014
408-447-5244

Abstract

This paper discusses RTE-A system generation for maximizing features of the operating system. Focus is on the areas of minimizing system table space, ways of increasing base page links, and optimizing the system.

Background and History of RTAGN

RTAGN was introduced at the first release of the RTE-A operating system in 1983. At that time, it was designed to be easy to use, very fast, and to enable quick turn arounds. The generator's philosophy was to generate a new bootable system without having to reload all of the operating system's programs. RTAGN was designed to sweep through the answer file from the beginning to the end without stopping. When the answer file contains an error, RTAGN displays the encountered error and continues. In this way most of the errors are flagged and the system manager can correct all of the errors in the answer file right after the first run. After all the errors are corrected, the generator is run again and the system is immediately bootable without having to reload all the remaining system programs. This was quite an advance for system generation at that time.

RTAGN is a program with four segments written in the Macro language. The generator is supported by its own loader library routine called \$ldrln. The generator has been updated and modified many times during the many releases of RTE-A, but the basic form and function still retain the original concept and design.

Overview

The RTE-A system generation consists of preparing a system generation answer file and running the system generator program RTAGN. The system generation process is shown in diagram 1. The system processor, the interface cards, and the system peripherals must be considered during the configuration planning portion with the aid of worksheets and a sample answer file. After the appropriate worksheets have been filled out, an existing answer file can be edited according to the worksheets to create a new system generation answer file. The system generator program RTAGN is run using the new answer file to create a system file, a snapshot file, and an output map list file. The system and snapshot files are used to install the new operating system.

The system generator RTAGN executes in five functional phases: initialization, system relocation, OS module/driver partition, table generation, and memory allocation. The generator accepts only those commands expected in each phase of execution.

1. Initialization Phase. In this phase, the generator initializes the tables, variables, and parameters used by the operating system.

2. System Relocation Phase. During this phase, the generator relocates specified relocatable modules of the operating system.

3. OS Modules/Driver Partition Phase. In this phase, system modules and I/O drivers are relocated and grouped into physical partitions. Partitioning allows more system table space and the use of more system code, including large drivers.

4. Table Generation Phase. In this phase, the generator allocates space for various system tables and configures the tables.

5. Memory Allocation Phase. During this phase, the generator allocates space for ID segments, system available memory, system memory block, the maximum number of classes, resource numbers, memory descriptors, system common, and error message relocation. The number of shared programs, number of concurrent users, swapping limit, timeslice value, and default libraries to be searched are also defined.

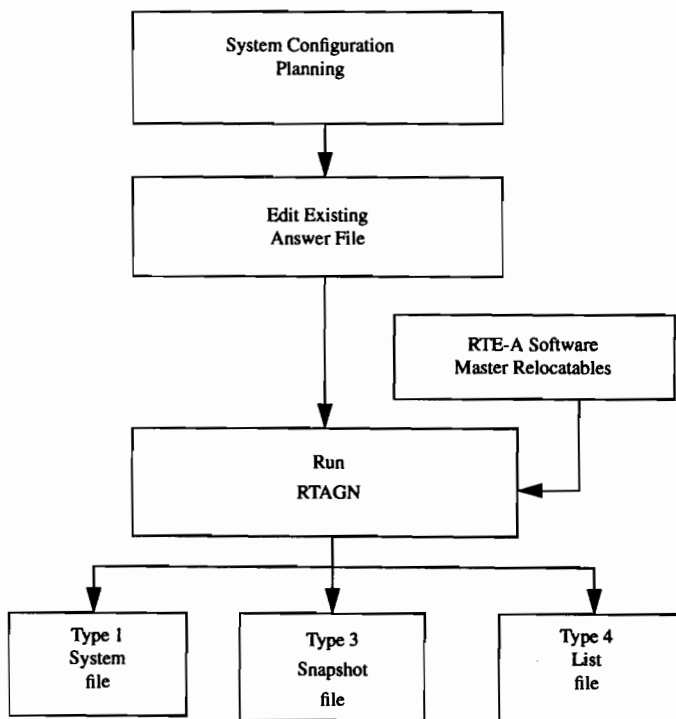


Diagram 1. RTE-A System Generation Process

Methods to Maximize System Space

The HP 1000 computer system has been in reliable operation for over twenty-five years. The latest member of this family is the A990 system released in 1991. The maximum memory configuration for the A990 is 32 megabytes, however the system is designed to operate in 64 kbytes of logical space. This limitation is dictated by the 16-bit architecture. As the system has grown over the years, the need to maximize system space has become very pressing. The following sections describe methods to conserve system space.

HP-IB Interface Controller

For an RTE-A system containing HP-IB cards there are two methods of addressing the device from the programmer's viewpoint. The first method is automatically addressing the device via the LU thru the DVT. The second method is a direct I/O method, in which a DVT must be assigned for the bus controller. In the direct I/O method, the user is fully responsible for the control and correct operation of the HP-IB bus. In order to use the direct I/O method, a DVT of twenty-six words must be assigned for the bus controller. The value of octal 36 is assigned to represent direct I/O in the first word of the driver parameter table. For systems with HP-IB cards that do not perform the direct I/O method of accessing the devices, the controller DVT can be removed and twenty-six words are saved in system memory.

A900/A990 Generations and the Parity Module

High-speed computer memory is subject to intermittent errors. RTE-A detects these errors through parity checking or corrects them with Error Correcting Circuitry (ECC). If an error occurs that cannot be corrected by hardware, the RTE-A system handles the error.

If the error occurs in memory occupied by the operating system or the operating system tables, or if the parity error handling module (PERR) was not included at generation time, the system is halted. This is an unrecoverable error.

On all A-Series systems except A900/A990, if the error occurs in a program, the system marks the page as bad, aborts the program and reports the error. Other programs continue to run and the offending program can be rerun later, usually without error. On an A900/A990 (due to caching), if an uncorrectable error is detected by the ECC memory the system is halted. This is an unrecoverable error.

Since the A900 and A990 will always halt on a parity error, the parity module (PERR) is really not needed in the generation. The deletion of the PERR module will save base page links, tag space, and partition space.

Reduce the Number of Memory Descriptors

Memory descriptors are system data structures designed to manage reserved and dynamic memory partitions. The number of memory descriptors can be reduced in order to save system table space. The generator can determine from the replacement (RPL) value whether the system is a code and data separation (CDS) or non-CDS system. This information is used to calculate the default number of memory descriptors for the system, according to the following formula. For a CDS system the number of memory descriptors is four times the number of ID segments. For a non-CDS system the number of memory descriptors is two times the number of ID segments. A CDS system requires more mem-

ory descriptors because a CDS program has at least two partitions, one for program code and one for data. A non-CDS program requires only one memory descriptor for its program code. The memory descriptor command (RS) is used to change the default allocation of memory descriptors. A positive value for the RS command will add the specified value to the calculated default value. A negative value for the RS command will subtract the specified value from the calculated default value. A system manager can evaluate the number of programs actually run (ID segments used) on a stable basis and determine a new number for the RS command. This number can be safely subtracted out of a first trial RS number. Since each member of the memory descriptor is 7 words in length, the system table will gain valuable space from each memory descriptor reduction. For a general purpose system with 85 ID segments, a RS value of -60 can be used.

The Tag Area Can Be Reduced

The tag command (TG) must appear in the system relocation phase if operating system modules are relocated into partitions during the partition relocation phase. The TG command specifies the size needed for tags. Tags are interface routines created by the generator for use by RTE-A to enter and exit partition modules. The number of tag area words required for each partitionable system module is found in the System Generation and Installation Manual. To determine the tag area size for your generation, add up the number of tag area words for all system modules placed in a partition, and add two to this value for each module (except \$IDRPL, IORQ, NSABP, or ABORT) that you use in your generation but do not include in a partition. Add one to the final value and use the total value in the TG command. For example, the partitionable modules \$IDRPL(6,0), CLASS(91,2), MEMORY(85,2) and SIGNL(53,2) are selected for a partition and their number of words for the tag area in a partition and in main memory are given inside the parenthesis, then the total number of tag words is $236 = 6 + 91 + 85 + 53 + 1$. RTAGN allocates the specified number of words for the tag area. When the partitioning phase is complete, RTAGN states how many words remain in the tag area and how the tag was used. If you allocate a larger tag area than is required, the extra space is wasted. If there are wasted words, you can change the TG command so that it specifies the actual number of words needed and then regenerate the system. Thus with a quick look at the generator listing, you can gain additional system space.

The Resource Number Command

The resource number command (RESN) allocates the maximum resource numbers for use in the system, and allocates the maximum number of programs that can be debugged simultaneously. The second parameter of the RESN command is the value that specifies the number of programs that can be debugged simultaneously. This optional parameter for debug, if not specified, is defaulted to five words. However if simultaneous program debugging is not required, then the value zero (0) can be specified for a savings of five words in the system table area.

The Number of Users Command

The maximum number of concurrent users in a system is specified by the users command (US). The format of the user command is US,n where the value of n will create a user table of $n \times 22$ words. This user table or user ID table is a memory-resident table containing all pertinent user identification data. It keeps a record of all users logged on to the system. Each time a user logs on, an entry is made to this table. When that user logs off the system, the entry is released for use with another user that is logged on to the system. If the number of concurrent users can be reduced without reducing productivity, then it is

worthwhile to examine how many users are actually logged on concurrently. A new generation can be done once this new number is determined, thereby saving twenty-two words per each user deleted.

System Memory Block

System memory block (SMB) or memory block (MB) is a memory area in the system map that is specified at system generation time by the generator MB command. SMB is used for DS/1000-IV Compatible Services tables for transaction and user information. When you initialize a node, NSINIT prints the number of words required for the SMB according to configuration values you specify. If you allocate a larger memory block than is required, the extra space is wasted. If there are wasted words, you can change the MB command so that it specifies the actual number of words needed and then regenerate the system. SMB requirements are summarized in table 1.

If you do not have DS/1000-IV Compatible Services generated into the system, then you need only one word for the SMB.

If you configure the default quantities for DS/1000-IV (RTE-RTE), you will need 170 words for the SMB.

If you configure the maximum quantities for all of the items in the table, you will need 3312 words for the SMB.

The following four sections are a guideline in calculating the SMB value for each type of network service. The sum of the words required for all four types of network services is used in the generator command.

DS/1000-IV Transaction

Each programmatic or remote request (such as REMAT, REMOTE, PTOPI, HELLO, Remote I/O Mapping) that uses DS/1000-IV Compatible Services requires a Transaction Control Block (TCB). Each TCB requires six words in the SMB and is configured in the DS/1000-IV Compatible Services section of the NSINIT dialogue. The default number of TCBs is twenty.

DS/1000-IV Remote Access (RTE-RTE)

Each concurrent PTOPI or EXECW schedule request from a remote node to the local node uses an entry in the POOL table. The POOL table header requires one word in the SMB; each POOL entry requires seven words in the SMB. The maximum number of concurrent DS/1000-IV remote transactions is configured in the DS/1000-IV Compatible Services section of the NSINIT dialogue. The default number of POOL entries is seven.

DS/1000-IV Remote Users (RTE-MPE)

Each concurrent NS/3000 or DS/3000 request that accesses this node to use DS/1000-IV Compatible Services (RTE-MPE) requires an entry in the Transaction Status Table (TST). Each TST entry requires fourteen words in the SMB. The maximum number of concurrent DS/1000-IV remote users (RTE-MPE) is configured in the DS/1000-IV Compatible Services section of the NSINIT dialogue. The default number of TST entries is four.

LU's for DS/1000-IV Compatible Services (RTE-MPE)

Each PSI/Bisync link and X.25 pool LU used for DS/1000-IV Compatible Services (1000-3000) requires a DS/1000-IV - DS/3000 LU table entry. Each table entry requires two words in the SMB. The number of X.25 pool LUs and the PSI/Bisync LUs used for DS/1000-IV Compatible Services (RTE-MPE) are configured in the DS/1000-IV Compatible Transport section of the NSINIT dialogue. The default number of entries is zero.

Table 1. System Memory Block Guideline Table

Activity	Item	Amount of Memory	Comments
DS/1000-IV transaction	Transaction control block(TCB)	6 words	Each program or remote access that uses DS/1000-IV Compatible Services requires a TCB. The default number of TCBs is 20; the maximum number is 200.
Remote Access (RTE-RTE)	POOL table entries	1 word table header plus 7 words per entry	Each concurrent program scheduled by a PTOPM or EXECW request from a remote node at the local node uses an entry in the POOL table. The default number of POOL entries is 7; the maximum number allowed is 253.
Remote Users (RTE-MPE)	Transaction Status Table (TST) entries	14 words	Each remote user that accesses the local node to use DS/1000-IV Compatible Services (RTE-MPE) requires a TST entry. The default number of TST entries is 4; the maximum number allowed is 10.
LUs for DS/1000-IV Compatible Services (RTE-MPE)	DS/1000-IV-DS/3000 LU table entries	2 words	Each X.25 pool LU and Bisync LU used for DS/1000-IV Compatible Services (RTE-MPE) requires an entry. The default number of entries is 0; the maximum number of entries allowed is 100.

System Generator Linking

The A-Series architecture enables certain machine instructions to address only one page of memory directly. If the operand for any of these instructions does not lie on the same page as the instruction then a link word must be allocated on either the same page or on the base page. Due to the original architectural design, the system contains only one base page of 2000B (octal) words. In addition the system's interrupt trap cells must also reside on the base page. Thus base page usage for linking purposes in the system generator is a precious commodity which must be used carefully. The following sections describe in detail generator linking and base page conservation.

Introduction to Generator Linking

A link word is created by the RTAGN program whenever a one-word memory reference instruction accesses a location not on the same page as the instruction. This is required because these memory reference instructions contain only a 10-bit field for the address to be referenced, the remaining bits being used for the OP code of the instruction itself. These 10 bits allow the instruction to directly access 1024 words of memory, or one page.

In the case where the data is not located on the same page, the system generator changes the instruction to an indirect reference (the sign bit is 1) to a link word on the same page (current page link) or on the base page (base page link). The relocating loader (LINK) also creates link words for programs that it links.

The link word contains the 15-bit address of the location to be accessed. A 15-bit address allows 5 bits to be used as the logical page number (accessing 32 pages), and 10 bits to be used for word offset on the page (1024 words per page).

For example, if the instruction is an LDA at location 2020 and the location to access (load into the A-register) is 4000B, then the generator can use either of the following:

Base Page Linking			Current Page Linking		
	Location	Contents	Location	Contents	
Base Page	0300	4000 ← BP Link	0300	????	
Page 2	2010	????	2010	4000 ← CP Link	
	2020	160300 ← LDA Data	2020	162010 ← LDA Data	
Page 3	4000	xxxx ← Data	4000	xxxx ← Data	

Where ??? indicates that the contents are irrelevant.

Note that the LDA instruction format is:

I 110 OPx xxx xxx xxx

x = memory address

P = 0 for base page link

P = 1 for current page link

I = 0 for direct addressing

I = 1 for indirect addressing

Generator Current Page Linking

The system generator can be directed to use either base page links or current page links. If current page linking is specified, some base page links may have to be allocated by RTAGN due to the following factors:

1. If a module crosses more than one page boundary, any links generated by the portion of the module that completely fills a page must be put on the base page.
2. The links required for references to other pages are estimated before the module is relocated. In some cases, the estimate may not be large enough and any links over the estimate must go to the base page.

Diagram 2 shows how memory is used for current page links.

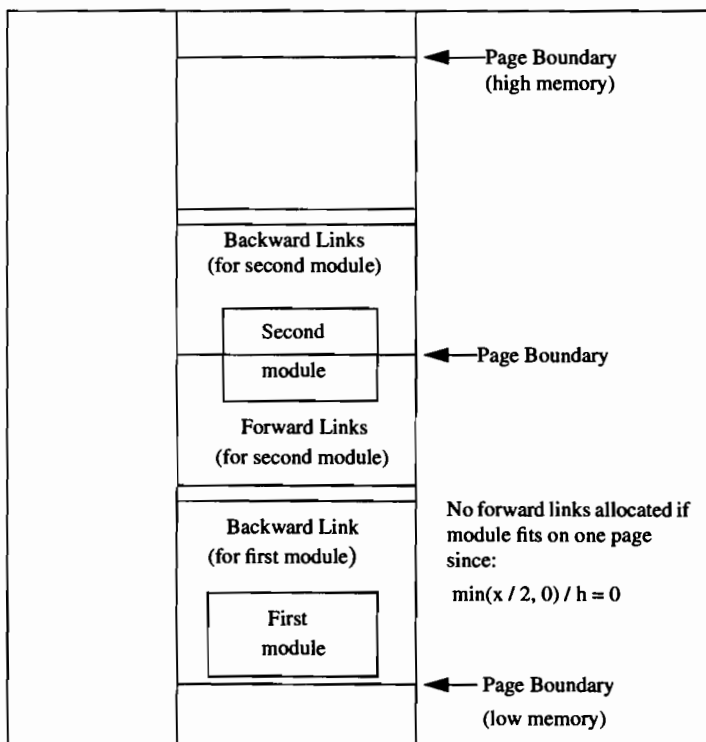


Diagram 2 . Memory Usage for Current Page Links

The algorithm for estimating the number of forward links that may be required is:

$$\min(x/2, y) / h$$

min = function to take the minimum of the two values

x = number of words left on the page at start of process

y = number of words of module on the next page

h = a constant (equal to 16)

The algorithm assumes an even distribution of the references requiring the formation of links. In some cases, this may cause an overestimate of the links required and some of the forward link area may be unused. In other cases, the algorithm may underestimate the requirement and then base page links have to be generated for the overflow.

The start of the backward link area is computed by adding the size of the module (determined from the NAM record) to the starting address of the relocated module. The size of this area is equal to the number of links required by the portion of the module relocated on this page. The relocation of the next module starts immediately after the last link is generated.

Generator Base Page Linking

During the relocation process, the generator creates some base page links whether or not current page linking was specified. If current page linking was specified, significantly fewer base page links are generated than if base page linking was specified.

During system common relocation, any base page links required are allocated starting from the top of the user page (location 1777B) down toward low memory. The links are stored in the snapshot file, not in the system file. They are copied into the base page link area of each program that accesses system common as it is relocated.

Links required by the operating system modules and drivers are allocated starting from the VCP/Loader temporary storage area in the system base page (location 1700B) toward the lower boundary (location 112B).

Base Page Optimization Method

When RTAGN reports base page overflow, the user can quickly scan the map listing for system modules with large base page usage and page boundary crossings. The modules that fit the above criteria can then be rearranged in a new relocation order. For example: if the current listing shows that the MAPS module has a size of 1520B, uses forty-five base page links, and is located from 15311B to 17031B crossing a page boundary, then it is a good candidate to be moved. A good potential spot would be a page with greater than 1520B locations that can be allocated to the MAPS module.

Base Page Optimization Study

In the RTE-A Operating System, there are a total of nine required and non-partitionable system modules. These modules (with their 6.1 Release size in octal) are EXEC (1776B), IOMOD (2106B), MAPS (1520B), PROGS (1375B), RPL_A990_CDS (0B), RTIOA (1737B), SAM (363B), UTIL (2011B), and VCTR (1550B). A study was conducted to determine if an optimal relocation order can be found that will use a minimum number of base page links for these required and non-partitionable modules. In addition, the system module SCHED was included in the base page minimization study.

To start the optimization process, the required modules were first randomly ordered. This order was placed in a generation answer file and a system was generated. At the end of the generation, the base page usage for each required module was analyzed with the generation map. A new order was then created and a new system was generated. The generations and analyses of base page usage were repeated. The results of the study, obtained by many generations, suggest the following order of system modules can produce a system that uses fewer base page links than other orders. The order is VCTR, RPL_A990_CDS, UTIL, SCHED, EXEC, RTIOA, IOMOD, SAM, MAPS, and PROGS. Note that this order is not an absolute minimum since the number of order combinations is so large that it was impractical to try every combination.

Conclusion

RTAGN has been in existence for over a decade and is doing its job of running reliably, very similar to the HP 1000 hardware and the RTE-A Operating System. By using the above methods of saving system table space and the methods of base page minimizations, many users can gain additional system space and base page links.

Acknowledgment and References

Special thanks to Dave Medicott who studied the base page optimization problem earlier. Information and graphs were obtained from the RTE-A manuals, RTE-A Technical Specifications, and from the actual code in the system generator.

HP-RT 2.0 — A Mile High Above the Rest by Eric R. Mostardi

Hewlett-Packard
11000 Wolfe Road
Cupertino, CA 95014

HP-RT 2.0 represents the latest milestone in HP's 29 year history of real-time products. With this latest release, HP-RT adds new functionality, enhanced performance, and support of HP's newest VME board computer, the Series 9000 Model 743rt. Supported at HP-RT revision 2.0 is our newest software offering, STREAMSrt, providing a standard's based API for developing communications services. Also featured is a new VME Backplane Networking Product, support for compilers for both FORTRAN and Ada (from a third party), and many other enhancements.

Detailed below are many of the new enhancements included in the latest revision of HP-RT.

Model 743rt

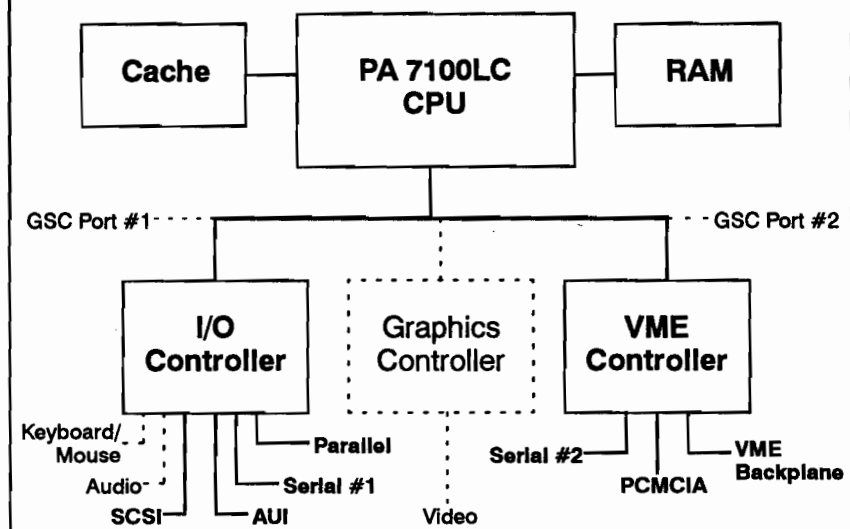
Hewlett-Packard's newest member of the real-time VME board family is the Model 743rt, offering unparalleled performance in a single slot 6U form-factor VME card. With performance options of either 64MHz or 100MHz, the Model 743rt is HP's fastest real-time processor. Compared to the Model 742rt, HP's popular 2 board VME computer, the Model 743rt provides greater memory expansion (up to 256MBytes), and faster DMA transfers over VME using the Model 743rt's new on-chip DMA controller. New with the Model 743rt is support of full D64 DMA transfers for both Master and Slave processors.

For Product Ordering Information, see Appendix A.

Model 743rt Hardware DMA Engine

A new feature on the Model 743rt CPU board is the addition of a VME DMA controller chip which provides Hardware DMA to and from the VME backplane. The DMA controller chip is used whenever the software detects that a copy is being performed between the Model 743rt's physical memory and a SWSM mapped to a remote VME device. These new VME routines support the D16, D32, and D64 data transfer mode. D64 is not supported on the Model 742rt processor. True Read-Modify-Write has also been implemented within the Model 743rt's DMA controller. Read-Modify-Write is used to read and modify semaphores which are used to protect the backplane networking shared resources.

Model 743rt Block Diagram



Dashed lines indicate functionality not currently supported under HP-RT

BOOTP

The BOOTP protocol is an industry standard remote boot protocol which will allow the HP-RT operating system to boot from either HP-UX or HP-RT systems.

The BOOTP/TFTP methodology will replace the proprietary RMP (Remote Maintenance Protocol) used on the Model 742rt. The BOOTP protocol also provides faster boot up times when compared to the Model 742rt. BOOTP is used to boot over LAN or the VME backplane. To boot a Model 743rt over the VMEbus, Version 2.0 of the VME Backplane Networking product is required (HP Part Number B3802AA and B3803AA).

Since BOOTP is an industry standard boot protocol, it also provides the capability to boot from non-HP servers. Although this functionality is not currently supported, we do expect that in general, we will boot from an arbitrary boot server.

Table 1. Boot protocol support matrix

	RMP	BOOTP
Model 742rt	Supported	N/A
Model 743rt	N/A	Supported

STREAMSrt

The newest member in the HP-RT software family is the STREAMSrt product. The STREAMSrt product supports STREAMS SVR3.2 to assist developers in writing modular I/O drivers using standard STREAMS API's and system calls. STREAMSrt supports over 25 STREAMS utilities and macros to assist in the development of STREAMS modules and drivers.

Backplane Networking 2.0

Revision 2.0 of the VME Backplane Networking product combined with revision 2.0 of HP-RT, now features enhanced support for networking over the VMEbus. Support of the new Model 743rt processor includes both booting and communicating over the VMEbus, as well as taking advantage of the Model 743rt's on-chip DMA controller for faster VME transfers. The new revision of the VME Backplane Networking product also supports HP-UX to HP-UX communication over the VMEbus, allowing Series 700i Systems to communicate with each other.

The boot capabilities of the VME Backplane Networking product have also been enhanced, allowing the booting of an HP-RT system from either HP-UX or another HP-RT system. Booting can be accomplished by either the RMP protocol used in the 742rt product, or the BOOTP protocol used on the Model 743rt product.

Table 2. Backplane Networking Compatibility Matrix

	BP Net rev 1.0	BP Net rev 2.0
HP-RT rev 1.0	Supported	Not Supported
HP-RT rev 2.0	Not Supported	Supported

With the capabilities of the on-chip DMA controller on the Model 743rt, the VME Backplane Networking product supports true Read-Modify-Write cycles. This allows networking communication between the Model 743rt running HP-RT and other non-HP computer systems that implement hardware Read-Modify-Write cycles.

POSIX Certification

The HP-RT Operating System is currently undergoing NIST certification to the POSIX 1003.1 standard, using FIPS test suite 151-2.

HP-RT also complies with draft 9 of the POSIX 1003.4 standard, and draft 4 of the POSIX 1003.4a standard.

New Peripheral Support

For HP-RT revision 2.0, the following new peripheral devices have been added to the list of supported peripherals for HP-RT. In addition to the devices supported

under earlier releases of HP-RT, the following devices will be supported using HP-RT revision 2.0. Please contact Hewlett-Packard for a complete list of supported peripherals on the HP-RT operating system.

Table 3. New Peripherals Supported at HP-RT Revision 2.0

Product Number	Option	Description	List Price
C3040R	002	2GByte Rack Mount Disk Drive	\$4,560
C3040T	002	2GByte Mini-Tower Disk Drive	\$4,560
C3041R	002	2x 2GByte Rack Mount Disk Drive	\$7,450
C3041T	002	2x 2GByte Mini-Tower Disk Drive	\$7,450

Header File Constant Alignment

In order to facilitate future binary compatibility between HP-RT and HP-UX, the header (include) files for HP-RT have been modified to make them as similar as possible to HP-UX. All constants and structures which are in both HP-UX and HP-RT with the same name have been aligned to have the same value and structure.

HP-UX release 9.0 values and structures will be used unless there is a conflict with the POSIX 1003.1 specification, in which case, the POSIX structure will be used. Because of this change, all programs developed on HP-RT revision 1.0 or 1.1 will need to be re-compiled to run on HP-RT 2.0.

PCMCIA Support for the Model 743rt (Special Product Only)

Using the Model 743rt's mezzanine board support capability, the first mezzanine card offering, the PCMCIA adaptor, will be available soon. At its initial release, HP will support a 10MByte and a 20MByte Flashdisk as supported peripherals in the PCMCIA slot. The Flashdisk will appear as a disk drive to the HP-RT operating system, onto which an HP-RT file system can be loaded.

PCMCIA Features:

- HP-RT Operating system can be booted from an on-board PCMCIA flashdisk
- Fast transfer rate for reads from the PCMCIA card. Flash disk read performance is typically better than reads from either LAN or hard disk drive.
- Choice of either 10MByte or 20MByte flashdisk drive
- The flashdisk appears as a fully functional hard disk with full read/write capabilities for support of the HP-RT operating system

- The file system on the PCMCIA flash disk can be NFS mounted to other HP-RT systems via LAN or VME Backplane Networking.

PCMCIA Usage Notes:

Memory Limitations: On the Model 743rt CPU board, there are two memory connectors available for use, one at each end of the processor board. Since the PCMCIA adaptor covers up the memory connector on the left hand side of the CPU board, the left-hand memory connector cannot be used if the PCMCIA adaptor is installed. This limits the amount of supportable RAM on a Model 743rt CPU board to one memory module, or 64MBytes when used in a single slot configuration. To expand the memory past 64MBytes, the Model 743rt expansion module is required, converting the Model 743rt to a dual VME slot computer.

VME Backplane Limitations: The Model 743rt's VME backplane DMA controller chip controls both VME backplane transfers and PCMCIA transfers. Because all PCMCIA transactions occur using the VME backplane DMA controller chip, and since the DMA controller requires that it control the VMEbus prior to any transactions, the VMEbus is inaccessible to any card in the card cage during PCMCIA transactions.

Model 743rt Boot Firmware: The first units of the Model 743rt CPU board that were shipped do not support booting from the PCMCIA adaptor. Please contact Hewlett-Packard to obtain information on acquiring the correct version of the Model 743rt firmware required to boot from the PCMCIA adapter.

FORTRAN Support

New with HP-RT 2.0 is support of the HP-UX FORTRAN compiler and the HP-UX 9.0 standard FORTRAN library. With this functionality, porting existing applications written in FORTRAN to HP-RT is greatly simplified. HP-RT 2.0 supports the complete FORTRAN 77 instruction set, as well as several FORTRAN 90 enhancements.

To support the compilation and linking of FORTRAN programs under the HP-UX development environment, HP-RT utilizes the standard HP-UX FORTRAN compiler (`f77`) and common language libraries. Support of the FORTRAN environment also includes support of the symbolic debugger, `rtdb`.

The HP-UX FORTRAN library is not thread safe, so thread based FORTRAN programs cannot be guaranteed to operate correctly.

Clock-Tick Handler

Prior to revision 2.0 of the HP-RT operating system, the clock-tick handler ran entirely on the Interrupt Control Stack. Since code that resides on the Interrupt Con-

trol Stack cannot be context switched, the entire clock-tick handler was being executed with context switching disabled.

At 2.0, the clock-tick interrupt is handled by a very fast assembly language interrupt handler which schedules a kernel thread only when a time-triggered event occurs. When a time-triggered event does occur, the clock-tick handler thread will run at boosted priority (one half priority above the level of the requester).

This new enhancement will increase the performance of your critical applications since significant clock tick servicing now runs at a user priority, NOT at interrupt priority via execution on an interrupt stack.

rtmon Improvements

rtmon, the real-time monitor program provides a snapshot of the current state of the HP-RT operating system. rtmon provides status on system resources, interrupts, CPU usage, and other values relevant to the processes and threads currently being executed.

For HP-RT revision 2.0, the following enhancements have been made to rtmon:

Next Process/Thread ID: This panel will display the name of the thread that will be executed when rtmon sleeps, and the name of the process which spawned that thread.

Virtual Memory Usage: This panel will display the number of swappable pages, the number of locked pages, and the number of page table pages used by the Virtual Memory System.

Additional entries into the "Interrupts per second" table:

- VME Backplane Networking Interrupts
- PCMCIA Status Change Interrupts

User definable counters: There are 40 user definable counters that can be used to display data from user processes or I/O device drivers. These counters can be thought of as general purpose integral accumulators.

The configurable parameters for the rtmon program are located in the `/etc/rtmonrc` file. After the `/etc/rtmonrc` file is read, the kernel searches for an `rtmonrc` file located in the user's startup directory. The user's `rtmonrc` file may be used to add more elements to the display that are of special interest to a particular user. This `rtmonrc` file will describe the location, label, print format, and the `user_counts` array elements to be added to the display. The user then adds code in his/her driver, or other kernel level code to increment the `user_counts` elements.

Performance Improvements: There have also been changes regarding how data is accessed within the kernel by the rtmon program, improving the performance of rtmon.

Profiling

Support of C and FORTRAN program profiling has been added to HP-RT 2.0. The HP-RT Profiling feature can be used to analyze an application's execution time as well as time spent handling system calls. This will allow the `prof ()` and `gprof ()` utilities to interpret and display the profile data.

Additional Enhancements

- Up to 512 User Priority Levels – User definable
- Support of up to 256MBytes of memory on the Model 743rt computer
- Improvements to man pages
- Floating point support within the kernel
- Improved Paging Algorithm
- Modem support
- Serial port performance improvements
- Fixes for over 150 SRs
- Increased number of shared memory segments per process from 16 to 114
- Support of HP-UX 9.0's vi editor

Appendix A - Ordering Instructions for HP-RT and related products:

Table A-1. HP 9000 Model 743rt

Product Number	Option	Description	List Price
A4261A		HP 9000 Model 743rt VME Board Computer	\$4,355
	201	64 MHz Version	N/C
	203	100 MHz Version	\$7,200
Model 743rt Accessories			
A4262A		Model 743 Expansion Kit	100
A4263A		8 MByte RAM Card	\$640
A4264A		16 MByte RAM Card	\$1,280
A4265A		32 MByte RAM Card	\$2,880
A4266A		64 MByte RAM Card	\$5,760
A4300A		Special HP Parallel Cable	\$35
A4301A		Special RS-232C Cable	\$35
A4303A		Special LAN AUI Cable	\$35

Table A-2. HP 9000 Model 742rt

Product Number	Option	Description	List Price
A2260A		HP 9000 Model 742rt HP 9000 Model 742rt is a VMEbus board computer using state-of-the-art HP-PA RISC processor to provide high real-time performance when coupled with the HP-RT standards based real-time operating system.	\$7,995
	#ANB	Substitute 16 MB DRAM for 8 MB DRAM	\$800
	#ANC	Substitute 32 MB DRAM for 8 MB DRAM	\$2,240
	#ANE	Substitute 64 MB DRAM for 8 MB DRAM	\$5,120

Table A-3. HP-RT Developer's Kit License

Product Number	Option	Description	List Price
B3799AB		HP-RT Developer's Kit License This product provides a license for one developer to use the HP-RT Developer's Kit to develop high performance real-time applications. One license is required for each developer.	\$9,995
Price Breakpoints			
		Price per license for 4-8 concurrently purchased license	\$3,995
		Price per license for 9 or more concurrently purchased license	\$2,995

Table A-4. HP-RT Developer's Kit Software and Manuals

Product Number	Option	Description	List Price
B3800AA		HP-RT Developer's Kit Software and Manuals This product provides the C/ANSI C, C++, X client and Motif libraries and tools for developers to develop HP-RT real-time target drivers and applications. It is hosted on an HP 9000 Series 700/800 system running HP-UX Release 9.0. Simultaneous purchase of an HP-RT Developer's Kit License (B3799AA) is required. Manuals are included.	\$1,195
One of the following options must be specified.			
	#AAH	Provides software on DDS tape.	N/C
	#AAU	CD-ROM Certificate.	N/C

Table A-5. Backplane Networking License

Product Number	Option	Description	List Price
B3802AA		VME Backplane Networking License for One HP 9000 System This product provides one license for an HP-RT or HP-UX system using the Backplane Networking product.	\$1,495
Price Breakpoints			
		Price per license for 10 or more concurrently purchased license	\$395

Table A-6. Backplane Networking Software and Manuals

Product Number	Option	Description	List Price
B3803AA		VME Backplane Networking Software and Manuals This product provides software and manuals to install and use the VME Backplane Networking drivers. It allows an HP-RT and HP-UX systems to communicate with other HP-RT, HP-UX (Release 9.0) and VxWorks (Release 5.0) systems. Simultaneous purchase of a VME Backplane Networking License (B3802AA) is required. Manuals are included.	\$495
Must order one of the following media options.			
	#AAH	Provides software on DDS tape.	N/C
	#AAU	CD-ROM Certificate.	N/C

Table A-7. OTSrt License

Product Number	Option	Description	List Price
B3805AA		OTSrt License for One HP 9000 S/700rt System This product provides one OSI Transport Service for HP-RT (OTSrt) license to use on one HP 9000 Series 700rt System. One license is required per system.	\$2,995

Table A-8. OTSrt Software and Manuals

Product Number	Option	Description	List Price
B3806AA		OTSrt Software and Manuals This product provides the tools and files necessary to install and use OTSrt in HP-RT. Simultaneous purchase of an OTSrt License (B3805AA) is required. Manuals are included.	\$495
Must order one of the following media options			
	#AAH	Provides software on DDS tape.	N/C
	#AAU	CD-ROM Certificate.	N/C

Table A-9. STREAMS_{rt} License

Product Number	Option	Description	List Price
B3808AA		STREAMS _{rt} License for One HP 9000 S/700 _{rt} System One STREAMS _{rt} Run-time license is required for each HP-RT system that has STREAMS _{rt} installed.	\$1,995
Price Breakpoints			
		Price per license for 10 or more concurrently purchased license	\$595

Table A-10. STREAMS_{rt} Software and Manuals

Product Number	Option	Description	List Price
B3809AA		STREAMS _{rt} Software and Manual This product provides the files and tools necessary to develop drivers using the STREAMS interface into the HP-RT kernel (Release 2.0). Simultaneous purchase of an STREAMS _{rt} License (B3808AA) is required. Manuals are included.	\$495
Must order one of the following media options			
	#AAH	Provides software on DDS tape.	N/C
	#AAU	CD-ROM Certificate.	N/C



**Paper Number 7002
Porting from HP-UX to HP-RT**

**Frances Huang
Technical Marketing Engineer
c/o Ella Washington
Hewlett-Packard Co.
408.447.1053**

Handouts will be provided at time of presentation

7003

RTE-A to HP-UX Migration Tools - An Update

Compiled by: Kathy Anderson
Hewlett-Packard
11000 Wolfe Road
MS 42UN
Cupertino, CA 95014
408-447-6955

Abstract: Hewlett-Packard has committed to providing migration tools for HP 1000 RTE-A users who have a need to migrate their RTE-A programs to an HP 32-bit platform. This paper will give an update on those tools which will assist users in migrating their programs from RTE-A to HP-UX when absolute real-time is not a requirement.

Overview: The RTE-A to HP-UX Migration Tools comprise a package consisting of documentation and software tools. The tools will assist RTE-A users in their migration of FORTRAN programs from the HP 1000 running RTE-A to the HP 9000 running HP-UX.

The following are the planned components for the RTE-A to HP-UX Migration Tools package:

- *RTE-A Migration Tools User's Guide*
- *RTE-A Migration Tools Reference*
- Migration Analysis Tool (mat)
- FORTRAN Migrator Tool (ftnmig)
- RTE-A Migration Library

All of the above components are provided on HP-UX. The library and tools reside and run on HP-UX. The Migration Analysis Tool (mat) also runs on RTE-A.

The purpose of the RTE-A to HP-UX Migration Tools is to help HP 1000 users migrate RTE-A FORTRAN programs to the HP 9000 HP-UX native mode environment. The Migration Analysis Tool (mat) analyzes RTE-A FORTRAN source code for portability. The FORTRAN Migrator Tool (ftnmig) migrates FORTRAN 7X source code that is incompatible with HP FORTRAN 77. The RTE-A Migration Library provides HP-UX implementations of over one hundred HP RTE-A functions without the need for operating system dependent daemons and resources. The *RTE-A Migration Tools User's Guide* provides step-by-step "cookbook" instructions, as well as complete examples to further assist users in migrating these programs to HP-UX native mode. The *RTE-A Migration Tools Reference* provides

detailed information on the use of the above tools and the functions provided by the library. With these tools and documentation, HP 1000 users will be able to migrate, compile and run their RTE-A FORTRAN programs under HP-UX as quickly as possible, thus allowing them more valuable time in debugging to ensure continuity.

The following sections provide an overview of each component included in the RTE-A to HP-UX Migration Tools package.

Documentation: The documentation is a resource and cookbook to assist in migrating a FORTRAN program from RTE-A to HP-UX. It will explain which programs can be migrated, which ones cannot be migrated and it will explain the steps necessary in performing a migration. It will also provide information on how to make programs native to HP-UX.

The target audience is FORTRAN programmers experienced with RTE-A who may not be familiar with HP-UX.

Man pages and a User's guide will be provided both online and hard-copy. Online examples will also be provided.

The *RTE-A Migration Tools User's Guide* will provide the following information:

- Introduction
- Installation and Setup
- Cookbook Approach to Migration
- Migration Tools
- Other Considerations for Your Migration Project
- *or* How to Go Native
- Error Messages
- Glossary

Migration Analysis Tool (mat) Overview: The Migration Analysis Tool, mat, analyzes RTE-A FORTRAN source code for portability to HP-UX native mode. This is usually the first step in migrating RTE-A programs to HP-UX native mode. The tool scans source code, flagging RTE-A routines that are not implemented by the RTE-A Migration Library. The tool also flags EXEC calls used as functions and EXEC calls with an alternate error return followed by a subsequent GOTO statement since the HP-UX FORTRAN 77 compiler does not support these capabilities. The tool provides metrics to help users estimate the effort required to migrate RTE-A FORTRAN programs to HP-UX.

Output from the Migration Analysis Tool includes all flagged statements together with warning messages describing why the statements are flagged. The output concludes with a statistical profile of the number and type of routines flagged.

The Migration Analysis Tool is system independent and runs on both RTE-A and HP-UX systems. This enables FORTRAN source files to be analyzed on the RTE-A system and then transferred to HP-UX, or first transferred to HP-UX and then analyzed.

FORTRAN Migrator (FTNMIG) Overview: The FORTRAN Migrator Tool, `ftnmig`, scans FORTRAN 7X source code for language incompatibilities with HP FORTRAN 77. This tool modifies, comments out or removes most of the FORTRAN 7X compiler options, directives and statements incompatible with HP FORTRAN 77. Because the HP-UX FORTRAN 77 compiler does not support the FORTRAN 7X capabilities of EXEC calls used as functions or subroutine calls with alternate error returns via a subsequent GOTO statement, this tool will migrate these so they are compatible with FORTRAN 77.

RTE-A Migration Library Overview: By making use of the RTE-A Migration Library, users will be able to utilize over one hundred HP 1000 subroutines/functions/EXEC calls/and FMP calls on the HP-UX platform. Routines in the RTE-A Migration Library will have the same name as their HP 1000 counterparts and accept the same parameters, but they will perform functionality native to HP-UX, not relying on any shell or cooperating daemon.

The benefit to using such a library will be reduced time and effort in migrating a program. Users will be able to leave HP 1000 calls in place and gain functionality with little change to existing code. Also, HP-UX programming techniques (such as working with files and directories) would not have to be dealt with right away by the HP 1000 application developer; a familiar RTE-A interface could still be used to some extent.

The drawback is these routines may not be as efficient as possible given an individual program's use of the routine and not all functionality may be supported. For instance, leaving RTE-A FMP calls in place will continue the use of a DCB buffer. This buffer is not required in HP-UX, nor will fields of the DCB be updated as RTE-A would update them. Use of `fopen()` or other HP-UX file input/output routines would most likely make a more compact program. If a user's program looks at information within a DCB, that part of the program will have to be reworked.

Overall, the RTE-A Migration Library will be a good starting place for program migration. Once the program is up-and-running using the library, it can be converted to a more native-mode HP-UX program as time and expertise permits; the user can slowly remove library calls replacing them with HP-UX code.

Conclusion: Hewlett-Packard understands that many users plan to stay with the HP 1000 for years to come. It is Hewlett-Packard's strategy to provide HP 1000 users with lasting value for their systems well into the future; continued RTE software releases and the commitment to support RTE through the year 2010 are part of this strategy. In addition, in response to those users who are considering a move to an HP PA-RISC platform, the above described migration tools are being developed to further enhance this strategy.

Note: Since this project is still under development, more details and an update will be provided at the paper presentation.

Acknowledgements: Special thanks to Gary Thomsen, Scott Anderson, Gene Marshall, and Kathy Anderson for their contributions to this paper.

Paper 7004

Use of HP Optical Disk Libraries with HP 1000 Computers

Clifford E. Buffett
Warner-Lambert Company
170 Tabor Road
Morris Plains, NJ 07950
(201) 540-4764
buffetc@aa.wl.com

Introduction

A small computer group is responsible for several computers systems including four HP 1000 A990 computers. Three of the HP 1000 systems have 40 Gb HP optical disk libraries (jukeboxes) attached to them. Two of the systems collect and analyze data from several laboratories using HP Analytical's LAS (Laboratory Automation System) software. The third HP 1000 with a jukebox is a development system used to test software, to convert E-Series LAS data, for training, and many other supporting functions. Thousands of analyses are performed each month using analytical instruments attached to the LAS systems. The laboratories generate about 30 Mb of data daily and the data generated must be retained for many years and be readily accessible at any time. Data is often retrieved to compare previous results to current results and substantial data is restored during FDA inspections or in preparation for those inspections. Only optical disks have been used for backups of data during the past two and a half years. The jukeboxes were used to create those backups.

Brief History and Current Configuration

The first LAS data acquisition system used an E-Series HP 1000 computer. It adequately served the needs of a small department for many years. As the needs grew and newer peripherals became available, it was gradually upgraded. A major upgrade occurred when the system was replaced by an A-Series machine to serve a much larger laboratory group. Currently, the system consists of an A990 with 32 Mb of memory, 48 serial ports, 5 instrument loop interfaces, 2 disks, 2 tape drives, and a 40 Gb optical disk library. A second LAS system with similar capabilities has been installed and is in operation. In addition, there are two other HP 1000 Model A990 computers and one of those also has an optical

disk library attached to it. Each of the three jukeboxes are either a Model C1700T or an equivalent upgraded model, capable of storing 40 Gb of data on 32 1.3 Gb disks and equipped with two high performance, multifunction, double density drives.

Jukebox Advantages

Previous to the purchase of any optical disk drives, all backups were performed using either cartridge tapes or reel tapes. Even using 2400 ft. reels at 6250 bpi, only about 140 Mb of data can be stored on a tape. This is one of the initial reasons that optical disks looked attractive, storing 650 Mb on each 5.25" disk. The recent introduction of double density disks increased the capacity to 1.3 Gb on each disk. The main reason that optical disks were considered was the much longer lifetime, a minimum of 30 years. Other advantages include much faster access and reduced physical space requirements. Although stand alone optical disk drives, such as the Model C1701A, were considered, a jukebox provides some additional capabilities. Besides providing two drives, the Model C1700T contains 32 shelves for disks and an autochanger to move disks between the drives, shelves, and an import/export slot. This allows unattended access to many disks, to perform backups or other activities that involve more than one disk. Although HP jukeboxes are available in several sizes, this particular size seemed to match our requirements.

Possible Drawbacks

At the time we considered purchasing the first jukebox, HP only supported the stand alone optical disk drive, not the jukebox. The purchase of the first jukebox included the development of drivers for the HP 1000 computer by Herstal Automation Ltd. as part of the order. Although the drivers proved to be a problem initially, the problems have been resolved and the jukeboxes are fully operational with the HP 1000 computers. With all the advantages of jukeboxes, we were willing to trust our developer to supply a working driver and we assumed some risk that HP might not ever support jukeboxes on HP 1000 computers. If a working driver could not be obtained, we would have been required to load disks using the front panel controls and use standard HP drivers to access only the two disk drives. In that case, the jukebox would be no better than two stand alone optical disk drives.

Data Organization and Backup Strategy

Data is divided into six global directories spread over two systems and several disk drives. By dividing the data files among several directories and disks, several advantages are realized. A single directory will not hold all the data that is generated in most 30 day periods. Access time is reduced and loss of one disk or system will not shut down all the laboratories.

Each user is assigned their own subdirectories to hold their data files. Data files are purged automatically every 30 days, to limit the size of the directories and to keep access time fast. A decision was made to perform two different backups, a backup of all updated files each day and a separate backup that copies all files in the data directories twice monthly.

Backups Before Jukeboxes

Until we obtained our first jukebox all backups were written with FST to reel or cartridge tapes. The management of data files was controlled by a command file that executed every night at 1 am. It scanned the data file timestamps for any files updated the previous day and appended them to the tape loaded on the system. Using 2400 ft. reels at 6250 bpi, about 140 Mb could be stored on each tape (from 4 to 7 days of data). The only requirement was to load a tape with sufficient space into the tape drive before leaving for the day. The command file would also delete data files more than 30 days old. In addition, the second type of backup performed twice monthly, a snapshot of all the data files, required multiple tapes.

Backups with Jukeboxes

The previous backup command file was changed to create directories on an optical disk and copy the files using the CO command from CI instead of using FST. All updated files from all the different data directories are copied to a single optical disk using corresponding directories on the optical disk. The names of the global directories on the optical disk have the update date as a suffix. One disk remains loaded and files are added to it each night. A single optical disk can contain several weeks of daily backups. The available space is monitored and when one side of the disk is nearly full, it is changed to Side B or replaced with a fresh disk. The following is a partial listing of the nightly backup command file (doesn't include commands to backup the /NEU, /COG, /PHA, and /QC directories) that runs at 1 am:

RU /TAPE/DATEX YYMMDD -1D
 puts date 1 day ago in yymmdd format
SET BACKUPDATE = \$RETURN_S
 stores date in variable \$BACKUPDATE
MC 27
 mounts optical disk
CRDIR /LAS\$BACKUPDATE 27
 creates global directory on optical disk
CO /DATA/@.DIR.K /LAS\$BACKUPDATE/
 create all necessary subdirectories
CO /DATA/@.@.KU\$BACKUPDATE /LAS\$BACKUPDATE/
 copies all updated files to optical disk
DC 27
 dismounts optical disk
RU /TAPE/DATEX YYMMDD -30D
 puts date 30 days ago in yymmdd format
SET PURGEDATE = \$RETURN_S
 stores date in variable \$PURGEDATE
PU /DATA/@.RES.SC-\$PURGEDATE OK
 purges all result (.RES) files from directory
 that are 30 or more days old
RU /PROGRAMS/CLOCK A990 SET
 resets A990 clock to current time
RU /PROGRAMS/FREES 27 +M
 displays free space on optical disk
 (the system console will show this in the
 morning - to determine when to change disks)
EX

For example, files from the /DATA directory that were updated on 6-14-94 were copied at 1 am 6-15-94 to a newly created /LAS940614 directory on the optical disk, then on the following night files updated on 6-15-94 were copied into directory /LAS940615, etc.

In addition, a separate backup of all data files is performed to optical disks twice monthly. Separate backup disks are used for each of the global data directories, for example, one disk has all the backups of the /COG directory. The names of the global directories on the optical disks have the backup date as a suffix. The twice monthly backups of the /DATA directory are performed with the following command file using the second drive:


```

*      CEB      <920729.0802 >
*
* CI command file to copy the global directory /DATA/ to optical
* disk on LU 28 using the name /DATAyymmdd/ on the optical disk.
*
*
RU /TAPE/DATEX YYMMDD 0D
                        puts date today in yymmdd format
SET BACKUPDATE = $RETURN_S
                        stores date in variable $BACKUPDATE

MC 28
                        mounts optical disk
CRDIR /DATA$BACKUPDATE 28
                        creates global directory on optical disk
CO /DATA/@.@.K /DATA$BACKUPDATE/
                        copies entire directory to optical disk

DC 28
                        dismounts optical disk

EX

```

For example, the backup of the files in the /DATA directory performed on 6-14-94 are in the /DATA940614 directory on an optical disk.

Accessing Optical Disk Backups

The second drive is also used to copy old data files from the optical disk backups to the active data directories. The files can also be looked at directly on the optical disks. To protect the files on the optical disks, the write protect tabs are engaged except when backups are scheduled. The driver can be controlled with CN commands to operate in either read only or read/write (normal) mode:

Read only mode is obtained with these commands (openoptical.cmd):

```

cn,28,40b
mc,28
cn,28,27b,no,wr,it

```

The following commands return the disk to read/write (normal) mode (closeoptical.cmd):

cn,28,27b,re,an,wr
dc,28
cn,28,41b

In the read only mode, no errors occur when reading disks with the write protect tab engaged or when reading WORM disks (the access timestamps are not updated). Also the backup disks are mounted and dismounted by the command files to protect them. To further protect data, each filled backup disk is duplicated onto a WORM disk and stored off site.

Evolution of the Three Jukeboxes

When the first jukebox (C1710M, multifunction, single density, "regular" performance, 2 drives, 32 shelves, 20 Gb) was obtained, the driver could not operate the autochanger so disks were loaded into the drives using the front panel controls. The C1710M jukebox was upgraded to the equivalent of a C1700T jukebox one year ago. The change to high performance drives has increased the speed of backups by a factor of two and a half.

An A-Series A600 system was obtained to use for training and development. A C1701C optical disk drive was purchased and attached to it. This provided an easy way to transfer a large number of files between systems on optical disks. Later the system was upgraded to an A990 and a C1700A jukebox was purchased and connected to the system. Using the C1700A and the C1701C together we were able to duplicate r/w disks to WORM disks. Eventually the jukebox was traded and upgraded to the equivalent of a C1700T jukebox and the C1701C was traded to another group in Warner-Lambert. The newer drives support both single and double density disks. Each of the drives were generated into the system with three 322 Mb LUs, one for the entire side of single density disks and two for the entire side of double density disks. Since RTE-A 5.27 was being used, it could only support up to 526 Mb LUs, each side of the double density disks had to be split into two LUs. RTE-A 6.1 supports larger LUs so a 6.1 system was generated with two LUs per drive, one 322 Mb LU for single density disks and one 650 Mb LU for double density disks.

Our third jukebox was part of a purchase of a second A990 based LAS system. The system was needed when the number of laboratories that wanted to be connected to LAS exceeded its capacity. The system was purchased with a new C1700T jukebox.

SCSI Driver

The problem that prevented the driver from operating the autochanger was identified as an error in the interface driver, IDQ35. HP supplied a patched version of the driver (the driver was fixed in 6.1) allowing the autochanger to be controlled by a program provided by Herstal Automation Ltd. The patched version was tested before upgrading existing jukeboxes or purchase of new jukeboxes.

Jukebox Command Files

Several command files were developed to utilize the unique capabilities of the jukeboxes. One command file is used to automate the twice monthly backup, not just for one directory, but for all five global data directories. Each global data directory has a separate disk for these backups.

The following command file displays the space required for each directory backup, the space remaining on each disk, loads the disks on to shelves, and then performs the backups unattended:

```
*
wd /jukebox
*
* Load /DATA Backup Disk
*
ask 'Load /DATA disk, Side A up, Ready (No to skip)?YN'
      if the /DATA directory is to be backed up, the
      current /DATA backup disk is placed in the
      import/export slot of the jukebox
if is $Return3 = 0
      if the answer is Y,
then acmove lu:26 fr:ie to:d2
      move the disk from the i/e slot to drive 1
ask 'Need to know size of /DATA directory?YN'
      ask if the size of the directory needs to be known
if is $Return3 = 0
      if the answer is Y,
then fst tr size.data
      use FST to show the size
fi
```

```

wait,2
    wait 2 seconds for the disk to be ready
frees 28 +m
    show the available space
ask 'Enough space on Side A?YN'
    ask if the space is sufficient
if is $Return3 = 0
    if the answer is Y,
then acmove lu:26 fr:d2 to:s21
    move the disk from drive 2 to shelf 21
set data = yes
    set variable $DATA to YES
else acmove lu:26 fr:d2 to:d2 i
    if the answer is N, switch to Side B
wait,5
    wait 5 seconds for the disk to be ready
frees 28 +m
    show the available space
ask 'Enough space on Side B?YN'
    ask if the space is sufficient
if is $Return3 = 0
    if the answer is Y,
then acmove lu:26 fr:d2 to:s21
    move the disk to shelf 21
set data = yes
    set variable $DATA to YES
else acmove lu:26 fr:d2 to:ie
    move the disk to the i/e slot
ask 'Load blank disk, Side A up, Ready (No to skip)?YN'
    ask user to remove disk and load blank disk
if is $Return3 = 0
    if the answer is Y,
then acmove lu:26 fr:ie to:s21
    move the disk to shelf 21
set data = yes
    set variable $DATA to YES
fi
fi
fi
fi

```

this completes the loading of a disk for the /DATA directory to shelf 21 if a backup of /DATA is to be performed - \$DATA = YES

This is repeated for the other four global directories such that if the backup of that directory is to be performed, a disk is loaded on the appropriate shelf with the side to receive data oriented up and the corresponding variable is set to YES.

Directory	Shelf	Variable
/DATA	21	\$DATA
/NEU	22	\$NEU
/COG	23	\$COG
/PHA	24	\$PHA
/QC	25	\$QC

Then the backups are performed unattended:

```
*
* copy data
*
if is $data = yes
then
acmove lu:26 fr:s21 to:d2          move the disk from shelf 21 to drive 2
wait,5                             wait 5 seconds for the disk to be ready
/tape/databackup.run              perform the backup of /DATA
acmove lu:26 fr:d2 to:s21         move the disk back to shelf 21
fi
if is $neu = yes
then
acmove lu:26 fr:s22 to:d2        move the disk from shelf 22 to drive 2
wait,5                             wait 5 seconds for the disk to be ready
```

```

/tape/neubackup.run
    perform the backup of /NEU
acmove lu:26 fr:d2 to:s22
    move the disk back to shelf 22
fi
fi
if is $cog = yes
then
acmove lu:26 fr:s23 to:d2
    move the disk from shelf 23 to drive 2
wait,5
    wait 5 seconds for the disk to be ready
/tape/cogbackup.run
    perform the backup of /COG
acmove lu:26 fr:d2 to:s23
    move the disk back to shelf 23
fi
fi
if is $pha = yes
then
acmove lu:26 fr:s24 to:d2
    move the disk from shelf 24 to drive 2
wait,5
    wait 5 seconds for the disk to be ready
/tape/phabackup.run
    perform the backup of /PHA
acmove lu:26 fr:d2 to:s24
    move the disk back to shelf 24
fi
fi
if is $qc = yes
then
acmove lu:26 fr:s25 to:d2
    move the disk from shelf 25 to drive 2
wait,5
    wait 5 seconds for the disk to be ready
/tape/qcbackup.run
    perform the backup of /QC
acmove lu:26 fr:d2 to:s25
    move the disk back to shelf 25

```

restoring them to their normal directories if they need to update the files. This would allow users to access vast numbers of old data files without requesting a system manager to restore files and would reduce the amount of hard disk space required for restoring those old files.

Conclusions

These applications are specific for our laboratories but they demonstrate some of the ways that HP optical disk libraries can be very useful attached to HP 1000 computers. There is some bad news however, HP does not currently support HP optical disk libraries attached to HP 1000 computers. Regardless, those users who have HP 1000 systems that require quick access to large amounts of data should consider the many of advantages of optical disk libraries, especially now that they are inexpensive and available in several sizes, from one drive, 16 shelf versions all the way up to a 4 drive, 144 shelf version. During the last several years HP has brought down the cost of both the libraries and the optical disks. Disks have decreased from about \$200 to just over \$100 each while doubling their capacity. The cost of a 20 Gb jukebox has decreased from about \$42,000 (for a unit with 32 shelves and two single density "regular" performance drives) to under \$7,000 (for a unit with 16 shelves and one double density high performance drive). Even though a jukebox costs more than stand alone optical disk drives, the ability to quickly access vast amounts of data unattended more than justifies the additional cost.

FLOATING-POINT MANAGEMENT IN THE HP-RT KERNEL

Ren Wang and George Anzinger

Hewlett-Packard Company

11000 Wolfe Road

Cupertino, CA 95014

408-447-5125

HP-RT is the real-time operating system for Hewlett-Packard's Precision Architecture RISC computers. This paper discusses why and how the floating-point management is done in the HP-RT operating system, and what the performance gain is.

The Floating-point Coprocessor

The PA-RISC floating-point coprocessor is an assist processor of the central processor unit (CPU) in PA-RISC computer systems. The coprocessor has its own registers and instruction set, which are independent of the CPU's registers and instruction set. The coprocessor has its own internal state and hardware evaluation mechanism. The coprocessor executes floating-point instructions to perform arithmetic on its registers, and to move data between the registers and memory. The CPU and the floating-point coprocessor may be operating on a number of instructions simultaneously.

What Can the Floating-point Coprocessor Do For The HP-RT Kernel?

Since the floating-point coprocessor can pass double-word quantities to and from memory, it is particularly useful for transferring large quantities of data. If we use general registers to do memory copies, each instruction can only transfer a one-word quantity. But if we use the floating-point registers, each instruction can transfer a double-word quantity. The fast memory copy can work almost twice as fast by using floating-point registers.

In the HP-RT kernel, memory moves are frequently performed, some in large quantities. When a child process is forked, many properties of the parent process are copied to the child process. The larger the parent, the more time consuming

the inherent process gets. Memory copy is used in the following operations: asynchronous I/O processing, reading/writing from/to a block device, execing (loading an executable file onto the current process, replacing the current program), duplicating a copy of a current process's memory block to another process, writing a core file, sending/receiving messages, etcetera. Drivers in HP-RT may also use memory copies.

What Needed To Be Done To Make Better Use Of The Coprocessor?

All user-level and system level processes executing floating-point instructions share the same coprocessor. The ownership of the coprocessor is switched frequently from one process to another. Coprocessor state must also be managed in changes from user mode to kernel mode within the same process.

There could also be multiple levels of coprocessor usage for the same process. For instance, a process runs in user mode. It has floating-point calculations in its user level code. This is the first usage of the coprocessor. When a signal is delivered to the process, the signal is initially handled in the context of the process in kernel mode. During the signal delivery setup, the floating-point coprocessor is used to do a data copy. This is the second usage of the coprocessor. The user signal handler may use the floating-point coprocessor. This is the third usage of the coprocessor. In this case, there are two floating-point context switches for the same process, and three floating-point context save areas are allocated. The first coprocessor context switch happens when the kernel executing on behalf of the process, uses the floating-point coprocessor for signal delivery setup. The second coprocessor context switch happened when floating-point operations are invoked in the user signal handler.

When the user code executes its floating-point instructions, a save area is allocated for the process. Nothing will be stored to the first level save area until the coprocessor is used again when the process switches to the kernel mode to setup the signal delivery. Only then, the context of the coprocessor at the first level is stored to the save area (level-1 save area). The context of the coprocessor is its ownership, the state of the coprocessor configuration register (CCR), and the values of its status register, exception registers, and floating-point registers.

Another save area (level-2 save area) is allocated when the coprocessor is used inside the signal delivery setup code in kernel mode. The context of the coprocessor is stored to level-2 save area when the user signal handler requests for the coprocessor. And the level-3 save area is allocated for the third level of the coprocessor usage in the user signal handler.

When the user signal handler finishes its execution, the level-3 save area is released. The level-2 and level-1 save area will be released when the user code

completes its execution. On the exit of the process, all the floating-point save areas will be released.

If another process (process B) preempts the current process (process A), and requests for the coprocessor, then the context of the coprocessor will be saved to the save area of process A at the level when process A is preempted. If process B already has a save area, then the context in the save area will be loaded onto the coprocessor. The coprocessor will execute in process B's coprocessor context.

If process A later preempts process B, the coprocessor context will be saved onto the save area on process B at the appropriate level. The coprocessor will be loaded with the context from process A's save area at the appropriate level.

Without multiple level floating-point context management, the context of the coprocessor at each level will be lost when a newer level of coprocessor usage is encountered.

What Are The Options In Handling The Coprocessor Context Switch ?

A simple way to address this problem is to context switch the coprocessor whenever there is a context switch for any process. Conventional operating systems take this approach. A process executes in the context of its state, its text, the values of its global user variables and data structures, the values of the CPU registers it uses, its user and kernel stack contents, etcetera. When the kernel decides that it is time to execute the other process, a context switch is done. The context of the coprocessor could just be part of the process context, and be saved/restored at the time of the context switch.

What is the cost of including floating-point saving/restoring in the process switch? To store/load 32 double coprocessor registers takes 72 instructions. The process context switch takes just about the same number of the instructions. So if we make the store/load of the coprocessor registers part of the process context switch, we would double the context switch time in the HP-RT operating system. For a real-time operating system, context switch time must be kept as short as possible, since it is one of the components of the system response time.

Another way to address this problem is to context switch the coprocessor only when the going-away process has the ownership of the coprocessor, and the new process needs to use the coprocessor, or when there are multiple nested usages of the coprocessor. Depending on the application program, the coprocessor may or may not be used. We could save process context switch time if we do the coprocessor context switch on demand. Furthermore, since each process does not always need to save/restore coprocessor context, we can also allocate save areas for the process on demand.

What is the cost of the second option? It adds more complexity to the coprocessor management. But the overhead of the coprocessor management is much less than the overhead of a large context switch in HP-RT. So we decided to take the context switch on demand approach, minimizing the coprocessor context switch overhead while still providing for coprocessor usage in both user and kernel mode.

How Is The Floating-point Management Implemented in the HP-RT Kernel?

We will discuss the floating-point management data structures, a flowchart of the assist emulation trap routine, and some pseudo code of in function fork().

Before we explain the implementation of the floating-point support in the HP-RT kernel, let us clarify the relationship of processes and threads. Each process has an initial thread. Depending on the application, there are single thread processes (just the initial thread) and/or multithread processes. A process owns one or more threads. Tasks are actually ran by threads. Each thread has its own unique machine state. Hence, floating-point save areas are threads specific, not process specific. So only the thread structures have save areas for floating-point management.

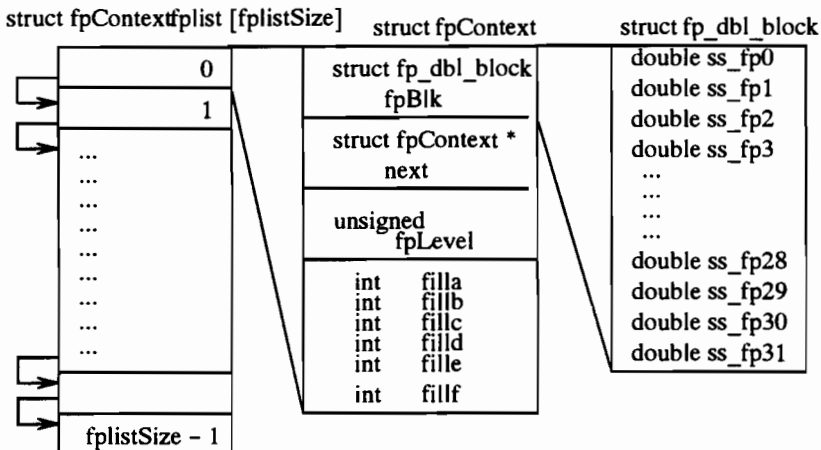
Data Structures

A free list of the floating-point save areas is allocated at the kernel initialization time. The total size of save areas is the maximum number of the threads in the system. The total number of floating-point save areas is fixed at system generation time. The free list is not dynamically allocated. This is done for the following reasons:

- * In general, the default size of the fp free list is adequate (according to the high water mark we see in the fpLevel after many tests).
- * Since the Q-bit in Processor Status Word(PSW) is off when the most of the floating-point management code is invoked, the interrupt state is not collected. If we want to dynamically allocate a floating-point save areas, we would need to move the shadow registers onto the kernel stack, turn interrupts on, then call a kernel function to get the memory. The overhead is high for these operations. There is also the risk of memory exhaustion, causing thread delay.

Each floating-point save areas has a structure of fpContext. Each thread structure has a pointer to the next floating-point structure (fpContext). The initial structure of the free list of fpContext structures, and the internals of the fpContext structures is as follows:

The Free List of Floating-point save areas



Initially, the value of the pointer in the fpContext structure for each thread structure is set to null. Only when the thread uses the coprocessor will a structure be acquired from the fp free list, and the fp pointer will then be pointing to that structure.

Note that there are 6 integers filla though fillf in the fpContext structure. They are used to align the fpContext data structure to a cache line boundary (8 words on PCX-L or PCX-T processors), so we can take advantage of cache hint when we save/restore floating-point context. Without cache hints, if a store misses, the whole cache line is read in. Then the write (store) is done on the line. With cache hints, the read will not occur, since the whole cache line is going to be replaced, so only the write(store) will occur on that cache line. This improves performance.

Initially, fpLevel in all the fpContext structures in the free list is set to -1. The fpLevel will be altered if it has been used. By checking the fpLevel on the free list, we know how many floating point save areas are actually used. A user can increment or decrement the size of the floating-point free list by setting ADDFPS(Additional Floating-point Structures) to a negative or positive number in param.h (default value for ADDFPS is 0).

The coprocessor configuration register (CCR) is an 8-bit control register, CR10. It is used to indicate the presence and usability of a hardware implementation of a coprocessor. Bits 0 and 1 in the CCR correspond to the floating-point coprocessor. When both bits are set to 1, floating-point instructions are passed to the co-

processor and the defined operation occurs. When both bits are set to 0, any floating-point instruction will cause an assistant emulation trap.

A note about fpLevel

fpLevel is used to keep track of the levels of floating-point coprocessor usages in one thread. Initially, the fpLevel of each thread structure is set to 0. FpLevel is incremented on a trap entry or an interrupt. FpLevel is restored whenever we return from kernel stack or interrupt control stack (ICS) to the user. In the setjmp()-longjmp() pair, fpLevel is saved during setjmp, and restored either in longjmp() or in sig_rewind() (in the case of a longjmp(), this is done from a user signal handler). All of the floating point structures allocated between the current fpLevel and the saved fpLevel is released and put back onto the free list, then the fplevel is restored to the level before the setjmp() is called.

Note that when a user makes a system call, and the process is switched from user mode to kernel mode, there is no floating-point context switch or level change. Since the system call is a procedure call, the C compiler generates code to save the caller_saves registers (fr8 to fr11 and fr22 to fr31 for PA-RISC 1.1 processors). For details, see Chapter 3 "Register Usage and Parameter Passing" in "PA-RISC Procedure Calling Conventions Reference Manual". The HP-RT kernel defers to the C compiler to manage the saving/restoring of the floating-point processor in these calls.

A note about Who_has_fp

Who_has_fp records the ownership of the floating-point coprocessor. It is initialized to NULL. When any thread acquires the coprocessor, Who_has_fp is set to the thread pointer. When the thread releases the coprocessor, Who_has_fp is set to NULL.

A note about ccr in the thread structure

There is a ccr in each thread structure. When a thread is context switched out, the CCR value is saved onto the thread->ccr. When a thread is context switched in, CCR is loaded with the thread's ccr. These are done in resched().

Here is an example of how thread->ccr is used:

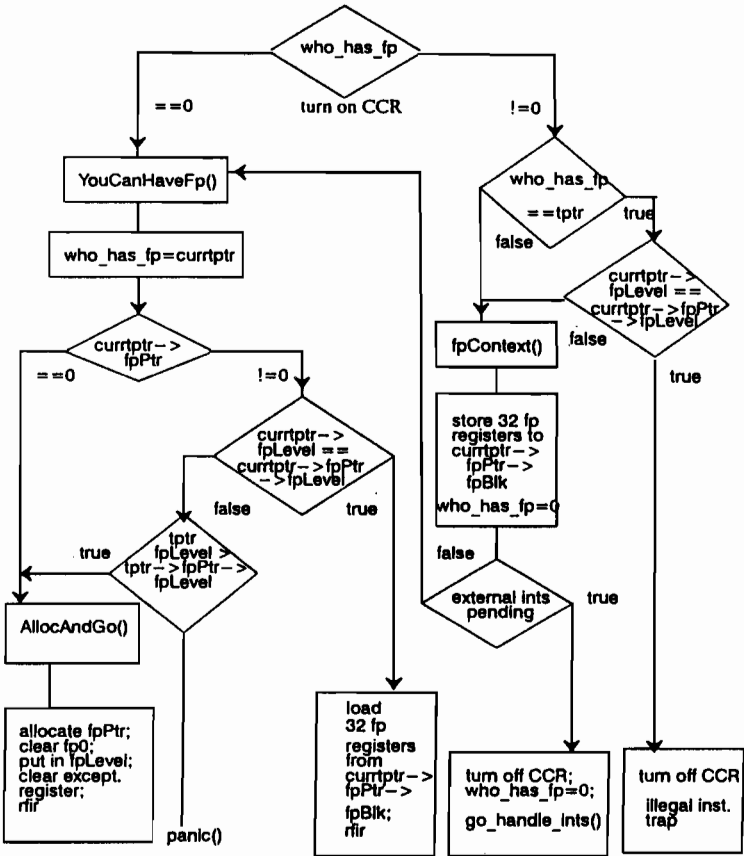
Thread A owns the coprocessor. So Who_has_fp is A, CCR is turned on. Thread A then is context switched out. CCR's value (0xC0) is first saved onto A->ccr. Then CCR is loaded with the value of B->ccr. Thread B gets to execute. Thread B does not need to use the coprocessor. When thread A is back to the execution, CCR will be loaded with A->ccr (0xC0). Note that thread A does not have to take an assist emulation trap since CCR is turned on by using the saved value.

By using thread->ccr, the coprocessor does not have to be turned off/on between process context switches if one of them does not use the coprocessor.

Flowchart of the Assist Emulation Trap

The assist emulation trap handler is the core of the floating-point management inside the HP-RT kernel. The code is run with the Q-bit off. The Q-bit is the interrupt state collection enable bit in Processor Status Word (PSW). When it is set to 0, no interrupt state is collected. The code is running under the shadow register environment: interrupt is disabled, virtual memory system is off (both code and data are addressed with physical addresses).

The following is the flowchart:



The allocation of the floating-point save areas and saving/restoring of fp register contents

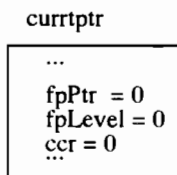
Most threads' pointers to the floating-point save areas (fpPtr) are null at the time of thread creation. However, fpContext will be allocated and initialized for the initial thread of a process if its parent has any fpContext.

When the first floating-point instruction in either user code or kernel code is executed, the process will take an assist emulation trap since the coprocessor configuration register is 0. (Assist emulation trap happens when an attempt is being made to execute a coprocessor instruction for a coprocessor whose corresponding bit in the Coprocessor Configuration Register is 0). In the assist emulation trap handler, who_has_fp shows nobody owns the floating point coprocessor. So YouCanHaveFp() will be executed where who_has_fp will be updated to the current thread pointer. If the floating-point save area has not been allocated for the thread, AllocAndGo() will allocate and initialize the save area. Then we will restart execution of the interrupted floating-point instruction.

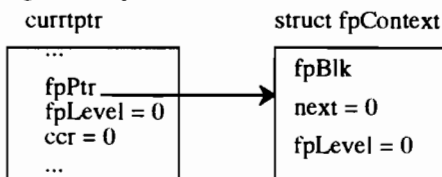
When the thread finishes the floating-point instruction execution. There are a few situations that can happen. We will discuss some of them here:

A. The thread finishes its task and exits. All the floating-point save areas for the thread will be released, both who_has_fp and CCR will be reset to 0. No thread owns the coprocessor and the next thread who executes any floating-point instruction will take an assist emulation trap.

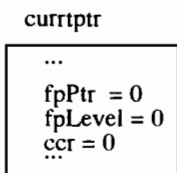
- 1) No floating-point structure is allocated yet.



- 2) A floating-point structure is allocated for the thread, preparing for a coprocessor context switch.

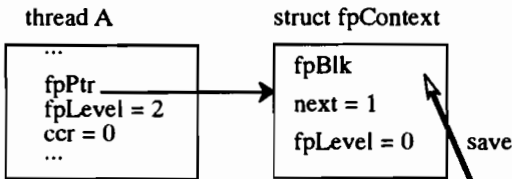


- 3) The thread only had one level of floating-point execution, then exited, so the floating-point structure is released.

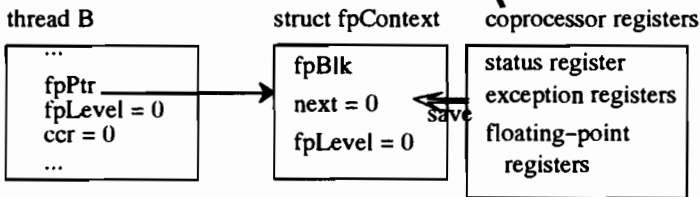


B. The thread (thread A) is preempted by some other thread (thread B). During a process context switch, CCR is loaded from the ccr of the thread structure. The thread ccr is initialized to 0 on all threads. When thread B attempts to use the coprocessor, it will take an assist emulation trap since CCR is 0. It will find that who_has_fp is set to thread A. That means a floating-point context switch needs to be done. The fpContext() routine saves all the floating-point registers to the fpPtr of thread A, and clears A->ccr. Then YouCanHaveFp() is called to set up the floating-point structure for thread B. Later, thread A gets to run again, and it needs to use the coprocessor. It will do a floating-point context switch from thread B to thread A. The coprocessor context will be saved onto the save area in thread B, then the contents of fpContext on thread A are loaded into the status register, exception registers, and floating-point registers of the coprocessor. who_has_fp is set to thread A.

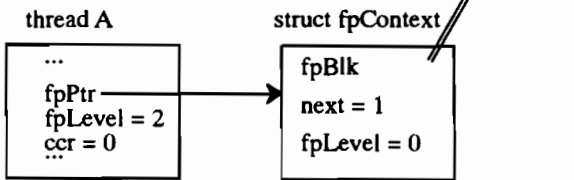
- 1) fpPtr is allocated for thread A. The coprocessor context is saved into the fpPtr on thread A.



- 2) fpPtr is allocated for thread B.



- 3) The coprocessor's registers are loaded with the context saved on fpPtr of the thread A.



C. The thread sets up a user defined signal handler, and there are floating-point operations inside the user signal handler. A signal of that type has been sent to the threads.

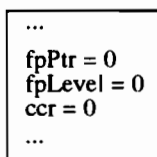
During the process of delivering the signal, the kernel uses the coprocessor. So the fpLevel of the thread is incremented by 1, and a new save area is allocated for that level.

When executing the user defined signal handler, another level of the floating-point save area is allocated.

Here is an illustration of what happened in the above example:

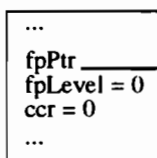
- 1) No floating-point structure has been allocated yet.

currtptr

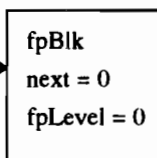


- 2) The thread executes the first floating-point instruction in user mode. An assist emulation trap is taken and the first level fpContext is allocated.

currtptr

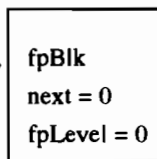
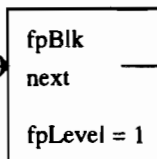
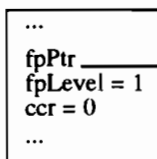


struct fpContext

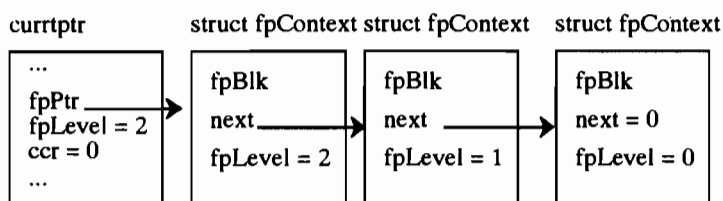


- 3) During the process of delivering the signal, fpLevel is incremented by 1. Another fpContext is allocated.

currtptr



4) The third level of the save area is allocated when executing the floating-point instructions in the user signal handler.



The release of the floating-point save areas

Release of floating-point save areas happens in a few different cases:

- * In setjmp()-longjmp() pair, all the floating-point save areas allocated between the setjmp()-longjmp() will be released at the time of longjmp().
- * On the trap exit, if the level of the current floating-point save areas is the same as the level of the current thread structure, the current floating-point save areas will be released.
- * If a user signal handler calls longjmp(), then all the floating-point save areas allocated between the setjmp() call before executing the user signal handler and the longjmp() will be released at the time of longjmp().

In execve(), exit(), and st_exit(), all the floating-point save areas on the exiting thread are released.

The following illustrates the release of the floating-point save areas for example C.

entries will slow down the system performance, as we will need to wait for process/thread to release a fpContext structure back to the free list.

A note about fork()

Special care has to be taken since the floating-point coprocessor is used in fork() for the floating-point management. The function fcopy() uses floating-point registers, and it is used to copy the fpPtr list of the parent process onto the child process. If the parent process has any fpContext allocated, then the child process will have the same levels of the fpContext allocated. The context of each parents' fpContext will be copied onto the child's fpContext at each fpLevel. Here is the code for the floating-point management code in fork():

```
/* allocate and copy parent's floating-point structures onto the child process. Because the most recent fpContext is at the head of the list, we allocate and copy the last fpContext on the list first, and the first fpContext on the list last. That way, we will copy the complete list. */
```

```
childPtr->fpPtr = Lp = 0;
```

```
disable interrupt;
```

```
while (Lp != (fltPtr = currtptr->fpPtr)) {  
    while (fltPtr->next != Lp)  
        fltPtr = fltPtr->next;  
    newFpPtr = 0; /* start to allocate fp structures */  
    while (newFpPtr == 0) {  
        if (newFpPtr = fpHead) {  
            fpHead = fpHead->next;  
            newFpPtr->next = pptr->fpPtr;  
            newFpPtr->fpLevel = fltPtr->fpLevel;  
            pptr->fpPtr = newFpPtr;  
            fcopy(&newFpPtr->fpBlk, &fltPtr->fpBlk, sizeof(struct  
fp_dbl_block));  
        }  
        else { /* Ran out of the free list, nobody releases any fp structure.  
            User need to used a larger ADDFPS in param.h */  
            inform the use that free list for fp structure has ran out;  
            set errno = EAGAIN;  
            return(SYSERR);  
        }  
    }  
}
```

```

    }
  }
  Lp = fltPtr; }

```

turn interrupt back on;

Performance gain

Some measurements were taken to see the performance gain in fork() after we implemented floating-point. Depends on the size of the data structure which is copied from the parent process to the child process, the performance gain varies from 15.05% to 15.86%.

The measurements were taken on a HP 9000 model 742rt system, with 64 megabytes physical memory. Both the parent and child processes perform floating point calculations, so the measurements include the overhead introduced by the floating-point context switch save/restore time.

size(megabytes) time (usec)	6	4	2	1	0.5
w/ fp in kernel	1302943	871394	435869	219361	111248
w/o fp in kernel	1548586	1035267	517575	260174	130949
performance gain	15.86%	15.82%	15.79%	15.69%	15.05%

Paper Number 7006
Client/Server Process Control Over a WAN
Kevin B. Wong
East Bay Municipal Utility District
375 11th St.
Oakland, CA 94607
(510) 287-1166

Abstract

Client/Server applications over a LAN is tough enough. What do you have to watch out for when you deploy a Client/Server application over a WAN?

This paper covers how EBMUD extended its HP1000 based SCADA/process control system with the addition of client/server based "CIM/21" software on HP9000/700 series workstations.

WAN upgrades, loading, back-up, and client/server implications across the low speed WAN links will be discussed. We will briefly review the features of CIM/21 from Industrial Systems, Inc. (ISI), and focus in on client/server safeguards of the entire process control system.

The original SCADA system, called the OP/NET System, consist of paired HP1000's at five Area Control Centers (ACC) linked to a pair of HP1000's at the Oakland Control Center (OCC). The ACCs each poll and control 40-60 Remote Terminal Units/PLCs located throughout the service area. The OP/NET System currently monitors/controls about 8400 points spread around 300 facilities.

Introduction

East Bay Municipal Utility District (EBMUD), based in Oakland, California, treats and distributes an average of 220 MGD of water to 1.2 million people spread over 325 square miles in Alameda and Contra Costa counties. The OP/NET System monitors and controls:

- 175 reservoirs (almost 1 billion gallons of treated water storage),
- 135 pumping plants,
- 24 rate control valves, and
- 6 filter plants

in the water distribution service area.

The OP/NET System also monitors five weather stations in the 575 square mile Mokelumne watershed in the Sierras and also controls:

- 2 hydroelectric plants with 3 turbines each,
- 2 chemical feed plants and 6 safety release stations along the 3 aqueducts between Walnut Creek and Campo Seco.

The original OP/NET System, which is based on Fisher Controls' GV1000 SCADA software has been fully operational since January 1989.

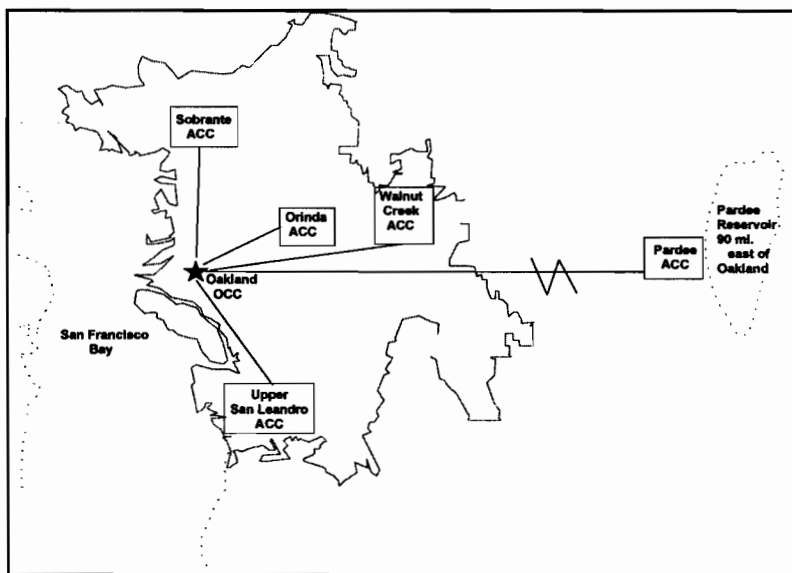


Diagram 1 Geographic layout of OP/NET ACCs to OCC

Background: Original System Layout

The Area Control Centers have changed very little over the past six years even though 35 new Remote Terminal Units (RTUs) were added and the I/O point count increased from 6100 up to 8400. A fourth mux along with PCIF and Fisher code was added to each of the 4 ACCs in the East Bay to accommodate monitoring and control of Programmable Logic Controllers (PLC) in the treatment plants.

The five ACCs all have very similar configurations. HP1000-A900 series computers are paired for redundancy and drive the following peripherals:

Paper 7006 - Client/Server Process Control Over a WAN

- 2 operator color workstations (2397 compatible),
- 2 printers, 1 plotter, and a tape drive,
- 10-12 modems, and
- a mirrored pair of 571 MB disk drives.

The A900 computer boxes include:

- 6 MB of RAM,
- 4 8-channel serial MUX cards,
- 1 DS/1000 HDLC network card (9600 Baud),
- 1 DI/DO card for watch dog time out switch over,
- 1 Parallel Interface Card for CPU to CPU handshaking,
- 1 Async card for system terminal, and
- 4 HPIB cards to talk to disc drives, tape, plotters, and clock.

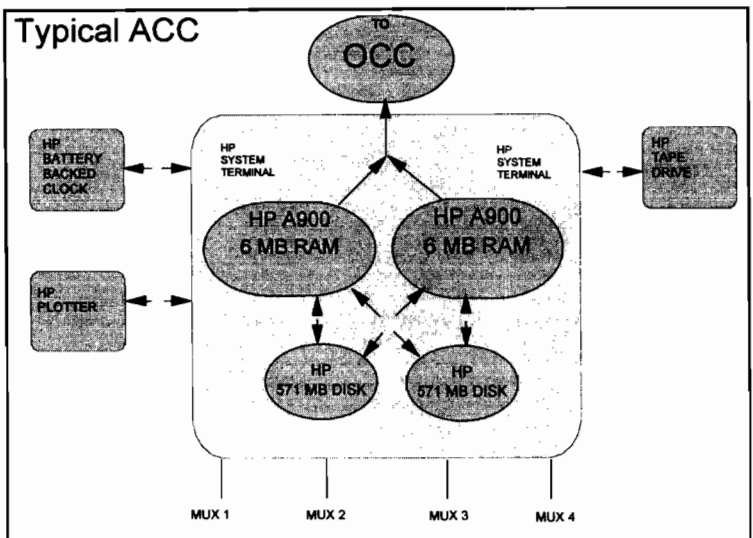


Diagram 2 Typical Area Control Center layout

The ACCs are networked to the OCC in a star point to point configuration.

The ACCs can function -- "stand alone" -- if the OCC or associated communications links are down. The ACCs monitor and control 40-60 RTUs in its geographic area and passes data up to the OCC. The OCC can pass down commands and pump control schedules to the ACCs.

The OCC HP1000-A990 (A900s upgraded to A990s in December 92) boxes include and or share:

- 12 MB of RAM,
- share 6 571 Mbyte disks,
- drive 10 operator stations,
- 5 DS1000 HDLC cards,
- share 4 printers, 1 plotter, and 2 tape drives,
- 3 8 channel serial mux cards
- 9 HPIB cards, 1 async, 1 DI/DO, and 1 parallel interface card.

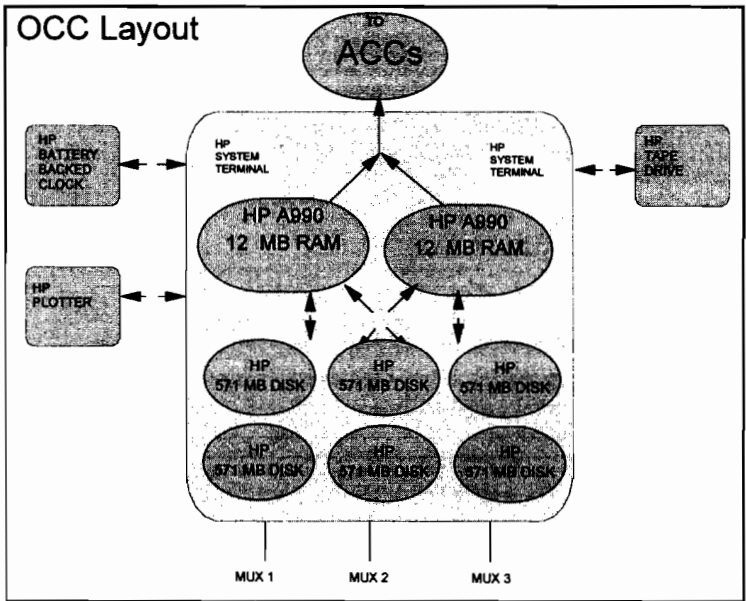


Diagram 3 Oakland Control Center - HP1000-A990 arrangement

The OCC has limited external access where a user could dial in and view OP/NET screens with a Vectra 286 and special video hardware.

The GV1000 provides comprehensive SCADA control with RTU communications (multiple types and protocols), real time process display graphics, trends, alarm management, I/O database management, logs, process calculations, and pump control.

The Lay of the WAN

ACC to OCC DS1000 network communications were originally set up with 9600 baud synchronous modems through EBMUD's microwave communications systems. Parallel analog leased lines were added to provide a redundant communications path. These analog based lines were upgraded to 56 kbps digital circuits in mid 1991 when the OCC was relocated to the -- then new -- administration center. The monthly recurring cost only increased about 15 percent for this upgrade. DSU/CSU's with a 4-channel synchronous serial multiplexer were installed with the DS1000 using one channel at 9600 baud. We would essentially only be utilizing 17 percent of the bandwidth over the next two years, but the added reliability of the digital circuits has been well worth the added expense. 9600 baud has proved to be very adequate for communications between the ACCs and OCC. Exception data typically transmits short bursts every six to 15 seconds and the occasional file transfer takes a few minutes.

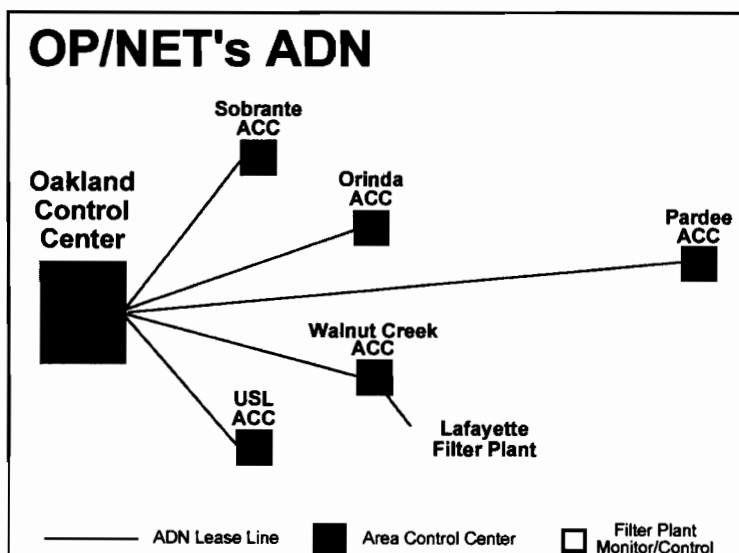


Diagram 4 OP/NET's Advanced Digital Network (56kbps links)

In actuality, the HP1000 with GV1000 software operates like a "closed system" client/server architecture with distributed processing over a proprietary network.

The ACCs take care of RTU communications, maintain a local historical archive, and can operate completely standing alone without the OCC. The OCC also does parallel processing on the "exception" data coming from the ACCs. Each ACC can only monitor/control the RTUs/PLCs that it polls and controls.

The OCC with the original A900s' response time was growing slower and slower due to CPU loading. The A990s brought the CPU loading back down to a level of reasonable performance. I/O point growth was constrained before the A990 upgrade.

The Changing WAN

The OP/NET System Capacity Improvements (OSCI) was approved in December 1992. The four components consisted of:

1. WAN equipment, training, installation, and testing;
2. HP9000 755, 715/50 workstations, X terminals, and associated peripherals and LAN equipment;
3. CIM/21 software and ISI integration services and ISI integration services and support; and
4. HP Unix training and HW/SW support.

OP/NET engineers directed and provided overall integration for OSCI.

The existing 56 kbps digital leased lines were beefed up with MICOM NetRunners to handle the existing DS1000 9600 baud links, bridge ethernet, and add voice (telephone hot lines) between the OCC and ACCs. The original four channel MUXes were bypassed and the full 56 k was piped through the NetRunners to compress the voice and data and use fast packet cellular switching. In other words, the 9600 baud links no longer uses Time Division Multiplexing, but only utilizes bandwidth during data burst and is squeezed about 50% with RLL compression. The voice channel utilizes between 4.8 to 16 kbps during usage and adjusts dynamically and will use the full 16kbps if available. Three kbps is utilized for management overhead at 56 kbps. The ethernet bridge typically provides about 100 kbps with compression. Throughput utilization went from 17% to about 200%. About 10-12 kbytes/second of ethernet data can pass. The NetRunners were also tested at 9600 baud and will still pass 2-3 kbytes/second of data with no telephone hot line usage and the DS1000 data still pulsing between the ACC and OCC.

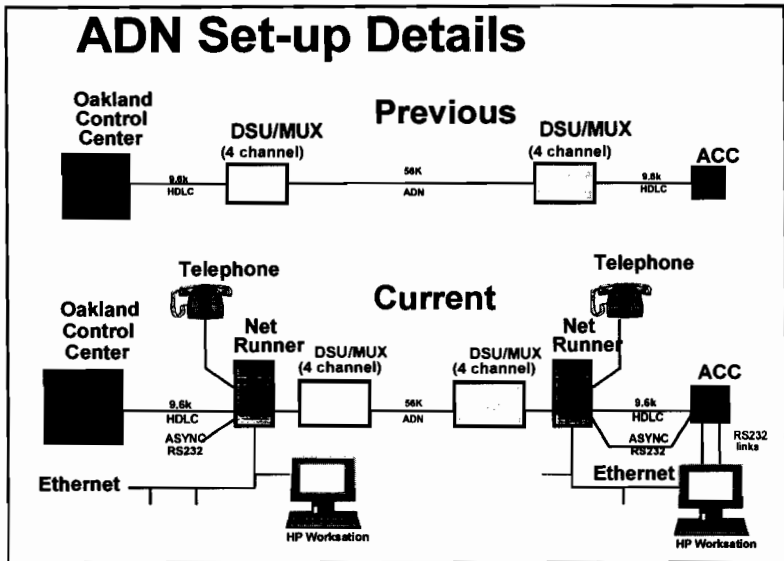


Diagram 5 OP/NET Wide Area Network upgrade

Added Bonus

One HP9000-715/50 workstation with a 1GB disk, DAT, 32 MB RAM, and 19" color monitor was deployed to each ACC. Both serial ports were connected to the HP1000 MUX ports. One serial port was dedicated to passing GV1000 OP/NET data to the CIM/21 scan module in the workstation. The second serial port is available as a "Kermit" terminal connection to the HP1000. We can now telnet out to any ACC workstation across the WAN and establish a "local" terminal session with the HP1000 via Kermit. Everything works great except EDIT/1000 only works in line mode. These "local" terminal sessions have greatly enhanced Fisher and our ability to troubleshoot and support the GV/HP1000 without the telnet CPU load on the HP1000. Instead of FTP, we use Kermit or DS copy to transfer files around. A modem has always been available on the OCC HP1000. I have actually had a sessions to all 5 ACCs opened at once on a single workstation. X-windows is great!

CIM/21

The OSCI project originally had 2 primary goals:

1. Offload the OCC HP1000-A990s to accommodate future I/O point

- growth and extend the life of the original system;
2. Make OP/NET System data more accessible to other EBMUD personnel such as planners, maintenance support, engineers, and management

ISI's theme for CIM/21 is "Open Access to Process Information." CIM/21 is an "Open Systems" client/server process data archive with extensive distributed processing. CIM/21 has many parallels to GV1000. Real time process display graphics can have embedded trends. Trending and data archive capabilities are much more extensive, having been built around a high performance archive compression engine. CIM/21 also adds SQL-Oracle (or Sybase) integration with the ability to push or pull data directly to or from the RDBMS. CIM/21 also adds more analytical tools such as SQC statistics and a spreadsheet. The typical open systems interoperability HP/UX includes such as X-windows, TCP/IP, Networking (Ethernet), Client/Server Tools, Telnet, and FTP are fully supported.

CIM/21 is very modular and is designed for distributed processing with minimal network loading. Each ACC HP9000 715/50 workstation has a scan module and graphics module.

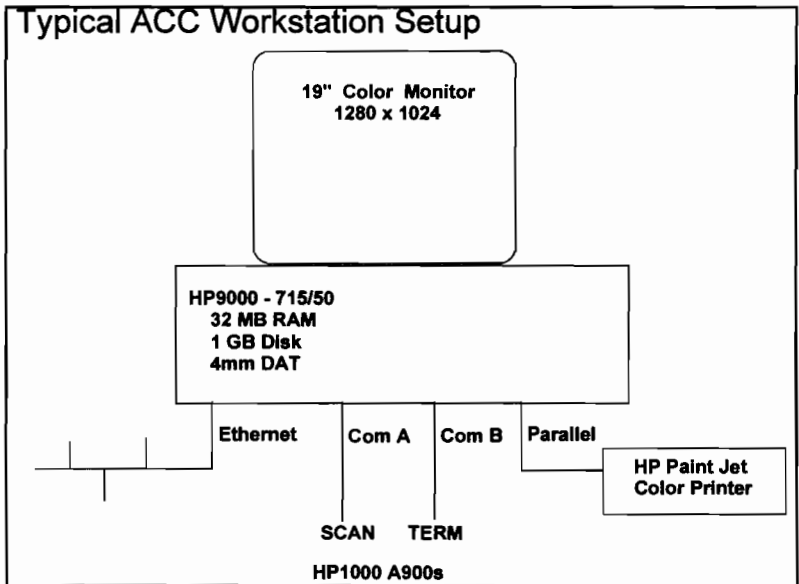


Diagram 6 CIM/21 Area Control Center Workstation Layout

ISI developed the communications "scan" software modules for the HP1000/GV1000 side and HP9000 side. CIM/21 scans the real-time process image of GV1000 every ten seconds (scan rate configurable). Any process values or point status values that have changed from the previous scan are passed across the 9600 baud serial link to the HP9000-715 workstation. The workstation scan modules forwards these "exceptions" across the WAN to the archive server.

Each ACC monitors/controls 1500~1900 I/O points. Typically, about 50 to 90 values change every 10 seconds. The serial link usually takes a few seconds to transfer the changes to the HP9000 scanner and the WAN link takes a fraction of a second to forward the data to the archive server in Oakland.

The Graphics module in each workstation utilizes a local disk copy of "process graphics", "tag lists", and menus. Process graphics and tag lists typically update every five seconds with process information. Thus, only process data is fed across the WAN link when a user fringes up a process graphic or tag list. The user is now able to access data from the entire system now whereas before, GV1000 would only show data collected locally at that ACC. The graphics module also support X11r5 and thus X-terminals can access a workstation.

In fact, the graphics server, HP9000-755 workstation with a 2GB disk, DAT, 128 MB RAM in the OCC, drives many X-terminals, PCs running X-emulation in the Oakland headquarters and many remote sites (non-ACC). The graphics server minimizes the X-terminal load on the archive server typically to one or zero users and also serves as a "cold" redundant standby server which can easily be switched over to act as the archive server. Just move the SCSI cable from the 5GB disk array over and run a short script. The HP9000-715/50 on my desk also serves as a "cold" redundant standby for the ACC workstations.

WAN/LAN Disruptions

CIM/21 has backup buffer mechanisms to buffer scan data in the event of a LAN or WAN failure. Scan data in the HP9000-715 scan modes will automatically buffer to disk with time stamps already attached when the WAN or LAN link goes down. The buffer will automatically flush to the archive server once the link is restored. The buffer size is configurable and should currently handle over two days of process data changes.

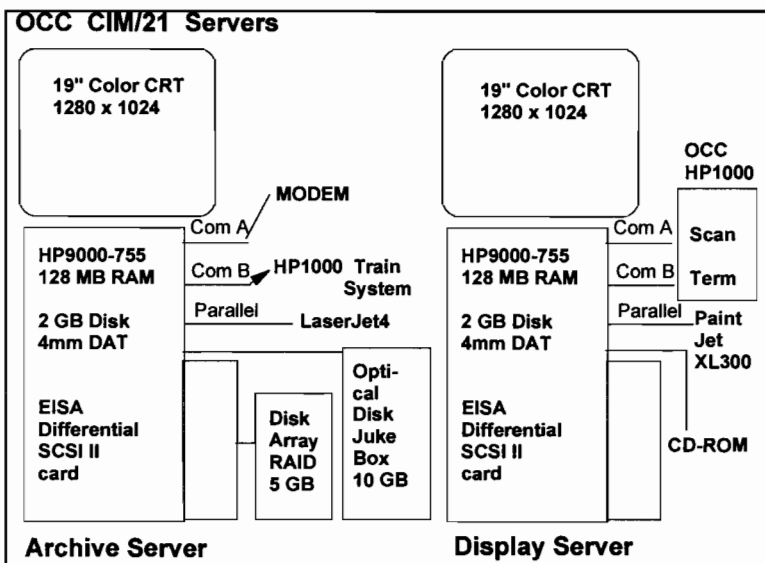


Diagram 7 CIM/21 Server Layout at Oakland Control Center

The HP1000 scan module also has a local buffer in the event the workstation, CIM/21, or the serial link is down. This buffer will automatically flush once the HP9000 scan module starts requesting data again. This buffer is sized to hold about four hours of data.

In the event of a 56kb leased line failure, the NetRunners can be moved over through the 9600 baud synchronous modems on the microwave system if need be. We also maintain a spare NetRunner in our stock. Alternative plans for higher speed ethernet WAN links through the microwave system are under consideration.

Conclusion

Mechanisms, such as local buffering are essential for client/server and distributed processing to minimize data loss when the WAN, workstations, software, or network equipment fails.

CIM/21 and the OSCI project have met its two primary goals and operates very reliably and robustly with only partial redundancy. The HP1000s with GV1000 will still be in full use for at least the next four or five years and the HP9000 workstation

and CIM/21 provide a major horse power boost for future applications and sharing data.

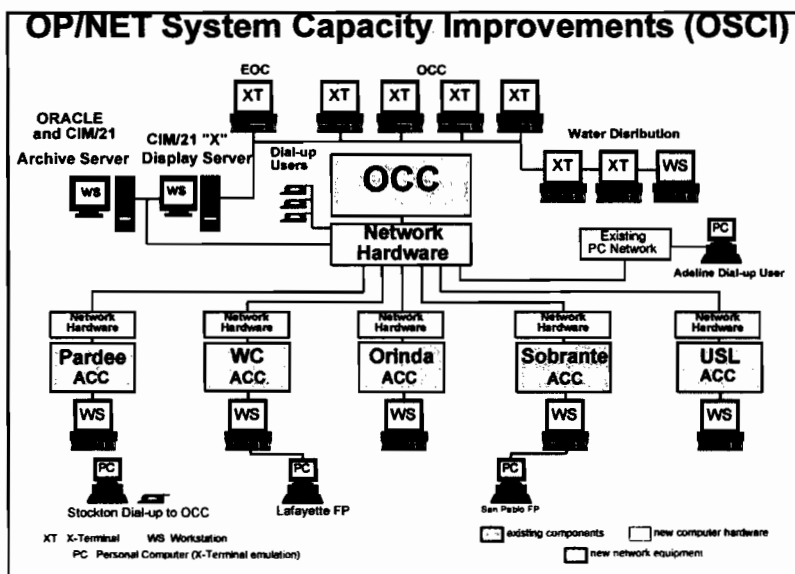


Diagram 8 OSCI CIM/21 OP/NET Overview

Paper Number: 7007
Turbo Your IMAGE

Donald L. Clapp
5620 W. CR 700 N.
St. Paul, IN 47272
(317) 525-9935



IMAGE-I (92069) is a robust hierarchical database. I have worked with this package since it was first available. Here is a definition of a database: "A collection of logically related files containing both data and structural information".

Although I equate IMAGE and "database", I have seen applications implemented with other techniques. My first example is a FLAT FILE. It can be used to store data, but you will suffer with slow access. This may be OK for small, simple applications.

The next example is a SORTED FILE. This would allow you to do a binary search. This kind of search is fast, but adding a record to the file causes a need to sort the file. A sort of a large file will have major impact on the system. It will use a lot of resources, disc, and CPU time. Plus, the file will not be available while it's being sorted. This might be OK if the data were static; for instance, if records were changed or added only once a week. Another major shortcoming is that you can sort on one key only. If you need access by a second key, then you have to use a serial search -- slooow.

The last example is an INDEXED FILE. This can be implemented with two files or two sections of the same file. One part contains the key and the position of the

corresponding data. The second part contains the data. This style will provide for fast access and can be set up with multiple keys. After adding several records, the file will need to be "reorganized". This is about the same as unloading and loading a database. An example of an indexed file package is KSAM on the HP3000.

IMAGE data files are type 2 (fixed length record) files with a root file that is type 1. The root file contains the definition of the fields, records, and files of the database as well as passwords.

There are several reasons to use a database. By defining the structure in one place (root file), the applications can be made to work across multiple databases. For example, QUERY will add, delete, modify, and report from any database. Using a database provides a consistent programming interface through a set of subroutines. These routines are a level above the standard file access routines. With these routines you can access data by field as well as by record. This allows some structural change to the database with no effect on your applications. It is possible to access the database using the file routines directly, but is NOT recommended. Using a database for your applications can provide much better performance with additional and more flexible security. The performance improvement applies to both reading and updating the database and assumes that you have a handle (key) for the data. Without a key to search with, the database routines will use a serial search. The extra layer of software will cause this kind of search to be slower than a plain vanilla flat file. So if performance is what you want, be careful what you ask for. With a key, a search is

nearly file size independent and very fast. Without a key, the search time is proportional to the file size. On a file of 50 records, a serial search may be OK. With a file of 100,000 records, bring a book to read while you wait. Not only will the search be slooow, the disc will be very busy, and therefore, everything else using that disc will slow down or stop.

The security features of an IMAGE database address two issues: the protection of sets and individual fields with passwords and protection from concurrent update problems. There is provision for 15 protection levels with a password for each. This protection can be applied to individual fields as well as sets. Opening the database with QUERY using the level 5 password might allow you to read half of the fields and write on none of them. If you opened the database with the level 15 password, then you would have read and write access to all of the database. The concurrent update protection is implemented with a class number. The class number is acquired by DBCOP when a database is first opened and returned to the application in the DBOPN call. The lock/unlock is done by calls to the DBLCK/DBUNL routines. The other database routines check that you have the database locked when modifying it. Even though this seems to be quite good protection, I have been told by others not to trust it completely. Two applications updating the same area of the database apparently can corrupt the data.

A properly designed database will remove and eliminate redundancy 8-).

These benefits do not come without cost. The design and implementation of a database does take some time and effort. Your applications will be larger because of the additional layer of software, and the data files will be larger because of the pointers and flags that are added to all of the records.

UTILITIES

Now that you have an IMAGE database, what are some of the tools for working with it? I'll leave the discussion of the subroutine library to the manual. I have divided the utilities that I am aware of into 4 groups. Within each group are utilities supplied by HP and those found in the CSL-1000 from INTEREX. The groups are: read only, data editing, structural change, and performance.

READ ONLY

QUERY (HP) This very large program can select and produce reports from any database. The reports can have sorts on multiple columns with subtotals and grand totals. It can do counting and averaging. Fields can be formatted with commas, periods, and dollar signs for accountants. There are two shortcomings of QUERY: it will work with only one set at a time, and it is slow. It is an excellent tool for confirming that your application did in fact update the database correctly.

DBSTR/DBRST (HP) These programs are used to perform a physical backup/restore of a database. No changes can be made to the structure or the data. I would

be inclined to use FST or ASAVE. Either of these would likely be faster and accomplish the same thing.

DBSPA (HP) This program goes through a database and counts the records. The number of records flagged as in use is compared to the number that the root file shows to be in use. You have a problem if they are different. As the program finishes, it prints a report of its findings. Two problems with this program are that it is slow, and most important, it requires exclusive use of the database.

DBSPB (CSL) This program was written to address the performance problem of DBSPA. By accessing the data sets directly with a large DCB, it is several times faster. It also provides more information. Like DBSPA, it needs exclusive use of the database.

DBSPC (CSL) Frequently, the only information you want is: "How am I doing on percent full of each set?" "Did I guess the data set sizes correctly?" "Has a user or program run wild and filled the database?" This program will report that information very quickly on any size database. The time it takes is generally limited by how long it takes to print the report. It's fast because it displays only the information contained in the root file. Not only is it fast, it will even run against a database that is already open exclusively. That piece of magic (violation of the rules) will be explained later.

DBVfy (CSL) If an application that is updating a database does not complete its work, it can leave the database in an inconsistent state. DBSPA/DBSPB might tell you there is a problem, but not where or what the

problem is. This program will do a detailed examination of a database or just one set. For each detail set, all of the paths to that set are checked. If there are three keys to a set, then every occupied record in the detail will be accessed three times. The forward and backward pointers are checked, and the number of links in a chain is compared with what the master record shows. Any discrepancy is reported in detail. With this information you might be able to fix the problem; however, there is no provision in this program for that. The program is slow and opens the database exclusively.

DBSYN (CSL) The design rules for a master set of a database call for the capacity to be a prime number and to have approximately 20% free space. This is because the capacity count enters into the hashing calculation of which record number to use for a particular key. This works well most of the time, but wouldn't you like to know how well? This utility will read an existing master set and re-hash the keys, keeping track of where the records should go. After processing all of the records in the set, it counts the synonyms and also reports the lengths of the synonym chains. Lots of synonyms and/or long synonym chains will adversely affect performance. If you wanted to "improve" the hashing algorithm, this would be the tool to use to measure the results of your efforts.

DBHLP (CSL) This program will decode error numbers for IMAGE-I and IMAGE-II. You pass it an error number and it returns the corresponding text. For example, "CI> DBHLP 106" will cause the program to display "Master set is full". This is quicker and easier than finding the message in the manual or memorizing all 97

error codes. The program will provide much more information if called by the subroutine DBERROR, which is also found in the CSL. The calling sequence of the subroutine DBERROR is set up like the other IMAGE routines. The first four parameters are: database, set, mode, and status. The next parameter is a variable called "who" which has a value between 1 and 15 that identifies the IMAGE routine that failed. The sixth parameter is called "local" and defines where to display the decoded information. The low order 3 bits are used to direct the output:

- Bit 0 on for a one line output to the local crt
- Bit 1 on for long report to the console
- Bit 2 on for one line report returned to the calling program.

All combinations are legal, including a zero for NO report. The one line report decodes the error number and the name of the subroutine that failed. The long report also displays the user logon, session number, date and time, database name, set name, decoded mode, complete status array in two formats, program name and location within the program where DBERROR was called. The reason for providing the one line report back to the calling program is so that, if you were in a block mode screen, then the local application should handle the display. Your application may still check for some error codes for branching purposes, but you don't need to provide the text for messages or the code to display them. DBERROR is only 136 words and provides more information than most programmers would put in an application.

OPEN LOCKED FILE (CSL) This is the "magic" routine that will let DBSPC open the root file even if

already open exclusively or open to seven other programs. DBSPC calls this routine only if the regular OPEN fails with an error code of -8. This is for FMGR files only. The DCB is set to read only, which is fine for DBSPC. Remember that you are peeking behind the curtain; therefore, you could see peculiar results. If you want to use this routine for some other application, it will also open a file in spite of an Lu lock and without the security code. Warning!! the DCB must be at least 144 words, because it is used as a buffer for the directory search.

DATA EDITING

QUERY (HP) This program provides for adding, deleting, or modifying almost any field in a database. The exceptions to this are: keys can only be added or deleted, and automatic masters are not accessible. The generality makes this a very cumbersome way to interact with a database. I view it as acceptable to add a record or to delete a few records.

DBBLD (HP) This utility will load an existing database with data. The data input is a text file in a well defined format. Each record in the database can require one or more physical records in the input stream. The input file can be produced with a text editor or an application program from some other form of database. The input data could come from another system that represents data in an incompatible way. For instance, the real numbers could be in IEEE format in the foreign system. This would prevent a simple binary transfer. Another variation is to produce an input file from some database,

EDIT this file to add, delete, or move a field and then use as input to DBBLD.

SKD DBSAVE (CSL) This utility probably should be called DBUBL for DataBase UnBuiLd. It is the inverse of DBBLD. It will unload a set or a complete database into a flat text file in a form directly usable by DBBLD. It is a read only utility but belongs with DBBLD. The program does use the standard IMAGE subroutines; therefore, it should work with IMAGE-II, except it is not set up to handle I2 or R4 fields. Adding the ability to handle these field types should be easy. The database is opened with a mode of 1, shared read/write.

While we were porting a database application from a 1000 to ALLBASE on our HP-UX system, I used this program to periodically copy the information between the systems. A small command file executed this program for each manual and detail set. One of the fields was a different size on ALLBASE, so we ran EDIT in batch mode on this set expanding the field by four characters. When all the files were in the form that the HP-UX system was expecting, we used FTP to transfer the files. When this was completed, we purged the files from the 1000. This provided a way to keep the databases on two systems synchronized with very little effort.

This program could also be classed as a structural change routine. In examining one of our databases, we determined that an additional key in one of the sets would provide a significant increase in performance. We changed the schema and built a new database. The programs DBUNL/DBLOD were used to put the existing

data into the new database. DBLOD added all of the manual masters but would not add the detail records because of the new key. Next we used SKD_DBSAVE to pull the data out into a flat text file, and used EDIT to replicate the data in the record into the new field at the end of the record. Then we passed this file to DBBLD and went home. The next morning our new database was ready to use.

DEDIT (CSL) This is a block mode QUERY. By using the standard IMAGE calls, this utility allows adds, updates, and deletes on most IMAGE-I databases and should work on most IMAGE-II databases. The database definition is used to build a block mode screen. The screen contains the field name in normal video followed by an inverse video box whose length is dependent on the data type. Both integer and real boxes are 6 characters, with text boxes being the length of the database field. There are some restrictions imposed for various reasons. The terminal must be an HP terminal or good emulation of one. The largest text field that DEDIT can handle is 78 characters. This is imposed by the terminal. The maximum number of fields is arbitrarily set to 80. The maximum number of characters in all fields combined is about 1000. The program will not handle compound items. IMAGE-II has two field types (I2 and R4) that this program will not recognize. Adding support for these field types should be easy, but a general solution for compound items would be difficult.

STRUCTURAL CHANGE

DBUNL/DBLOD (HP) This pair of programs can be used for backup/restore of a database or to change the

structure. This is the only method that is supplied by HP to make changes to the structure of a database. Most parts of the database can be changed using these utilities, but be prepared to do without the use of your database for awhile. The good news is that a corrupt database which is unloaded and reloaded will be clean. The bad news is a large database will take many hours to reload.

DBSEC (CSL) This program lets the database manager add or change a level word and change the security levels for an item. It does require exclusive access and modifies the root file directly.

DBITM (CSL) QUERY uses the item definition in the root file to control its formatting of each field. You may have set your database up with an item defined as real, 2 words, but decided later to use it as 4 characters. Since these are the same size, your application programs doesn't care. With this program, you can change the definition in the root file to reflect how you are now using the field. You can also change the name of an item. The program asks if you want to change the size of an item. This would be useful, but is difficult to implement, because it affects all of the sets that include that item. This feature is not available at this time.

DBMVR (CSL) This routine will move a root file to a different Lu. Root files are small type 1 FMGR files. It would seem that you could simply use the FMGR command to move the file. Although this will move the file, DBOPN will complain and not open the database. The root files namr is embedded in the root file. This self-referencing is

used as error checking. DBMVR moves the root file and then adjusts the inside of the root file to satisfy DBOPN.

DBMOV (CSL) For performance or for space reasons, it is sometimes necessary to move one or more sets to different Lus. The data sets are type 2 FMGR files, some are very large. Since the root file contains the namr of each set, this can be used to verify that the requested set is valid and to find the current location of the set. The root file is not changed to reflect the new location of the set until the file is copied to the new Lu. Copying a file works, because all of the links between and within the database are by record number.

DBCPD/DBCPM (CSL) These programs will change the capacity of detail sets and master sets respectively. Both programs compare the requested size with the number of records currently in the set. If you are making a detail set smaller, DBCPD examines the space beyond the new size to see if any active records are there. After passing these checks, DBCPD creates a file of the proper size, rounded up to be evenly divisible by 12. It then writes nulls to all of the records in the new set. The smaller of the new size or the old size number of records is then copied to the new file. DBCPD then creates a linked list of the free records, modifies the root file, and you are done.

DBCPM does round the file size up but does not change the capacity from your request, since this enters into the hashing algorithm. The program writes nulls to all the records in the new set, then reads the old set serially, hashing the key of each active record to find the position to store the record. When all the records have been

processed, it modifies the root file and quits. DBCPM will work on an automatic master even though it is not normally accessible.

PERFORMANCE TUNING

SET DCB SIZE (CSL) This seven word subroutine allows you to determine the size of the DCBs that DBOPN will use when opening the sets of a database. Unless you override it, DBOPN will use 272 words (2 blocks) for each DCB.

DBDCB (CSL) This simple program changes the fourth word of the root file. This word is used by DBOPN to determine the number of DCBs to allocate when opening a database. It is set when DBDS creates the database, and the allowed number is from 1 to 17. The limit of 17 is hard coded in DBOPN. The number is the maximum number of keys in a single detail set plus 1. If your database has only five manual masters, then this number would be 1. If your database contains a detail set with 5 keys then this number would be set to 6. This is fine for QUERY which will deal with only one set at a time. If your application has less DCBs than sets in the database, it will waste some time closing and opening files. Since the free space above your program is used for these DCBs, some care and planning has to go into deciding how many and what size DCBs to request.

DB RUN SIZE (CSL) This subroutine is called after DBOPN and returns the number of DCBs allocated, the total size of the IMAGE run table, and the amount of free space

available before the call to DBOPN. While doing this routine, I found GETBF, a routine that is used by DBOPN, to allocate buffers in free space.

TESTDB (CSL) This program is used to tune the size of DCBs to request. It calls SET_DCB_SIZE, DBOPN, DB_RUN_SIZE, and DBCLS. It then prints the results of the call to DB_RUN_SIZE.

DB MONITOR This program is NOT in the CSL. It is left to the user to provide their own. There are several advantages to having one program open the database and all of the applications passing requests to it.

By using a monitor, your application doesn't have all of the IMAGE and FMGR routines appended to it and doesn't need free space above the program for DCBs, Tables, etc.

By using the utilities DBDCB and SET_DCB_SIZE, your monitor can have more and larger DCBs. More DCBs means less file opens and closes. Larger DCBs improve serial access but will not help keyed access by much. You might want two monitors tuned differently. One for all updates, the other for searches.

This approach solves the concurrent update problem, if any. This is the way that the file system works using D.RTR .

IMAGE-I exists in FMGR land, therefore, only seven programs can have a file open at the same time. By using a

monitor, your applications are not constrained by the seven program limit.

By putting the database code in one program, it can error check everything and should (must) be very robust. This will make applications more simple and the database more stable.

You could add transaction logging if you needed the audit trail. This would be easy, because you only have the one program modifying the database.

The use of a combination of these routines can improve the performance of your applications by a factor of two or more. Both the speed and up time will be improved.

Now that you have decoupled your applications from the database, you have implemented client/server software. A small additional embellishment would provide for the database and applications to be on separate systems. See the paper by Paul Gerwitz in the 1987 INTEREX proceedings for a description of a distributed IMAGE database. With the 1000 as a database server, the clients could be any other systems that could communicate with it. There is a package called "DBACCESS/1000" from COMSCI DATA SYSTEMS, INC. that provides the tools to implement an HP-UX client to an IMAGE 1000 database. COMSCI also provides utilities with functionality that goes beyond the programs in the CSL for both IMAGE-I and IMAGE-II.

ROOT FILE STRUCTURE

Now that you have been exposed to the existing utilities, I will document the contents of the root file so that you can write your own tools. The IMAGE manual does describe the tables found in the root file but does not tell how to find them.

Root File Structure

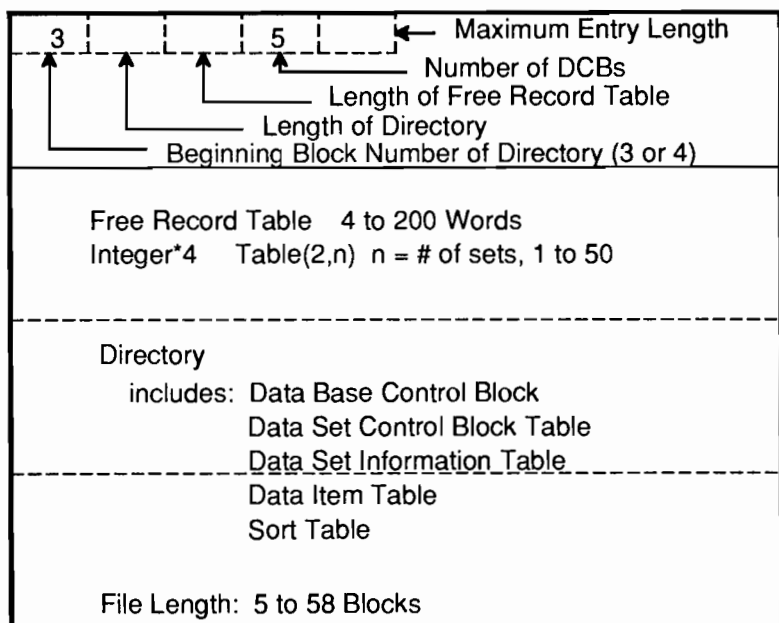


Figure One

Figure one is the overall structure of the root file. The first word of the file is the starting block number of the Data Base Control Block (DBCBCB). The only valid values for this word are 3 or 4. It is used to

confirm that you have opened an IMAGE-I root file and it is used in the address calculations for the other tables.

The fourth word is the number of DCBs that DBOPN will allocate, space permitting. This is the word that DBDCB changes. The range of valid values for this word is 1 to 17. Any other values will cause DBOPN to fail.

The Free Record Table (FRT) starts with the second block. If the database has more than 32 sets, the FRT continues into the third block. The FRT is a 2 by n array of 32 bit integer values. The table is indexed by set number. The first value of each pair is the number of free records. For a detail set, the second value is the record number of the first free record. The free records are linked using the second and third words of each record. The FRT is updated on disc whenever you add or delete a record. If the record you added or deleted is a detail, at least one other detail record is also updated. This provides a robust system, but it will be slower than you might think. All of this record keeping is hidden inside the IMAGE subroutines.

Data Base Control Block

1	Data
2	Base
3	Name
4	Security Code (FMGR)
5	Cartridge Number
6	
7	
8	Item Count
9	Item Table Pointer
10	Data Set Count
11	DSCBT Pointer
12	Sort Table Pointer
13	
14	Level Word Flag
15	Level
16	Word
17	One
18	Level
19	Word
20	Two

57	Level
58	Word
59	Fifteen

Figure Two

Starting with the third block, or for a large database, the fourth block, you will find the Data Base Control Block which is 59 words in length. See figure two. This structure contains some general information and pointers to the other tables. The general information includes the database name, FMGR security code, and cartridge

reference. This is why you can not use FMGR to simply move or rename a root file. The level word flag (word 14) is -1 if no level words are defined, 0 if there are level words defined. Starting in word 15 are the level words (passwords). These are three words (6 characters) each, and there are 15 entries. They are case folded, left justified, blank padded, clear text. Any level words that are not defined will be all blanks. The pointers are all relative to the start of the DBCB. The FORTRAN code to locate a table is:

$$\begin{aligned} \text{RECORD} &= (\text{POINTER}/128) + \text{START} \\ \text{WORD} &= \text{MOD}(\text{POINTER},128) \end{aligned}$$

To access word 4 of a table entry, add 4 to WORD for the subscript.

Word 11 of the DBCB points to the Data Set Control Block Table. See figure three. This table contains a 17 word entry for each set in the database. Since this table has the data set name, location, and size, you can not use FMGR directly on a set. Notice that word 8 is yet another pointer. The formulas for tracking down the table that this points to are the same as the others. The set type is found in bits 10 & 11 of word 5 of each entry. The values for set types are:

Automatic master =	0
Manual master =	1
Detail =	2

The information on the additional tables of the root file is found in the IMAGE/1000 Reference Manual, part # 92069-90001.

Data Set Control Block Table

1	Data	Set
2	Set	#
3	Name	One
4	Cartridge Number	
5	Set Type	Media Length
6	Data Record Length	
7	# of Items	# of Paths
8	DSIT Entry Pointer	
9	Capacity	
10	Count	
11	Search Item #	
12	
17		
1		
2	Data Set Name	
3	Set # 2	

Figure Three

SUMMARY

IMAGE-I is an old package with a few shortcomings, but it does perform well and is simple to set-up and use. With the application of the utilities I have described, it can meet or beat IMAGE-II while using less memory.

BIBLIOGRAPHY:

- Stephen R. Carter, Relational View Of IMAGE With Real Zip,
INTEREX TECHNICAL PROCEEDINGS - 1987
- A. Marvin McInnis, IMAGE/1000:Secrets HP Never Told You,
INTEREX TECHNICAL PROCEEDINGS - 1987
- Paul F. Gerwitz, A Database Management Subsystem For
IMAGE/1000.
INTEREX TECHNICAL PROCEEDINGS - 1987
- Don Wright, An IMAGE Data Extraction Accelerator,
REAL TIME INTERFACE, July 1992



HP3000 System Consolidation

By

Steve Cole

Northern Telecom

Paper No: 8000

Introduction and Disclaimers

With Hewlett Packard's release of the 99x and 9x7 processors the ability to consolidate data centers is now a reality. The question that is most asked is "*Why consolidate?*". This paper discusses the benefits achieved through consolidation that intend to assist the data center manager with that question.

If the decision is to consolidate, this paper discusses a method of consolidation that has proven reliable over a variety of dynamic environments. Every computing environment is unique, therefore the method described in this paper may not work in all cases, but should work in 80 percent of them.

This paper discusses consolidation from the remote system and remote site perspective. However, consolidation can provide business improvements within the same data center. The methodology provided works for both types of consolidations. The remote system example is used due to the complexity of remote communications.

This paper discusses the use of certain products and the personal interpretation of their output. This is not intended to be an endorsement or recommendation of these products over similar products on the market. The key is that you use standard products so that a true "apples-to-apples" comparison is made. The paper also demonstrates one method of interpreting the data for the purpose of consolidation. This interpretation is not intended to represent the only method of analysis.

***To Consolidate or Not To Consolidate?
That is the Question***

The decision to consolidate should be based on business requirements. The primary reasons to consider consolidation is cost improvement, simplification of operation and better utilization of computing resources. Briefly discussed below is more detail on the benefits that can be achieved through consolidation.

Reduced hardware and software costs are two of the major benefits that can be achieved through consolidation. Without attempting to go into detailed cost analysis to demonstrate this point, let us evaluate a couple of key issues. For each system in operation, there are a number of expenses that occur. Each system has a hardware support contract that provides the specified support for that system. The cost of the hardware support for a 995/200 is cheaper than two 980/100 systems and the performance level is higher. This concept will also apply to software support from Hewlett Packard as well as 3rd party vendors. A detail cost analysis needs to be performed to determine the actual savings, but in some cases the savings may justify the cost of the consolidation.

Simplification of operations is another area of cost improvement. By consolidating data centers, support costs are reduced. Any remote staffs can be freed up to support other activities. The consolidation also permits standardization of activities reducing duplication normally seen when there are multiple data centers.

Better utilization of computing resources is another reason to consolidate. When you have multiple systems running production environments, you don't have the benefit of sharing available capacity. If one system operating at one location that is 20% busy and another system at another location operating at 100% busy, your only option is to upgrade the one system. This situation is multiplied as more systems are involved. Through consolidation, computing resources become a sharable resource. Available capacity can be shared by all users at all locations. As capacity issues occur, a single upgrade can be purchased that benefits all of your customers.

Cost Performance

One of the key issues to evaluate when considering consolidation is the long term costs. The chart below gives the cost to purchase various HP3000 systems and the performance rating of the systems in RPI. (*The RPI ratings given are discussed later in this paper and represent the expected through-put of the system*). If your site is considering upgrades within the next year, then consideration needs to be given as to how the money can best be spent to achieve the most performance for the dollar invested.

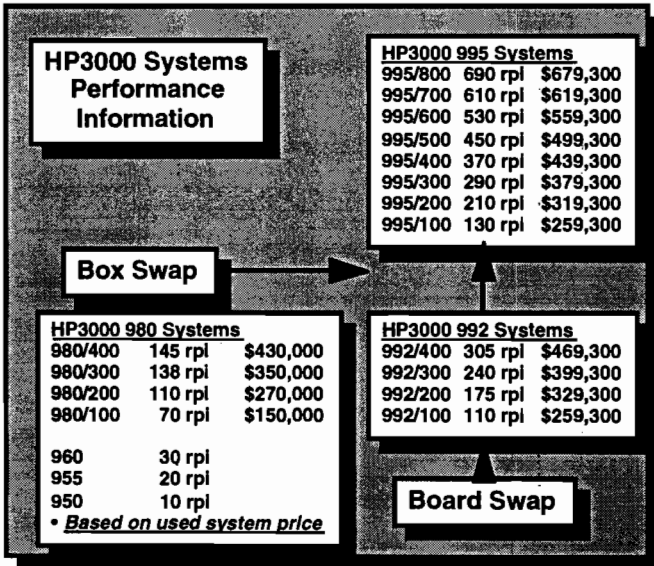


figure 1.

The above chart provides a cost comparison between the purchase price of the HP3000 Series 980s, 992s and 995 processors. The cost for the HP3000 980 Series is based on estimated used system prices. The cost of the HP3000 992 and 995 Series processor is based on Hewlett Packard's advertised price for the base system with a 100 user license and no data base manager software.

The chart below provides an analysis of the Cost Performance of the HP3000 Series 980s, 992s and 995 series processors based on the information provided in *figure 1*. Though the information provided is limited to only 3 series of HP3000 systems, the same trend is true when comparing the 9x7 and 9x8 Series processors to the older processors (i.e. Series 922s, 925s, and the 95xs).

HP3000 Purchase Cost	
<u>HP3000 995 Systems Purchase Cost</u>	
995/800	\$984/rpi
995/700	\$1,015/rpi
995/600	\$1,055/rpi
995/500	\$1,110/rpi
995/400	\$1,187/rpi
995/300	\$1,308/rpi
995/200	\$1,520/rpi
995/100	\$1,995/rpi
<u>HP3000 992 Systems Purchase Cost</u>	
992/400	\$1,539/rpi
992/300	\$1,664/rpi
992/200	\$1,882/rpi
992/100	\$2,357/rpi
<u>HP3000 980 Systems Purchase Cost</u>	
980/400	\$2,965/rpi
980/300	\$2,536/rpi
980/200	\$2,454/rpi
980/100	\$2,142/rpi

The long term savings with the new series HP3000s is the cost and performance of upgrades. The following chart provides upgrade costs and cost per RPI of the upgrade. By consolidating data systems and migrating to the new system technology, the long term cost associated with upgrades is reduced. The data center is now able to purchase more capacity with less capital expenditures than with the older technologies.

HP3000 Upgrade Cost		
<u>HP3000 995 Systems Upgrade Cost</u>		
995/800	\$70,000.	\$875/rpi
995/700	\$70,000.	\$875/rpi
995/600	\$70,000.	\$875/rpi
995/500	\$70,000.	\$875/rpi
995/400	\$70,000.	\$875/rpi
995/300	\$70,000.	\$875/rpi
995/200	\$70,000.	\$875/rpi
995/100		
<u>HP3000 992 Systems Upgrade Cost</u>		
992/400	\$80,000.	\$1,230/rpi
992/300	\$80,000.	\$1,230/rpi
992/200	\$80,000.	\$1,230/rpi
992/100		
<u>HP3000 980 Systems Upgrade Cost</u>		
980/400	\$100,000	\$14,285/rpi
980/300	\$100,000	\$3,571/rpi
980/200	\$100,000	\$2,500/rpi
980/100		

The above chart provides a cost comparison between the upgrade price of the HP3000 Series 980s, 992s and 995 processors. The cost for the HP3000 980 Series is based on estimated used system prices. The cost of the HP3000 992 and 995 Series processor is based on Hewlett Packard's advertised price for the base system with a 100 user license and no data base manager software.

Historical Trends

A number of MPE environments have migrated from the MPE/V based systems into the MPE/iX world. During the early stages at least one processor was installed at each location to support local computing functions. With the introduction of faster and lower cost communications, distributed processing from the individual data centers increased. Additionally, the capabilities of the DTCs, Network Services, X.25, ARPA Services, and Wide Area Networks encouraged the distribution of service from existing data centers.

With the demand on business today to reduce operating costs, continuing with the trend of distributed processing from multiple locations is becoming less cost effective. The determination of whether or not consolidation provides advantages needs to be determined by sites. In some cases, distribution of computing was dictated by business requirements. If applications and the user base (i.e. outside users dialing in to the system) require that multiple systems be used, then consolidation may not be right for your environment. However, if your site is running standard applications or has no reason for running multiple systems, then you are a prime candidate for consolidation and you should continue reading.

Defining our Consolidation Environment

Before we discuss the methodology for consolidation we need to define the environment we are consolidating. **Figure 2.** represents an environment of HP3000 980/100s and HP3000 980/200s supporting both local and remote customers. The processors are located at 4 sites within the US. with regional offices across the country supported by remote DTCs or network access.

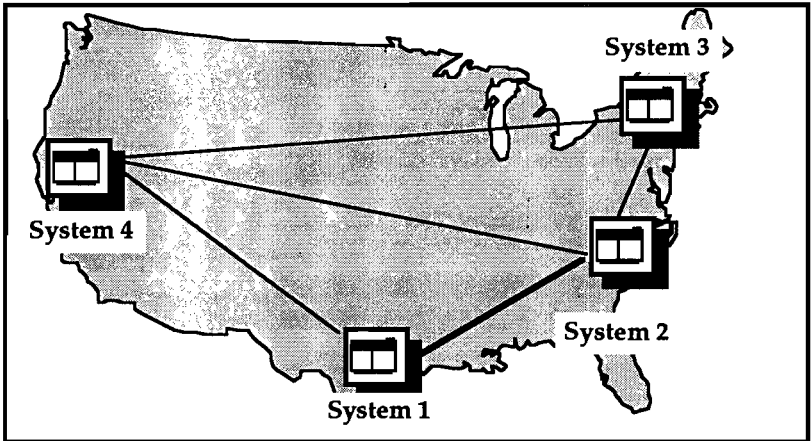


figure 2.

Figure 2 represents a distributed computer environment that is using remote systems and remote communications to connect users and systems. This environment is the example for our consolidation activity.

Relative Performance Index

Sizing the system that you are consolidating into is a significant problem. Relative processor comparison charts provided by Hewlett Packard contain some useful information relative to the speed of the CPU but does not accurately reflect the relative through-put of the system.

Through personal consolidation activities a new **Relative Performance Index** or **RPI** was developed. This **RPI** rating was developed by comparing the actual performance delta of a known application before and after an upgrade. This was done by comparing **LASERX** charts from one system before the upgrade to the **LASERX** charts after the upgrade. By comparing these charts for a relatively static application, the actual performance delta could be determined and converted into an **RPI** value. This process was first utilized for Series 70 consolidations and as such the Series 70 became the base with a rating of 10 RPI. As additional consolidations occurred, other systems were added to the chart as their RPIs were determined. The **RPI#2** rating in the chart below reflects the values observed.

Make	Model	RPI	Base	RPI#2	≈MIPSTM
HP	S70	0.3	917	10	3
HP	917	1.0	917		9
HP	927	1.0	917		9
HP	937	1.0	917		9
HP	947	1.0	917		9
HP	957	1.6	917		14
HP	967	2.0	917		17
HP	977	2.6	917		22
HP	948	1.1	917	18	9
HP	960	1.3	917	30	13
HP	980/100	2.4	917	70	20
HP	980/200	3.8	917	110	32
HP	980/300	4.9	917	138	42
HP	980/400	5.9	917	145	50
HP	987	3.2	917		27
HP	990S	2.8	917	100	23
HP	992/100	3.5	917	110	30
HP	992/200	6.0	917	175	49
HP	992/300	8.1	917	240	68
HP	992/400	10.0	917	305	86

figure 3.

Estimated RPI Ratings of the HP3000 995 System

Make	Model	RPI	Base	RPI#2	≈MIPS™
HP	995/100		917	130	
HP	995/200		917	210	
HP	995/300		917	290	
HP	995/400		917	370	
HP	995/500		917	450	
HP	995/600		917	530	
HP	995/700		917	610	
HP	995/800		917	690	

figure 4.

Figure 4. represents the estimated RPI rating of the new HP3000 995 systems. These estimates are based on performance information gathered from various sources and the relative I/O performance of the 99x system. Even though these numbers are estimated, I feel that they are reasonably accurate.

Using the RPI Ratings To Size Our Consolidated System

One of the first steps to consider in the consolidation activity is the size of the system required to support the consolidated computing. This section of the paper discusses this activity.

Phase I of the activity is to collect sufficient performance information from each of the systems over the same time period to evaluate. Usually a 30 day collection is sufficient if it includes any peak processing events such as month end processing. Other factors that affect the sizing of the system is any increased quarterly or seasonal processing that occurs. This collection should be performed with the same performance tool on all systems in order to compare similar data.

Phase II of the activity is to evaluate the information obtained from the collection. Don't attempt to evaluate the overall system performance, but instead concentrate on the most critical areas that impact your business. Determine if your most critical processing is related to your on-line users or your batch processing. This decision will affect the method you use to size your system.

On-line Performance is the area that most affects the average data center. These are the users that log-on daily and are dependent on the system to perform their job. If this is the most critical function supported, then this should be the primary focus of our evaluation. We need to determine the peak loads of our on-line users. An example of this is that we have users that start logging on at 5:00am and with the second shift, don't log-off until 10:00pm. Our window for on-line access is from 5:00am until 10:00pm. However, if we look at the system trend we find that our peak on-line activity is actually between 10:00am and 3:00pm daily. This is the time when the most users are hammering the system and, as such, the time window that we should evaluate. If the systems that we are consolidating span multiple time zones, then adjustments may need to be made in the window.

Batch performance is the hardest area to predict. In some cases, if one batch job is running the performance charts may show high batch utilization. This, in part, is due to the way the MPE/iX operating system operates with batch processing. Any idle CPU time is assigned to batch processing and can affect the interpretation of the information provided in the performance chart. If batch is your most critical processing, then each of the critical batch jobs must be evaluated by day and time to determine the batch loading. Once completed, then the load by day and time can be determined. If overlapping occurs, additional capacity is required to support the load. Perform an evaluation of your on-line users and compare their increase requirements against the estimated batch requirements and select whichever is greater.

Evaluating The Performance Charts

Once the performance information is obtained and charted, two critical pieces of information are needed. The information you are interested in is the peak load and the average load. The reason that these two pieces of information are valuable, is to scale the size of the system required. The evaluation of the average load determines the minimum processor requirement. While, the evaluation of the peak load determines the maximum processor requirement, another factor to be considered is how loaded do you wish to run your processors? For this paper, we will set the threshold limit for the system at 80% utilization. When the system utilization exceeds the 80% mark, system upgrades are considered.

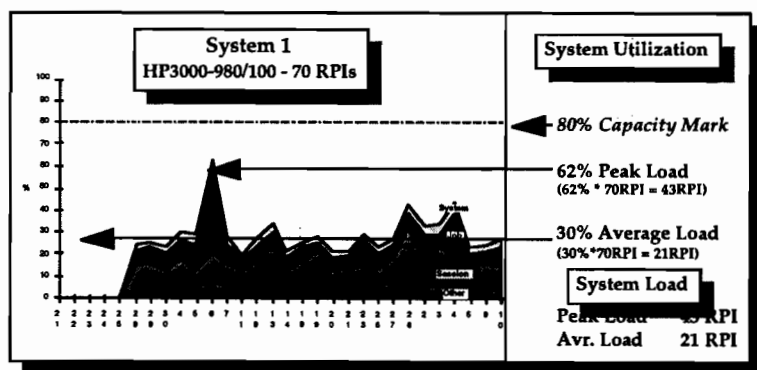


figure 5.

The performance chart for **System 1** (*figure 5.*) shows a peak load of 60% and an average load of 30%. If we refer back to the **RPI Chart** (*figure 3.*) we find that the RPI value for a 980/100 is 70 RPI. If we multiply the utilization times the RPI rating of the system we obtain the

system load in RPIs, as shown in the **System Utilization** section of *figure 5*.

Following is the performance evaluations for the remaining systems:

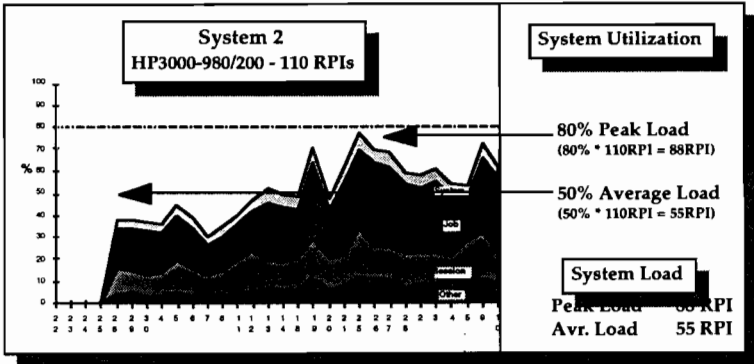


figure 6.

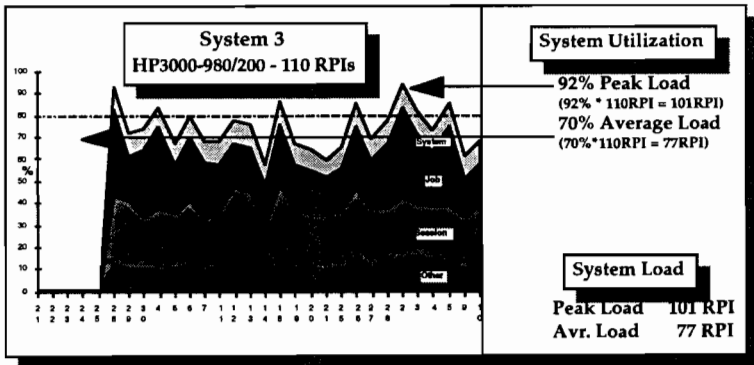


figure7.

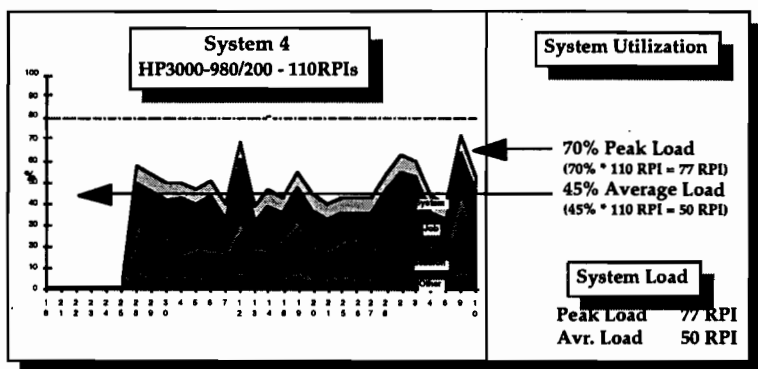


figure 8.

By evaluating and recording the information obtained from the four performance charts we obtained the following summary:

System Name	System Type	System RPIs	Peak Load	Peak RPIs	Average Load	Average RPIs
SYSTEM 1	980/100	70 RPI	62%	43 RPI	30%	21 RPI
SYSTEM 2	980/200	110 RPI	80%	88 RPI	50%	55 RPI
SYSTEM 3	980/200	110 RPI	92%	101 RPI	70%	77 RPI
<u>SYSTEM 4</u>	<u>992/200</u>	<u>110 RPI</u>	<u>70%</u>	<u>77 RPI</u>	<u>45%</u>	<u>55 RPI</u>
TOTALS		400 RPI	77%	309 RPI	52%	208 RPI

figure 9.

The summary above shows the peak and average RPI requirements by system. By using the RPI value, we are able to compare utilization values by system regardless of system type. By combining the requirements of each system, we obtain the minimum and maximum processor requirements.

Selecting the System

By evaluating the total line of the summary, we find that our maximum processor requirement is estimated to be **309 RPI** and our minimum processor requirement is estimated to be **208 RPI**.

	System RPIs	Peak Load	Peak RPIs	Average Load	Average RPIs
TOTALS	400 RPI	77%	309 RPI	52%	208 RPI

figure10.

If we take these requirements and look at the **RPI Chart (figure 4.)**, we find the system with **RPI ratings** to meet our requirements. **Figure 11.** represents the information obtained from the chart. From our look-up, we find that our minimum processor required for the consolidation is a **HP3000 995/400** processor with a **RPI rating** of 370. This processor is the smallest processor that meets both the average and peak processing requirements. At peak load, the processor is estimated to be 84% utilized. If we have the 80% utilization limit on the system, then the limit is exceeded. Under the 80% rule, we would need to move to the next level of processor or the **HP3000 995/500**. This processor would meet our processing requirements and would operate at approximately 69% utilization under peak load. However, if the 84% is within your system utilization threshold value, the **HP3000 995/400** will work just as well.

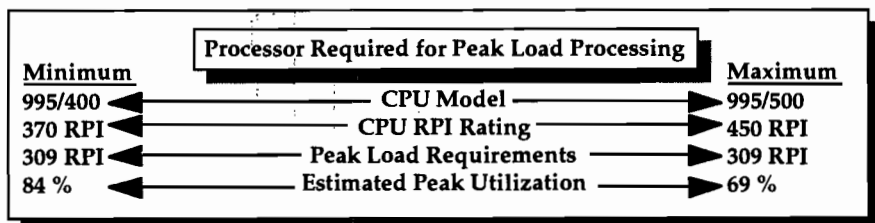


figure 11.

Methods of Communicating with the HP3000

Before discussing how we are going to support users at the remote sites, we have to have some understanding of the methods available to communicate with the HP3000 systems.

DTC Connectivity

The DTC or *Distributed Telecommunication Controller* is the telecommunications interface for the HP3000 systems. The DTC currently supports asynchronous terminal I/O, TELNET, and X.25 for the MPE/iX environment. The DTC connects to the system by a LAN connection and uses the AFCP protocol. The AFCP protocol used on the DTC is non-routable and requires bridging to support remote connectivity. (*MPE/iX Release 5.0 supports routable AFCP protocol with the new DTC 72*).

DTC Configuration Support

The DTC Configuration is supported by either a Host Based Configuration or through the ***Openview*** DTC Manager. The choice of DTC management is driven by choice or requirements. Certain functionality of the DTC require the use of the ***Openview*** product. If your site uses "*Port Switching*" (*Port Switching allows users connected to the DTC to switch between systems*) or Hewlett Packard's TELNET product, you are required to use ***Openview*** management.

The HP3000 systems support 2 LAN interface cards. Usually, one LAN card is used to support the network connectivity (*i.e. NS/3000 and FTP*) while the other LAN card supports the DTS (*DTCs*) network. This is the method I strongly recommend. As the company's local and wide area network grows and traffic increases, so do the problems. With the use of a single LAN card, unless specific firewalls are put in place, the DTC traffic must compete with all the other traffic on the network. This means that the DTCs are affected by any problems on the network such as streaming devices. By keeping the networks separate, then the congestion and the problems are also separated.

Asynchronous Terminal I/O

The DTC supports 3 primary connections for asynchronous connections for terminals and printers. The DTC supports 3 pin and 25 pin RS232 interfaces as well as the RS422 connections. Asynchronous connectivity is the most efficient method of connecting devices to the HP3000 but provides the least flexibility to the user, if connectivity to other platforms is required.

TELNET Connectivity

TELNET connectivity on the HP3000 can be supported by several means. Hewlett Packard supports TELNET by use of either TELNET cards that plug into the DTC, or the TELNET Express product. This product requires the use of the ***Openview*** DTC Manager.

Hewlett Packard's TELNET products support the protocol by processing the packets off of the DTC LAN and converting it to the AFTP protocol which is transmitted to the system. Output from the system is the reverse. Information is transmitted from the system to the DTC by use of AFTP where it's converted and outputs over the network. This solution creates several problems. One of the major problems is that the site is required to connect their DTC LAN to their local area or wide area network. This problem was previously discussed. Another problem that exists is determining the available capacity of the TELNET Express. The advertised number of users supported is approximately double the number of users that can be supported with reasonable response. The performance of the HP product is also related to the type of I/O performed. If the majority of the I/O is character mode, then the number of users supported is half that of block mode I/O. Difficulty exists in determining the physical loading of this solution as the number of users increase.

Another method of supporting TELNET is to use a TELNET server. Several companies provide this product that allows the server box to be connected to the local or wide area network and connects to the HP3000 by asynchronous connections to the DTC. This method of supporting TELNET provides certain advantages not provided by Hewlett Packard's solution. The first advantage is that the DTC network does not have to be connected to the local or wide area networks. This improves the reliability and performance of the DTC network. Another advantage of this solution is that all connections to the system are considered physical connections providing better system diagnosis and support. In addition to these advantages is the ability to set connection timers on the TELNET server. This addresses those cases where your network users hang and simply reset their PC and reconnect to the system. This approach leaves multiple unused sessions on your system that have to be identified and cleared. The connection timer on the TELNET server will clear connections that have no activity over a specified period of time. As the devices use modem connections to the DTC, the connection is cleared, the Carrier Detect signal to the port is dropped, and the session is automatically aborted. This solution also provides improved performance and capacity analysis.

Alternative Network Connectivity

Alternative methods of network connectivity are also available for the HP3000 system. One method that is frequently overlooked is the NS connection. Several products are available for PCs that permit connection to the HP3000 by NS/3000. This software permits a shared network stack on the PC, thus permitting simultaneous NS and TELNET connectivity. This method allows native connection to the HP3000 over a local or wide area network without the purchase of additional hardware and without impacting existing network connectivity.

The diagram below represents a network supporting remote desktop computers using a mixture of TELNET and NS Connect protocols. If NS Connect is used at all locations then the TELNET Server or TELNET Express box is not required.

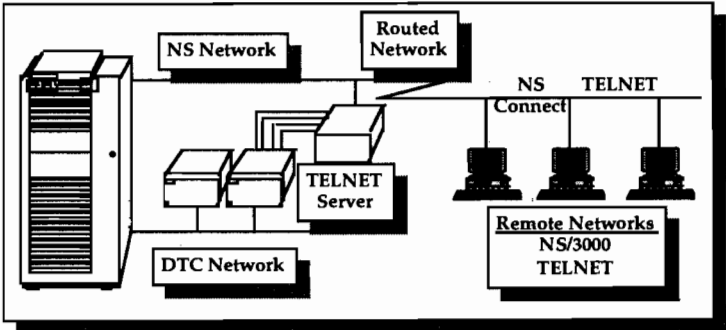


figure 12.

Supporting Consolidating Communication

Supporting a simple remote DTC configuration is quite simple. To implement, you need a couple of bridges and modems. The speed of the line required to support the remote DTCs is dependent on the number of DTCs and the type of utilization expected. Eight remote DTC-48s can be supported over a 56kb circuit with no problems. The efficiency of the DTC protocol is quite amazing. However, a clean communications line is a necessity. Redundant lines should be considered if access from the remote site is critical. The diagram below represents a very simple DTC network.

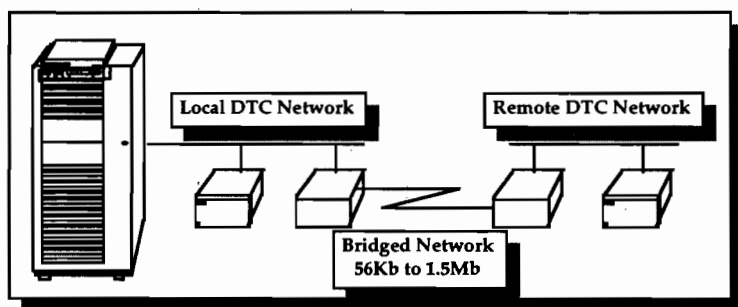


figure 13.

The complexity of the network increases dramatically when you have remote systems, remote network connected desktop units (*Workstations or PCs*) and printers. The diagram below represents a more complicated network utilizing HP products. The diagram below uses Hewlett Packard's *Openview DTC Manager* and *TELNET Express* product. This diagram also demonstrates how the NS and DTC Network must be connected together.

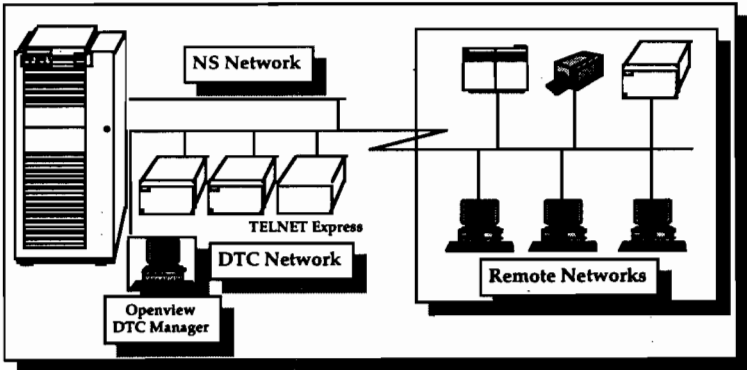


figure 14.

The diagram below represents a method of supporting remote connectivity without having to connect your DTC and NS Network. The diagram demonstrates remote DTC connectivity across a bridged network. This method of connecting DTCs can be used in multiple sites. The diagram also demonstrates the use of the TELNET server method of supporting network connected desktop computing. The diagram also demonstrates a configuration that supports NS connected desktop computing.

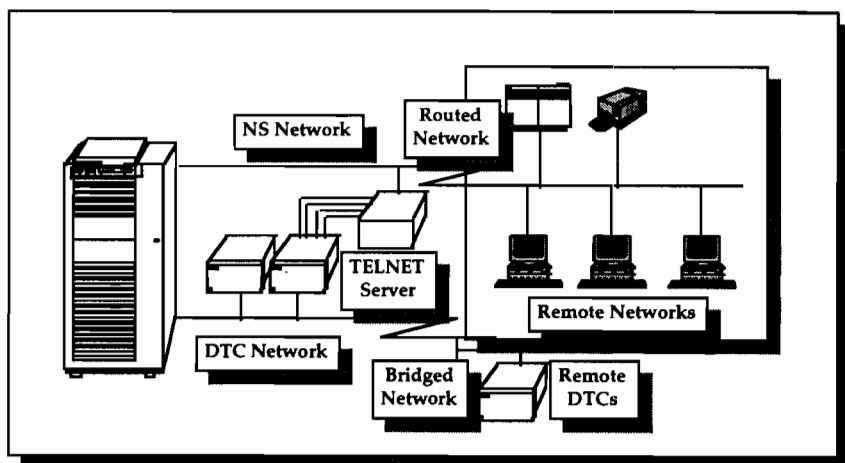


figure 15.

Remote Printing

One of the areas that we haven't discussed is remote printing. Consolidation of printing activities can be quite easy or a nightmare. The key to printing is how the devices are referenced. If the printers are referenced by device number, then you will have problems when you try to consolidate the sites. All printers should be configured with unique *Classnames*. All print activities should reference the device by its *classname*, not its device number.

Once the device class situation is resolved then migrating the printers becomes quite easy. By simply configuring the remote devices into consolidated DTS configuration and bridging the remote DTC network, the printers are available and ready for access. The network connected printers should work the same once the *classname* methodology is implemented.

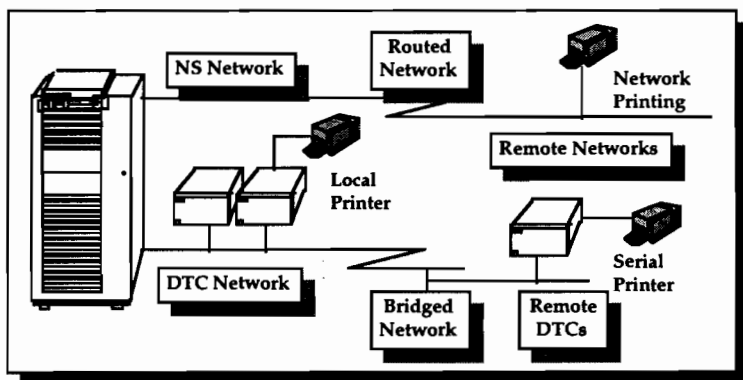


figure 16.

Data Center Printing

Support of data center printing on channel connected printers add a degree of complexity to the process. When the systems are consolidated, the channels these printers are connected to disappear. Two solutions are available for this problem. The first solution is to evaluate the central printing load and determine if serial connected printers can handle the load. If so, the problem is solved. The second solution is to find a method of supporting the central printing that has to be supported. Acquiring a system that the central printers can be attached to is the solution to this problem. Several companies provide software that will route the print from the consolidated system to the print server at the remote site. A system to drive the remote central printer can be acquired from various sources. If you are operating a pure MPE/iX environment, then you should look at a 3rd party for a used 922 or 925 type system to keep the cost down. An alternative is to price the HP3000 series 9x8 systems. Regardless of the system, the print server method will provide a low cost solution.

Summary of Consolidation Activities

When you are considering consolidating systems or data centers, then the activities discussed in the paper provide a starting point. Many of the activities can be performed parallel.

The first step that must be completed is to determine where the consolidated system should be located. Once decided, determine the size of the consolidated system. Again, the key is to use the same performance tool to analyze the utilization of the systems.

Another activity is to take an inventory of all software running at the remote sites. One of the major problems that occurs when consolidating is the differences in software running at each site. Activities should be initiated that standardize all conflicting software. This will simplify the consolidation activity.

Account and User names can also cause problems when consolidating sites. Audit all of the sites to determine conflicts in account names. When conflicts occur, account names have to be modified to ensure that each are unique.

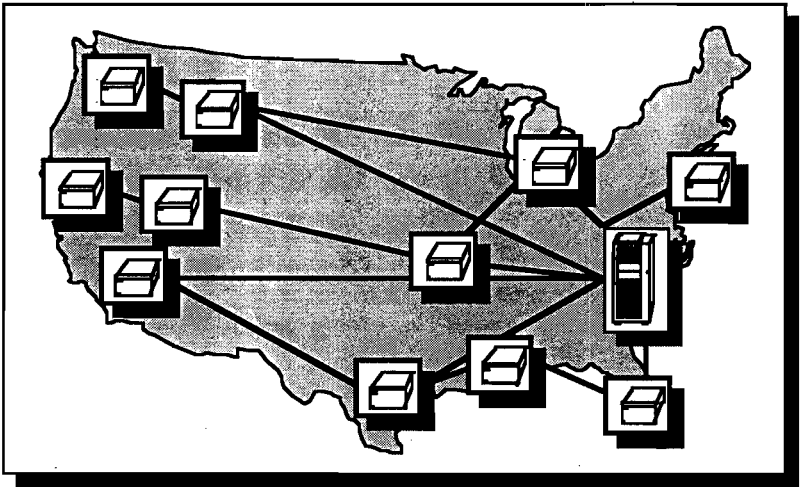
Inventory the network access at each of the sites. Determine the number and method of access used by users at each site to access the system. If you find more than one method of access for each protocol, consider standardizing on one method. The effort you spend here will be returned during the consolidation. In addition, consider standardizing the type of hardware used to support network access. Trying to troubleshoot problems on multiple types of hardware interfaces can be a nightmare.

Convert all printer access from device number to *classname*. By using this method, no changes will be required when the consolidation occurs.

One major problem that occurs when you consolidate sites is the amount of *Compatibility Mode* code you are running on the systems. The *Compatibility Mode* tables on MPE/iX are no larger than they were on the Series 70. When consolidating multiple sites and executing the *Compatibility Mode* code, the chances exist that you overflow some internal tables. Hewlett Packard has utilities that will help you identify this problem.

Conclusion

Consolidation of systems and data centers can provide long term cost advantages. Consolidation provide shared capacity, lower cost upgrades, reduced hardware and software maintenance, reduce operations and support costs, and standardization. The complexity of consolidation can be greatly reduced by planning and front-end activities. As you evaluate the requirements to consolidate, you should find that benefits should outweigh the investment. If they don't, then consolidation is not right for you. The key to all activities is to keep it as simple as possible. If things begin getting too complicated, re-think what you are doing. The end result can be quite rewarding.



Paper 8001

Closing the Holes in HP-UX

Joe Pennell
Paychex, Inc.
911 Panorama Trail, S.
Rochester, NY 14625
jpennell@paychex.com
(716) 383-3472

Abstract

The recent expansion of the information super highway has given all of us access to more data than ever before. Unfortunately, this opening of the lines of communication has also exposed our own data to the world. Although most of the browsing of our systems is typically harmless curiosity, steps must be taken to guard our most valuable resources. The surest way to protect your system is to never interface with the outside world. This would, however, deny you access to many valuable sources of information. It also does not protect you from tampering by individuals within your local network.

System security is a finely tuned compromise. As a security administrator, your task is to provide all of the tools necessary for your users to perform their jobs, without allowing them to reconfigure or destroy your system. The ability to grant various levels of access is built into the Unix operating system. Once these controls have been established, they must then be monitored to insure that your security scheme is solid.

The files which control the core of the operating system are the responsibility of the systems administrator. Since such files control access to the entire system, the systems administrator must maintain privileges to them. It is possible to automate procedures to confirm the validity of system files, and thereby increase the confidence in your security. The goal of this discussion is to show what files are critical, and provide methods for their authentication.

Introduction

Effective operating system security requires cooperation of all who use the system. In order to protect the integrity of your data, each user must understand the responsibility that comes with the assignment of a login. Once this is clear, then the system administrator's task of securing the operating system will be more manageable.

The management of security in a Unix operating system can be broken into three key components. The first, and perhaps most critical, factor is the prevention of unauthorized system access. If an intrusion does occur, it is important to know about it as soon as possible. Intrusion detection is the second key factor in security management. After a break-in, it is necessary to clean up the system, and verify that no corruption or re-entry points have been left behind. System recovery is the third component of security management.

Intrusion Prevention

The prevention of system intrusions is perhaps one of the most publicized topics today. With the rapid expansion of the information super highway, information technology managers have the difficult task of gathering all of this data, without exposing their own systems. In order to effectively reach this goal, a security plan should be formulated. This plan should contain a definite strategy on how access should be granted, as well as how any adverse situation should be handled.

Much of the security of a system can be dealt with through a few simple practices. Access for authorized users should be kept to a minimum, and granted on an "as needed" basis. It is easier to increase the access for a user than it is to remove privileges that were mistakenly granted. This access is controlled by membership in groups in the `/etc/group` file, and by permissions of files and directories in the filesystem. System directories, such as `/etc` and `/bin`, should never be writable by any login other than root. Otherwise, components of the operating system may be removed, or replaced with corrupted version. The home directory for root should also be included in this list. For the most part, if a directory was created by the operating system install, it should only be modified by the operating system, excluding `/tmp` and `/usr/tmp`. User access can also be controlled by the choice of shell that the accounts use. In addition to the Bourne, Korn, and C shells, the `rsh` and `rksh` shells are also available under HP-UX. The restricted Bourne and Korn shells perform much like their predecessors, while hiding parts of the operating system from the users.

Since the main philosophy behind the development of Unix was the sharing of information, it is difficult to take away the operating system without creating adverse effects. This creates some of the problems with security. If a hacker knows what system is being used, then the potential invader will have a better idea of commands to use to attempt to break in. The most common place to start are the `/etc/passwd` and `/etc/group` files. From these, it is possible to determine what accounts exist, what access they have, and if passwords have been set. This is a key reason for keeping user access limited, as it can be expected that end-users will not be as security conscious as systems administrators. Through SAM, it is possible to turn on security measures to help protect this vulnerability. One of the important safeguards here is the generation of a shadow password file. The encrypted password string is replaced with an "*" in `/etc/passwd`. Therefore, normal functions such as `finger` will still work, but attempts to crack the password will be thwarted. The real password file is moved under a directory call `/.secure`, which is only accessible to root. Since nobody else can read the encrypted password string, they cannot attempt to decode the password through the use of a popular public domain software package, known as "Crack". It is also a good idea to enforce password aging. This is done in the encrypted password field, where a two character entry is placed after the encrypted string, with a comma acting as the separator. The two characters in the aging string represent the mandatory frequency of password changes and the minimum time that must pass between changes. This measure acts as a deterrent to brute force methods of password breaking.

Another piece of information, which may be dangerous to announce, is created on the initial system set up. A file named `/etc/issue` is generated during the installation of the operating system. This file is displayed by `getty`, unless told to do otherwise in `/etc/inittab`. The problem with this file is that the initial default is to display the hardware and operating system versions in use, which is valuable information for someone attempting to crack the system. This file should either be emptied, or replaced with a message advising of the legal ramifications of an unauthorized access.

When a system is connected to a local area network, there are another group of files that are added to the list of critical system files. The `/etc/services` file specifies which internet services will be available from that particular host. Two more files, `/etc/inetd.conf` and `/usr/adm/inetd.sec`, control the usage of these services. The `inetd.conf` file specifies which program will be called for a particular service, such as a telnet request will be serviced by `/etc/telnetd`. The usage of these services may be restricted by using the `inetd.conf` file. Here, it is possible to allow or deny usage of a service to a particular host. The Hewlett Packard default is to allow global access to any service not specified in `inetd.conf`. This also makes `/etc/hosts` a critical file, if any restrictions have been placed on specific hosts. Access over TCP may be made easier through the use of `hosts.equiv` and

.rhosts. When a remote system is listed in `/etc/hosts.equiv`, any user common to both systems may use the `remsh`, `rcp`, and `rlogin` commands to access the local system from the remote without being required to provide a password. This functionality is not allowed to any super-user accounts. A `.rhosts` file in a user's home directory performs the same function, but on a "per-user" rather than a system-wide basis. This file may be configured to allow any user from any known host to access the local system as that user through `rlogin`, without requiring a password. The `remsh` and `rcp` commands would also be operational. Unlike `hosts.equiv`, root may take advantage of the use of a `.rhosts` file. This is a very important reason for making sure that no accounts can write to root's home directory, as it would be dangerous to allow users to create their own `.rhosts` file into the root account. Individuals should be advised to take similar precautions on their home directories.

In addition to software, there are two hardware entry points that should be noted. Unattended terminal sessions are perhaps the largest vulnerability on any system. A login session for the root user should never be left alone. While away from the terminal, the superuser session should at least be put into *lock*. A better idea would be to logout, and re-enter the system when it is again needed. The Korn shell contains the functionality to enforce non-interactive logouts with the use of an environment variable. When **TMOUT** is set, an unused session will be terminated in the number of seconds determined by this variable. The timeout will be enforced when a port contains no processes other than the shell. Modems are another area of concern. When possible, they should be brought through a terminal server, a router, or some other piece of hardware that is capable of screening out unwanted connections. If a modem is to be connected directly to the serial port or mux strip, then password protection should be used.

Password protection of a modem line is enabled by the creation of two files, `/etc/dialups` and `/etc/d_passwd`. The first is a list of all ports that have a modem directly attached. An example of this file would be:

```
#Modem port assignments
#
/dev/ttyd0p5           #mux port 5
/dev/ttyd0p6           #mux port 6
/dev/ttyd00            #serial port 0
```


The `/etc/d_passwd` file looks similar to the `/etc/passwd` file, except that the user names have been replaced by shell program names. An example of this file would look like:

```
/bin/ksh:/48UqDFc.kAFI:0:0::  
/bin/sh:84rOfhjarZcsm:0:0::  
/bin/csh:t4d/m84.095Fg:0:0::  
/usr/bin/keysh:*:0:0::  
/bin/posix/sh:*:0:0::
```

To create this file, use any editor to add all entries except for the encrypted password string. Then, execute the command `passwd -f /etc/d_passwd < shell name >` for each shell listed. Any shells that should never be used over the modem lines can be denied by adding a `*` to the encrypted password string field, such as `/usr/bin/keysh` and `/bin/posix/sh` above. Any shell not listed in this file is assumed to be unprotected. In this example `/bin/rsh` and `/bin/rksh` would not require a password. With this functionality enabled, the modem users will see two password prompts. Once a normal login sequence has been successfully completed, the operating system will then detect a protected shell being run on one of the listed ports. This will generate a challenge for another password, at which time the password for the shell is given. This method will allow at least a degree of filtering on modem access, but it should be remembered that it is only secure if the password distribution is kept to a minimum.

More than anything, securing a Unix operating system requires a modification to regular daily habits. If you want to make sure that your system is not subject to intrusion, then make sure that your data is restricted to the users that you intended it for. A little extra care in checking permissions, and cleaning up behind yourself will go a long way towards securing your system.

Intrusion Detection

No matter how much time and effort you invest in securing your filesystem, it is never safe to assume that it is totally impenetrable. It is vital to perform periodic checks for intrusion. Early detection is important to damage control. The big question is how to determine when your filesystem has been breached.

First, always know who is on your system, where they are connecting from, and what they are doing. The commands `who -u` and `w` show everyone that is currently logged in, and the IP address that they have connected through. The latter also indicates the current command being executed by each user. Periodic use of these will give you a general idea of what is occurring on your system.

Anything that seems to be out of place may be further examined by using the *ps* command. With this utility, it is possible to track all activity to a particular port or user. The use of *vi* on one of the determined system files should be questioned. Viewing the */etc/passwd* file should also be of possible concern. Another method of activity tracking is made available when using the Korn shell. Each user has a *.sh_history* file in their home directory, which serves as a record of the commands that they have executed. The one drawback to this method of tracking is that the file, in order to be used, must be modifiable by the user. If it is known that this file is in use, it may be edited to cover activities.

An intrusion into the system is often brief, and may not be detected during the incident. However, anyone who manages to break in will be likely to leave a re-entry point for future use. This is where the system files become critical. An intrusion may often be detected by changes in such areas as the password file, a root *.rhosts* file, or SUID programs. Management of these files may help maintain system security, as well as provide for recovery in the event of tampering. The most effective way to provide for this is to create a backup copy of your critical files in a safe place, such as a hidden directory or on removable media, if possible. Intrusion detection would be performed by comparing the contents of the live system file to the backup you have saved. Some differences may appear, such as the encrypted password string for a user who has recently changed passwords. But, most of the data should remain constant. The drawback to maintaining the full set of files is the space that is required. Another good way to verify the integrity of the system files is to create a file with the *sum* of each system file. For */etc/passwd*, it will be necessary to create an intermediate file with the encrypted *passwd* and the *gecos* stripped out before performing a checksum. This will help eliminate the inconsistencies created by modifications that users are allowed to make. The checksum method will give a reasonable level of confidence that the information is correct, and requires less space than maintaining a second copy of all files. However, it does not provide the recoverability the first method offers.

SUID programs are another source of concern when securing a system. There are some programs which must execute as root, and therefore have a permission string that resembles "r-sr-xr-x" when a *ls -l* is performed on the file. When checking the filesystem, it is important to make sure that no such file is owned by root and writable by anyone other than root. The *find* command has the options of *-perm* and *-user* which can be used to help find such programs. It might also be advisable to maintain a list of known SUID programs, and look for any that cannot be identified. If a system has been breached by a local user, this would be the easiest way for that individual to obtain root access at any time. One example of how this may be used is for a user to make a copy of */bin/ksh* in their own home directory. Once they have breached the root account, they may change the ownership of this file to root, with the permissions allowing SUID execution.

Thus, anytime they access their special version of ksh, they call a Korn shell with superuser privileges.

The `/usr/adm` directory also contains several useful tools for system monitoring. The `sulog` file shows all attempts where a user executes `su` to become another. Each entry contains the original user, the user that was switched to, and a record of whether the switch was successful or not (indicated by a `+` for yes, or a `-` for no). This directory also has a `shutdownlog` file, which holds a completed record of all known halts, reboots, and panics. The only activity that cannot be recorded is the powering off of a live system. This is important because a system is vulnerable during the power-up sequence. If someone has the ability to shut your system down and to reach the console on the power-up, then they may secure superuser privileges by interrupting the boot sequence and bringing the system up in single user mode. The shutdown log is the place to start searching if this activity is suspected. The `/etc/shutdown.allow` file may also provide a clue, since membership in this file is required to perform a normal shutdown. Another highly useful file in the `/usr/adm` directory is `syslog`. This file contains a complete listing of all system notices, warnings and errors since the last reboot. It also hold information on all TCP connections that have been made. Any session (`rlogin`, `telnet`, `ftp`, etc...), is recorded in `syslog` with a unique identification. Thus, that session may be traced for activity. This may be especially useful for determining which files were transferred by an `ftp`.

Once an intrusion has been detected, the number one question will be how to return the system to a normal state. This process will be very much dependent upon the amount of corruption that occurred before your visitor was noticed. This is where a plan of action becomes important. The first step is to make sure that the intruder is gone. If not, take whatever measures necessary to eliminate the threat. Next, change the root password for your system. It is possible that either your password was broken, or that the intruder changed it to see if you would catch this. Then, begin a sweep of the system in search of other re-entry points, such as `.rhosts` files and SUID programs. Once you believe that your system is secured, then you should follow the logs to determine how the entry was made. The files noted in the previous sections should be able to be combined to form a picture of the intruders login session. From here, you may be able to determine who this user was, and take appropriate action. If not, you may at least come away with useful information on how your system was breached, so that you may protect against such an attack in the future.

Once this information has been gathered, you must then determine what recovery steps are necessary. This will require a verification of your specific application data, to insure that corruption has not occurred. If data corruption is found, then you may wish to restore your system to the last set of successful backups prior to the detection of the intrusion. For a more thorough install, you

may even want to reinstall the operating system, and then restore your data from backups. This will help remove any undetected "backdoors" that may have been left behind. This is the most critical case for a recovery.

Most normal recoveries will require the removal of tampering from system files. If you have chosen the method of maintaining a complete set of system files for intrusion detection, then your recovery is simply to copy the known good files into place. If the checksum method was used, then you will want to read through all of your system files to insure that they are correct. Once all repairs have been made, then the checksum file should be updated, since one extra space in the file will change the checksum.

Conclusion

The most important part of system security is the amount of importance and consideration that it is given. Your operating system will only be as secure as the users allow it to be. This is a project that will require compliance by all who are granted access. Otherwise, your efforts will produce results, but not as effectively as you may desire.

For the systems administrator, security is an ongoing process. Monitoring does not necessarily need to be a constant process, but it should be done frequently. As much as we would like to think that our systems will never be breached, we should be prepared for this event to occur. Despite all efforts to thwart intruders, it must be recognized that this is a possibility. The goal is to be able to recover from an adverse situation, while minimizing any known points of exposure.

Building a Tool for Remote Unix Administration

Jeff Hodges, Bobby Siu and Mike
Milkovich

Abstract

As companies migrate from centralized proprietary systems to distributed "open" systems, a need has arisen for tools that allow centralized management of distributed systems. Due to the increased number of systems and their geographical distribution, core system administration activities such as backup, restore, user maintenance, and security checking become unmanageable. System administrators need a secure, reliable mechanism to perform these activities efficiently and effectively from a remote location. In this paper we will discuss and demonstrate several architectures of remote system management tools for Unix environments. We will discuss the major considerations, such as security, reliability, and maintainability, associated with each design. We will provide example code where relevant and will construct a basic tool from standard Unix functions.

Introduction

The proliferation of the Unix distributed computing paradigm has created a huge demand for new tools and techniques to manage applications and the machines and networks they run on. Today's applications are designed to take advantage of many different resources spread across a network (the application on the desktop, a workgroup compute server or a print server in the local

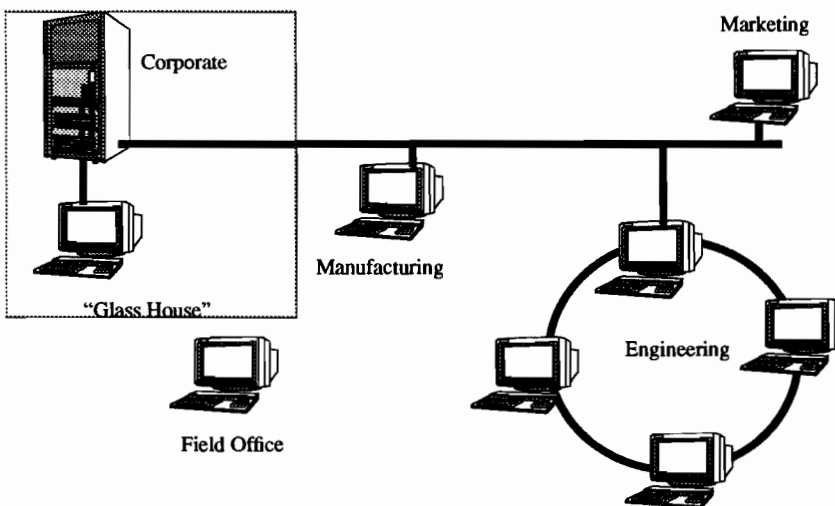
office, a corporate database at headquarters). To work reliably (or at all) these applications require a stable, functioning foundation. This foundation includes not only the local computer hardware and operating system, but also functioning local and wide area networks, functioning remote servers, and the services the remote servers provide. A network database server, for example, must not only have running hardware and OS, but must also have a functioning database engine with a working network connection. Today, a machine is not "up" unless the service it is supposed to provide is available.

Not long ago, a company with a Unix system managed it by hiring a Unix guru. The Unix box and its attendant were hidden off in some dusty corner of the office, while the "real" business machines were kept locked behind glass walls and run by an army of ITC people. Today, most of a company's computing power is found on desks, in closets, and occasionally in computer rooms. This dispersion of computing resources has increased the complexity of system management. The increased complexity of today's applications has increased the management load still further. In addition, today's business environment has forced many IT departments to limit or even reduce staffing.

Standard system management practices involve performing a set of tasks on the managed machine. A task might be to perform a backup, add a user account, or check that the printer queue is working. In our distributed environment, you need to do all of the tasks repeatedly on each machine administered. In addition, the system boundaries are extended out beyond the "pizza box," beyond the "glass house," and often beyond the borders of the building. (Figure 1 on page 3)

FIGURE 1.

The administrator's responsibility extends beyond a box or a glass house.



As our system grows, the complexity of administration sky-rockets. For example: Suppose we are performing regular backups, disk space monitoring, user administration, and print management on 3 Unix workstations. This gives us a total of 12 daily tasks. If we add two new workstations, our task count jumps from 12 to 20.

So the question facing system and network managers everywhere is, "How do I manage my users' machines with only 24 hours in the day and the machines scattered to the four winds?" The answer is a combination of smart processes and some simple tools.

Requirements

Distributed Unix administration tools must meet the following three requirements:

-
- Must handle standard system administration functions. These include such broad categories as backup and recovery, account maintenance, queue management, disk management, security monitoring, and process management.
 - Must be able to perform these functions on many remote machines from one “management console.” Tasks may be run on one machine at a time or on many machines simultaneously. The remote machines may be of the same type or of many different types, depending on your environment.
 - Must provide security equal to or greater than the standard access methods of telnet and rsh/rxexec.

In addition, the following abilities are highly desired:

- The monitored systems should be self-maintaining and self-scheduled. They should not have to wait for instructions from the master system.
- Monitored systems should run on an exception basis: the system manager should be alerted only if the problem cannot be handled automatically.
- There should be an easy way to extend the capabilities of your tools. You should be able to add tasks to an already managed machine, or extend your toolset to a new machine.
- Must have the ability to maintain consistent versions of software, scripts, and data and control files on each system monitored.

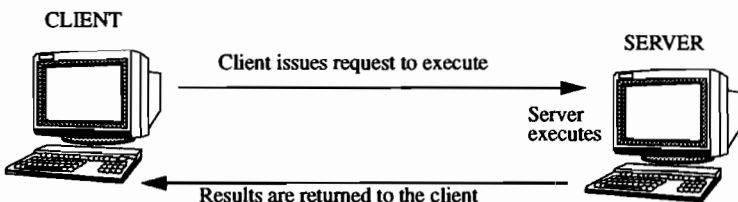
Once you have a “framework” with the capabilities listed above, you can start building your new system management processes to take advantage of your new framework.

Architecture

The basic model for remote operation is a client/server model. In this case, the “client” is the management station, and the “server” is the managed machine. In a typical environment, you will have many managed servers, and a few (or only one) management clients. To execute administrative tasks, the client contacts the server and issues a request for the server to execute the task. The server then validates the client and attempts to execute the task. The results of the task are gathered and sent back to the client. An example of this is shown in Figure 2 on page 5.

FIGURE 2.

A simple client/server model of remote operation.



The Management Environment

In the management environment, our toolset has 4 functional areas: the actual tasks to be executed; communications & security; execution control and evaluation; and scheduling.

Tasks

A task is simply some administrative or system management duty you want to perform. For many system management tasks, you can automate the drudge work, freeing you to concentrate on the more interesting parts of your system (i.e. the broken parts). We use the word "task" to mean either an administrative duty, or a script or command or program we have written to perform that duty. An automated task might be as simple as "watch the disk space usage on the root drive and scream when it gets above 95% used," or it might be as complex as "check that the accounting department can access the customer database with a reasonable response time. If not, fix it." Depending on the capabilities of your server, you might have tasks that automatically run at a given time (self-initiating tasks), tasks that run only if a certain set of circumstances are true (threshold-initiated tasks), or tasks that take actions on other machines (proxy tasks).

Communications & Security

There are many communications methods to choose from, including `remsh/rexec`, Berkeley sockets, and RPCs. Each of these methods has advantages and disadvantages; the method you choose will depend on such factors as ease of use, level of security, portability, network overhead, and reliability. You will also need to decide if you want to send a full Unix command line across the network, just a request for a task, or some combination of task name and command parameters?

`remsh/rexec`

`Remsh` and `rexec` are probably the simplest to use and run on many platforms. However, they are relatively insecure, and performing complex tasks is non-trivial. These tools are easy to write and maintain, and can often be pulled together from the scripts and tools you already have. The security risk for `remsh` lies in the fact that if the management node is compromised by a hacker, then all of the managed nodes are accessible. `Rexec` is more secure in this regard, but its failure lies in the fact that it sends the user's password in clear ASCII text when it requests that a task be executed. If you are developing tools for a small group, the functions you want to automate are relatively straightforward, and you can live with the potential security problems, this method is probably the way to go. (For an introduction to the problems and solutions to `remsh` security, and system security in general, see [Practical Unix Security](#), by Simson Garfinkel and Gene Spafford, O'Reilly & Associates, Inc., 1991. ISBN 0-937275-72-2) Figure 3 on page 6 demonstrates one of the potential security risks for systems using the `rexec` functions.

FIGURE 3.

Lan Probe output demonstrating a potential `rexec` security hole.

```
Pkt# 4 Len: 64/64 Thu Jul 14 14:11:19 1994 PDT
Ethernet: (0x080009273e24 -> 0x080009627490) type: IP(0x800)
Internet: 15.3.33.201 -> 15.3.33.148 hdr len: 5
  ver: 4 total len: 45 TypeofService: 0 Precedence: (Routine)
  Nrml Reliability Nrml Thruput Nrml Delay id: 0xe681 fragoff: 0
  Time To Live: 30
  flags: 00 (May frgmt, Last frgmt) prot: TCP(6)
  prot: TCP(6)
TCP: 1853 -> rexec(512) seq: 464d1207
  ack: 75f63e01 win: 8192 hl: 5 xsum: 0x36f8 urg: 0
```

and server are using the same version of RPC. For a large application this may entail patching hundreds of server nodes. (For more information on RPC see *Power Programming with RPC*, by John Bloomer, O'Reilly & Associates, 1992. ISBN 0-937175-77-3.)

Both RPCs and BSD Sockets have the advantage of increased security based on the fact that the protocol is programmer defined. A hacker would need to know some underlying details of the application to break into the system. A programmer could further complicate the hacker's life by encrypting some or all of the data in the packets. Programmers could also use a reference table of keys for each task to execute. Rather than send the actual command across the network, a corresponding key value known only to the server and client could be sent. Upon receipt of the value, the managed node would then reference a table with the actual command. Figure 4 on page 8 illustrates this principle.

FIGURE 4. An example of using a key reference table

```
Pkt# 26 Len: 171/171 Thu Jul 14 17:15:25 1994 PDT
Ethernet: (0x08000962dd3e -- 0x08002008cf27) type: IP(0x800)
Internet: 15.3.33.146 -- 15.3.33.208 hdr len: 5
  ver: 4 total len: 153 TypeofService: 0 Precedence: (Routine)
  Nrml Reliability Nrml Thruput Nrml Delay id: 0xfb34 fragoff: 0
  Time To Live: 30
  flags: 00 (May frgmt, Last frgmt) prot: TCP(6)
  prot: TCP(6)
TCP: 3987 -- 5557 seq: 4ba92401
  ack: 32c31e01 win: 8192 hl: 5 xsum: 0x71d4 urg: 0
  flags: .ACK--PUSH.
  data (100/113):.....m.....!...0...6...E...R...c.....9/3/360
  425.....15.3.33.146....774231832....TASK1000000
  *-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
  *-*-*-*-*
```

Execution Control and Evaluation

The management station must have the ability to control the execution of the task on the managed node, as it would a task run-

ning locally. It must be able to start, stop and evaluate the results.

Remsh provides the ability to start and stop processes in the same fashion as one would do locally. However, remsh does not give you the exit code of the remote process, so you need to write an output parser to determine the results. This parser can be a simple customized grep for a key word or phrase, or a generic, reusable, configurable regular expression parser.

Sockets and RPC calls give you the ability to collect output as well as exit codes from a process, and in addition, give you job control (signals, stopping and starting the process, and so on). Since you have more control over the data flow to and from the process, it is possible to have a customized "interaction" with a process (an example is an expect script). You can also evaluate the output from the process "on the fly," letting you take immediate action for a given output. For example, you can watch a backup script running, and when this backup script puts out the message, "end of this tape. insert another," your monitoring routine can run another program which causes an autochanger mechanism to change the tape.

Task Scheduling

Task scheduling is the ability to schedule the execution of a task at a predefined time or interval. For example, you may wish to perform a backup every night at 2:00 AM, or to check for security violations every 30 minutes.

Task scheduling is desired because it:

- Reduces the chance for human error.
Humans are not adept at performing repetitive tasks. When faced with performing the same task time and time again, the chance that a human will make a mistake increases each time.
- Reduces cost by eliminating manual labor.
The cost for human labor is constantly rising. By reducing the need for manual labor, cost savings can be realized today and in the future.

Task scheduling can be achieved by utilizing a scheduling facility, such as `cron(1M)`, on the management station or the managed systems.

Task scheduling control can be placed on the management station side by placing the entire task execution request in the cron file on the management station. For example, if we wanted a backup to be performed on systemX every night at 2:00 AM, we could add the following to the management station's cron/scheduling facility:

```
*0 2 * * * remsh systemX -l root /bin/doBackup*
```

Task scheduling control can also be placed on the managed systems to distribute the control to the managed systems. For example, the following can be added to each managed system's local cron file:

```
*0 2 * * * /bin/doBackup*
```

The advantage of centralizing all scheduled task control on the management station is that it is easy to maintain. There is a single local file to maintain. The disadvantage is that this method increases the amount of network traffic because all requests to execute a task must be sent across the line from the management station to the managed systems; additionally, if the line becomes unavailable the scheduled tasks cannot be invoked.

Distributing scheduling control to the managed systems requires less network bandwidth than centralized control and provides the added benefit that scheduled tasks will still be invoked if the network becomes unavailable. The disadvantage is that it can be more difficult to manage the distributed configuration.

A Basic Tool

To illustrate the architecture components we have discussed, we will now build a basic remote Unix administration tool. For simplicity, we will use the remsh mechanism as the communications/security component. Execution and evaluation will be handled by a combination of remsh features and a custom script. For scheduling we will use the local cron(1M) feature and a scheduling script.

Establishing Communication

The management station is responsible for the execution and evaluation of all of the tasks run on the managed nodes. To allow

the management station this functionality we will need to do the following:

- Enable the `remshd` process on all managed machines.
- Add the user `ruxad`, with a user id of 0, and a home directory of `/tasks` to all of the managed machines.
- Create the directories `bin`, `scripts`, `data` and `cntl` in the `ruxad` user's home directory.
- Create the file `.rhosts` in the `ruxad` home directory and make the following entry: `<Management Station> ruxad` where `<Management Station>` is the hostname of the system executing the tasks. For example:

```
hpnst123 ruxad
```

Now that we have the communications mechanism in place we can add the execution and evaluation component.

Executing and Evaluating

Since we are using the `remshd` mechanism we will need to encapsulate the output of our managed nodes' tasks. This will allow us to take specific actions based on the exit values of the process we are running. We will also want to execute tasks on multiple systems at a time. Figure 5 on page 11 displays the controlling process `Rem_run`.

FIGURE 5.

`Rem_run`: an example controlling process

```
#!/bin/ksh
#####
#
# Module Name: Rem_run
#
# Purpose:
# The purpose of this script is to execute a command on a number of
remote
# systems. The output of the command executed will be searched for
the
# keywords "Informational:" "Warning:" and "Critical:". If the key-
words
# are found in the stderr output of the command executed, an error
message
```

```
# will be returned from this script.
#
# Parameters:
# -h = Names of the systems where the command will be run on.
# -c = command to be run
#
# Exit Codes:
# 0 = Success
# 1 = Informational
# 2 = Warning
# 3 = Critical
# 9 = remsh failed
#
# Example Usage:
# Rem_run -h "hpnst sage basil" -c "/bin/ps -ef"
#
#####
#####
#set -x
set -u

#####
#
# Function Name: monsh
#
# Purpose:
# The purpose of this function is to execute a command on a remotely
managed
# system. The output of the command executed will be searched for the
# keywords "Informational:" "Warning:" and "Critical:". If the key-
words
# are found in the stderr output of the command executed, a return
code
# of 1, 2, or 3 will be returned from this script.
#
# Parameters:
```

```
# $1 = Name of the system where the command will be run on.
# $2 = Command to be invoked
#
# Exit Codes:
# 0 = Success
# 1 = Informational
# 2 = Warning
# 3 = Critical
# 9 = remsh failed
#
# Example Usage:
# monsh hpnst checkDisk
#
#####
#####
monsh()
{
INFO="Informational:"
WARN="Warning:"
CRIT="Critical:"

SYSTEM=$1
CMD=$2
OUT="/tmp/job.stdout"
ERR="/tmp/job.stderr"
TMP="/tmp/job.output"

/usr/bin/remsh $SYSTEM -l ruxad "$CMD" > $OUT 2> $ERR
if [ $? != 0 ]
then
return 9
fi

grep $CRIT $ERR > /dev/null
if [ $? = 0 ]
```

```
then
echo "$SCRIP $SYSTEM $CMD Failed:"
return 3
fi

grep $WARN $ERR - /dev/null
if [ $? = 0 ]
then
echo "$WARN $SYSTEM $CMD Failed:"
return 2
fi

grep $INFO $ERR - /dev/null
if [ $? = 0 ]
then
echo "$INFO $SYSTEM $CMD Failed:"

return 1
fi

cat $OUT

return 0
)
#####
# Main
#####
NUM_ARGS=0
HOSTFLG=0
CMD_FLG=0
USAGE_MSG="Usage: $0 -h hosts -c command "
while getopts :h:c: option
do
case "$option"
in
```

```
h) MACHINES=$OPTARG
HOST_FLG=1;;
c) CMD=$OPTARG
CMD_FLG=1;;
\?) echo $USAGE_MSG
exit 1;;
esac
done

# Check that all required parms have been supplied
if [ $HOST_FLG -lt 1 ]
then
    echo $USAGE_MSG
    exit 1
fi

if [ $CMD_FLG -lt 1 ]
then
    echo $USAGE_MSG
    exit 1
fi

for HOST in $MACHINES
do
    monsh $HOST "$CMD"
done

exit 0
```

It is clear from the example that we could extend **Rem_run** to take specific actions based on the exit status of the remotely executed function. This architecture requires that we encapsulate the output of each remotely executed script but with the added control the effort should be worth it.

Executing on Time

In order for the system to be fully functional we will need the capability of scheduling the execution of the remote tasks. To reduce the amount of network traffic required and the dependence on the management station we will use the managed systems' **cron(1M)** function and the script in Figure 6 on page 16.

FIGURE 6.

Scheduling example

```
#!/bin/ksh

#####
#####
#
# Name:
# updateCron
#
# Purpose:
# The purpose of this script is to update a cron file on a remote system.
#
# Parameters:
# -c = Entry to be added to the cron file. eg *0 2 \* \* \* /bin/doBackup*
#
# Exit Codes:
# 0 = Success
# 1 = Failure
#
# Example Usage:
# updateCron -c *0 2 \* \* \* /bin/doBackup*
#
#####
#####

CMD_FLG=0
USAGE_MSG="Usage: $0 -c command *
while getopts :c: option
```

```

do
    case "$option"
    in
    c) CMD=$OPTARG
       CMD_FLG=1;;
    \?) echo $USAGE_MSG
       exit 1;;
    esac
done

# Check that all required parms have been supplied
if [ $CMD_FLG -it 1 ]
then
    echo "Critical Error:$USAGE_MSG"
    exit 1
fi

TMP="/tmp/cronfile.old"
crontab -l - $TMP
echo "$CMD" >> $TMP
crontab $TMP
if [ $? != 0 ]
then
    echo "Critical Error: updateCron failed..."
    exit 1
fi
exit 0

```

It is obvious that this script would need to be on the system prior to using it through `Rem_run`. To accomplish this we could simply ftp the script to each system we are using, or we could make use of the remote copy (`rcp`) function. Using `rcp` we could automatically update all of the systems we are managing. The code fragment illustrated in Figure 7 on page 18 illustrates the use of `rcp` for software distribution.

FIGURE 7.

Example use of rcp

```
.  
.  
#  
# Example to redistribute all the tasks for ruxad  
#  
MACHINES="hpnst hpnst123 hprew"  
DIR=/tasks  
for HOST in $MACHINES  
do  
/usr/bin/rcp -r $DIR $HOST:$DIR  
done
```

Conclusion

Remote Unix administration is a very complicated process; system administrators are faced with obstacles never encountered in the traditional "glass house." If system administrators are to be successful in their function they will need to employ some sort of administrative tool. In the preceding pages, we discussed the major requirements of such a tool, some possible architectures, and a basic tool to learn from.

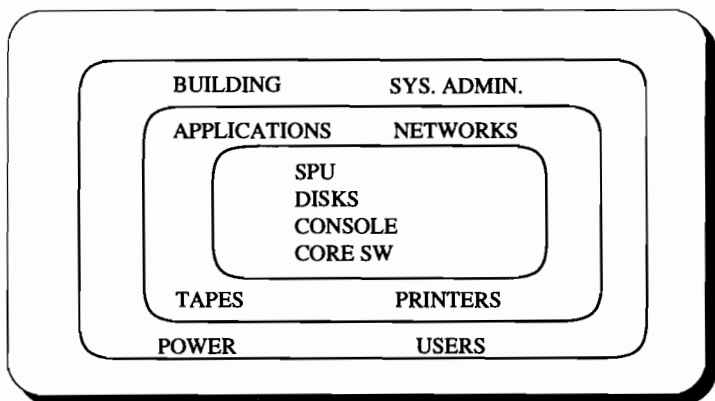
PAPER # 8003
HIGH AVAILABILITY, MIRRORING, RAID
A HOW-TO SYSTEM MGMT CASE STUDY
RAYMOND W. RIEDEL
HEWLETT-PACKARD COMPANY
250 N. PATRICK BLVD. SUITE 100
BROOKFIELD, WISCONSIN 53045
(414) 879-2242

Currently there are many businesses who base their operation and their competitiveness on mission critical applications, applications that are very important in the day to day operation of a company. In turn, there is a greater demand for greater system availability. I have been involved with several organizations where system downtime has caused some serious problems. These problems include loss of customer confidence, incomplete or late work, and loss of business opportunity. This paper will discuss how these "high availability" concerns can be addressed, and some specific customer cases utilizing some of the solutions available today that could provide you with greater system availability.

High Availability/Case Study

Typically when planning your computing environment, there are several issues that must be considered.

These issues include capacity, scalability, performance, functionality, and **AVAILABILITY**. When including high availability solutions in your environments, you must insure that these solutions are cost-effective and compatible. But what exactly is a Highly Available System? I would look at a system this way:



When I assisted several customers in configuring high availability and redundancy into their systems, the same three crucial requirements surfaced at every meeting:

- * Data integrity, availability, and recovery
- * Minimal or no planned downtime
- * Minimal unplanned downtime

However, "highly available" can mean different things to different customers. One of my customers defined it as "better than the system I used to have" or another described it as "having to deal with the system being unavailable only between 12:00am and 2:00am every other Sunday". Highly available must meet **your** expectations, and not one of these definitions.

My clients decided that for their particular operation, very minimal planned downtime would be necessary due to the nature of their business. They also wished to have some form of data redundancy in addition to SPU failover capability. Because hardware is only one piece which may contribute to system failures, it is usually not enough to address the issue of system availability with only redundant processors.

This is the main reason my client wanted more capabilities which eventually led the discussion on mirroring and RAID disks

I feel it might be appropriate to discuss in some detail why some of my clients chose to use RAID technology as their data preservation solution and why some chose to implement a "mirroring" solution. One trend that is very evident today is the migration of mission-critical operations away from mainframe platforms and toward open systems. Many customers agree that the disk subsystem is still a fragile component of a computer subsystem, and therefore have become the focus of redundancy solutions.

A disk failure can not only be very costly and time consuming but catastrophic for a business. A customer that I have been working closely with has seen first hand how devastating a disk problem could be. My client's main business is catalog ordering via the telephone. During one their busiest days, one of their disk drives failed resulting in lost orders.

Basically, RAID technology uses parallel disk drives to achieve higher data availability. The redundancy of the disk drives in RAID ensures that no data is lost due to the failure of a single drive. As with any technology, there were questions from my client about the benefits and costs of RAID as compared to the benefits and costs of *mirroring*. For any given situation, the user must make choices among three technical features: fault tolerance level, performance, and capacity. All these features must be used to determine the cost of the solution.

Mirroring on HP-UX allows up to three copies of data on separate disk drives so that the data is still intact after any single disk failure or disk interface card failure. While RAID disks provide data integrity via encoded parity across all the drives or dedicated parity on one disk drive using approximately 5% more disk space, *mirroring* requires up to 100% more space to mirror the data. However, mirroring allows not only data redundancy but hardware redundancy as well. My client decided that regardless of the price differential between the two solutions, mirror disk would be the solution of choice. They felt that this would provide them with the highest level of system availability.

There are several advantages to disk mirroring which also should be mentioned. Mirroring is typically transparent to the application. There are usually no application modifications required. The biggest advantage my client was that mirroring allows you to take a disk off-line to perform a backup, while applications continue to access data from the disk which remain on-line. All changes made to the on-line disk are maintained in memory and used later in the synchronization process when the backup disk is brought back on-line. In a three-way mirroring scenario data availability is maintained in case a disk failure were to occur during the backup procedure. This was a key point in my discussions with my client during our discussions about backup and recovery. This synchronization process is very important in case of a system crash. It would be up to the Logical Volume Manager to insure that all copies of every extent in the mirror disks are identical. As stated earlier, *mirroring* provides a much higher level of data protection, versus RAID, since all the cables, controllers, etc. are duplicated. This prevents the possibility of any of these hardware pieces becoming the single point of failure.

Discussing data availability is only one component of

High Availability/Case Study

8003-6

of the "high availability" solution. As I stated earlier, the clients that I have consulted with over the past several months were interested and willing to do what ever it takes to experience little or no interruption of system time. These were typically directives above the Director of MIS level that came about after first hand experiences in system down time or clients basically planning for the future.

SPU redundancy would be the other component of the "high availability" solution. My initial directive was to provide my client with not only data redundancy but system redundancy as well. This next level of high availability is addressed by a SPU "switchover" product such as SwitchOver/UX. SwitchOver/UX is a combination of software functionality and hardware interconnection that enables your computer system and the applications that run on it to recover from a failure. After proper configuration, the software automatically detects the failure of a processor by monitoring the "health" of the primary system(s). The "switchover" process causes another processor to "takeover" the disks of the disabled processor.

The replacement processor reboots using the disks from the failed processor. Several of my clients have the following characteristics:

- * Multiple networked systems
- * The business requires continuous operation
- * Databases are used

Because of these characteristics, they were ideally suited for SwitchOver solution.

Customer Case Study

Customer:

Travel Industry

Customer Profile:

Separate services group providing specific cargo services implementing a new mission critical application. The application is a customer service application developed by a third party. The application will be running on a HP9000 H50 and will use MirrorDisk/UX to mirror the data. It was backed up by another H50 with SwitchOver/UX.

High availability was key for this particular application. The application had to be available almost 24 hours a day.

Implementation:

The planning for this installation and configuration was done prior to the ordering of the hardware. The systems were to be installed and SwitchOver/UX configured from the beginning. A half-day was spent reviewing the configuration with the account manager to determine if anything had to be moved or if any other changes had to be made. We initially put together a plan for both the hardware and software installation. In addition to the SwitchOver planning we also had to plan for the Logical Volume Manager and Mirror/UX configuration.



Installation:

HP-UX had to be re-installed since it had come pre-installed on the internal disk drives. After the HP-UX and patch installation, we needed to configure the disks for Logical Volume Manager and for MirrorDisk/UX. The internal disks on both systems were used as dedicated dump devices.

There were also redundant LANS. One LAN connection went to

the company wide LAN. The second LAN connected the two systems together.

Testing:

There are many things that need to be tested during this type of MirrorDisk/UX and SwitchOver configuration:

- * turn off the root disk to make sure that the "mirror" takes over

- * Turn the root disk back on to indicate the system is using the "real" root disk

- * See if the system can be booted from the mirror boot disk

- * Make sure the "heartbeat" daemon is enabled on the primary system

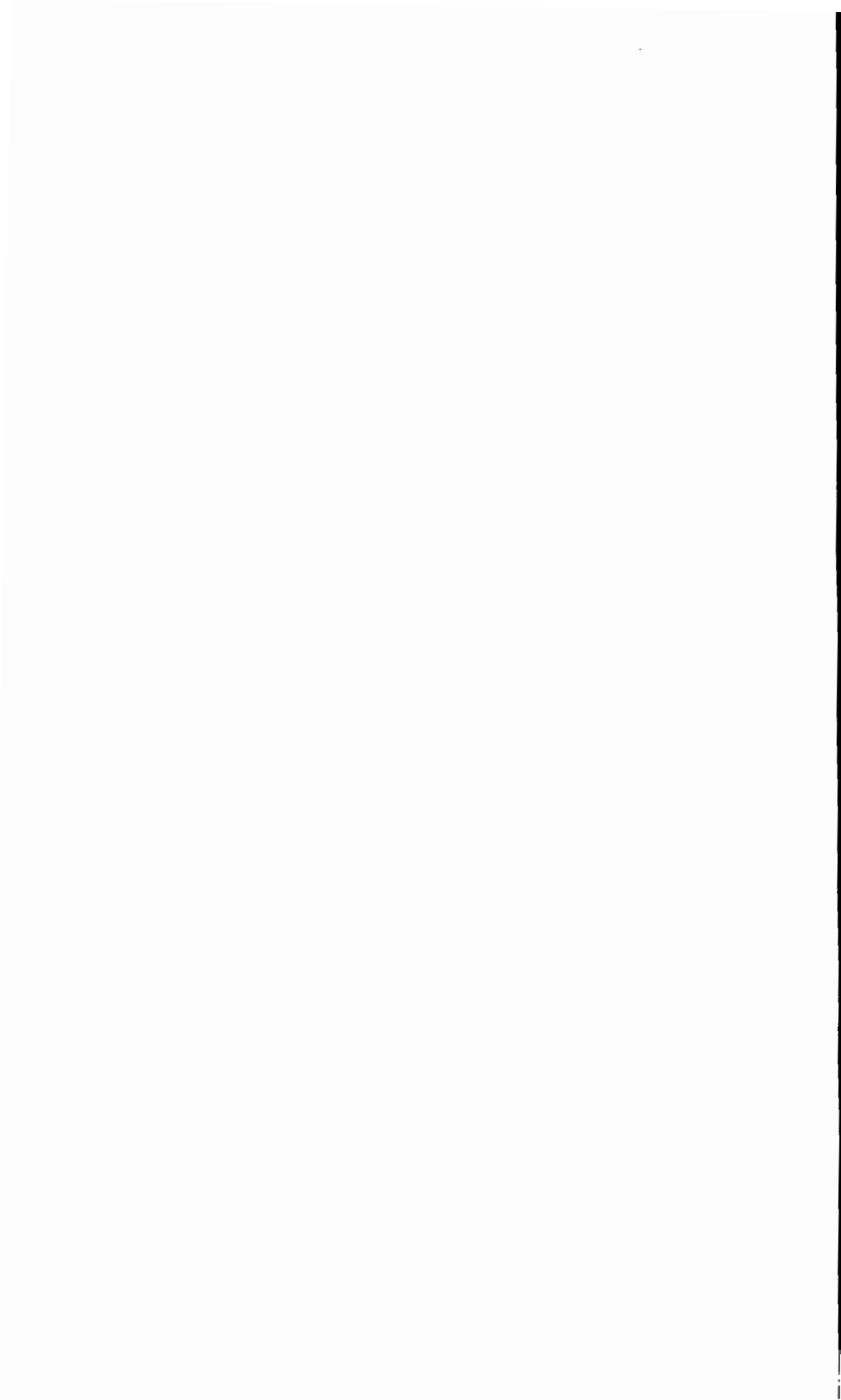
- * Make sure the readpulse daemon is running on the secondary system

- * Power off the primary and wait to make sure the secondary system takes over the disks from the failed system

- * Disconnect the primary LAN to allow the testing of the heartbeat daemon being "heard" on the second LAN

Results:

The overall results of all the testing of this particular client's solution was very positive. All the components of the high availability solution worked fine. It was very important to do a "dry run" of these components in order to make the client feel comfortable that everything functions as proposed in the configuration. Hopefully the my client will not have the opportunity to test these in real life production. But it was good to know that they are capable of providing their users a system which is highly available.



The HP 3000 Workload Manager

Paper #8004

Susan Campbell
Hewlett-Packard Company
Commercial Systems Division
19447 Pruneridge Avenue, MS 47UA
(408) 447-7038
susanc@cup.hp.com

Doug Perry
Hewlett-Packard Company
Commercial Systems Division
19447 Pruneridge Avenue, MS 47UA
(408) 447-6015
dperry@cup.hp.com

Abstract

This paper discusses the new HP 3000 Workload Manager, focusing on capabilities and their usage. The Workload Manager enhances process scheduling on MPE/iX to meet the needs of system managers who require more precise control over the relative performance of system workload components. Specifically, the Workload Manager provides two new features, workgroups and CPU percentages. The Workload Manager provides process scheduling only; it does not address job scheduling, which often depends on a knowledge of job interdependencies and may require scheduling jobs across multiple systems. CPU process scheduling involves allocating the CPU(s) of a system to the processes that comprise the system workload. MPE/iX makes use of a preemptive, shortest job first approximation algorithm for process scheduling. The Workload Manager builds on this algorithm by providing two new features for additional control over system workload partitioning and CPU access. The Workload Manager provides control to the system manager via the introduction of workgroups. A system manager can group processes into workgroups (based on logon, program being run, and/or scheduling queue) and then control the scheduling attributes of workgroups and thus the processes within those workgroups. The second major feature of the Workload Manager is the ability to control the minimum and maximum CPU percentage allocation for each user-defined workgroup. The needs for such features will be reviewed, and sample uses will be discussed.

Introduction

As noted in the abstract, the focus of this paper is the capabilities of the new HP 3000 Workload Manager and the needs they address. With the addition of the Workload Manager, MPE/iX provides the system manager with suitable control to customize their HP 3000 system to handle the particular needs of

their system workload. The remainder of this paper considers various workload management needs, first reviewing the needs and then providing details regarding the Workload Manager features designed to meet the needs outlined.

Needs and Workload Manager Features

The Workload Manager is the result of numerous interactions with customers to understand their needs to effectively manage their system workload. The design team collected information from a wide variety of customers, including those using only HP 3000 systems, those in more heterogeneous environments, and those more familiar with using IBM mainframes. In addition to external customers, input was sought from various HP field and IT personnel. Based on all of this information, the team was able to derive a set of needs and then design controls that would enable a system manager to meet those needs. Throughout the software development process the team has worked closely with our customer partners to ensure that the Workload Manager does indeed meet those needs.

More Specific Grouping

Needs

As the demand for system performance increases, it becomes increasingly clear that the traditional five scheduling subqueues provided by MPE/iX (AS, BS, CS, DS, and ES) are insufficient. In a typical environment, the AS and BS subqueues are reserved for system processes, leaving the CS subqueue for interactive users, the DS subqueue for important batch processes, and the ES subqueue for other batch processes. Obviously this granularity is extremely limiting. For example, the system manager is forced to combine multiple types of interactive users into the single CS subqueue, governed by a single set of scheduling parameters. Important batch jobs might be placed in the DS subqueue (adjusted so that its priority range overlaps with the lower portion of the CS subqueue), or perhaps such jobs are actually placed in the CS subqueue, where they compete with the various on-line users in the CS subqueue.

In addition to requiring more granularity in partitioning the system workload, the system manager needs to be able to more precisely control that partitioning. Currently, the scheduling subqueue of a process can be influenced by many factors, several of which can be controlled by the end-users.

The system manager can establish a MAXPRI for a particular account or user, but the user can manipulate the scheduling subqueue via the ;PRI option on the HELLO or JOB commands, or via the ALTPROC command. Additionally, the subqueue can be altered via the GETPRIORITY intrinsic or the AIFPROCPUT routine. As a result, the user has control over the scheduling subqueue, as long as MAXPRI is not exceeded. Thus, a system manager might give a user a MAXPRI of CS so that on-line work can be done in the CS subqueue, but has also given the user the ability to run a job or a compile in the CS subqueue.

Similar to the MAXPRI available on the user or account, the Link Editor allows the specification of a default and maximum priority for an executable file. The user can specify the subqueue through the ;PRI option of the RUN command, but is constrained by this maximum value.

Thus, there are currently a number of techniques for users to influence the scheduling subqueue of processes they create, and the system manager has limited means of constraining those users making use of these techniques.

Workload Manager Features

The Workload Manager addresses these needs through *workgroups*. While MPE/iX traditionally provided only five scheduling subqueues, the Workload Manager allows a system manager to partition the processes comprising the system workload into an arbitrary number of user-defined workgroups.

Thus, the partitioning can contain as many workgroups as the system manager feels are necessary to control the system workload. The workload might be divided into workgroups based on the particular users doing the work or the programs that are being run. A workgroup could represent individuals with a similar task to perform, users in the same department, or individuals accessing a shared database.

The Workload Manager allows the system manager to define workgroups that truly reflect their view of the system workload, allowing the controlling and reporting of workgroups to correspond more naturally to the way in which the system is used. In addition to the CI interface, the Workload Manager features are accessible through Architected Interface Facility (AIF) routines. Thus, performance monitoring tools such as GlancePlus, LaserRX, and third-party offerings are able to integrate with the Workload Manager.

The Workload Manager intentionally gives full control over the system workload partitioning to the system manager. It is the system manager who

defines the workgroups that are used to control system performance. The system manager dictates which processes belong to a given workgroup by specifying logon information (optional job/session name and user.account), the program being run (in MPE or HFS syntax), and/or the scheduling queue attribute of the process. For example, a logon might be simply user.account (e.g., MANAGER.SYS) or include the job/session name (e.g., "SLC,MANAGER.SYS"). A program executable file might be in MPE syntax (e.g., EDITOR.PUB.SYS) or HFS syntax (e.g., /SYS/PUB/WMTEST).

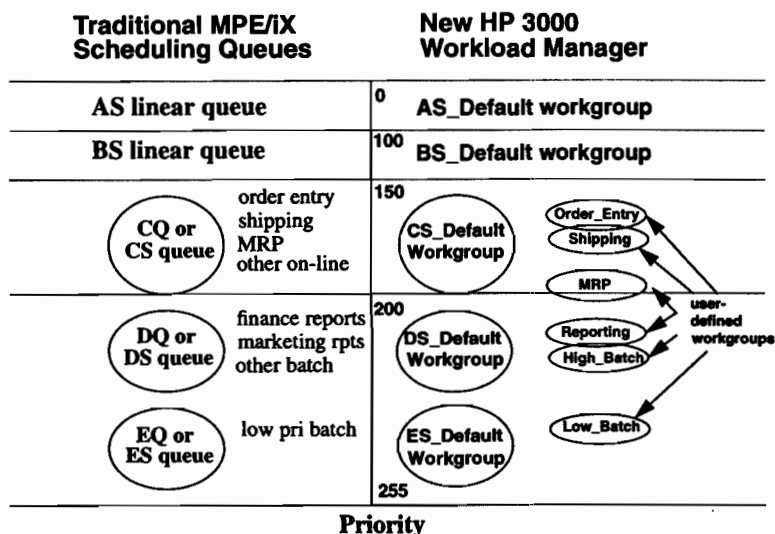
The current means of influencing the scheduling subqueue of a process remain the same, allowing the user to affect the queue attributes of processes they create. The queue can be specified via the ;PRI parameter on the HELLO, JOB, and ALTPROC commands. MAXPRI can be customized for particular users and accounts. The Link Editor allows a maximum and default scheduling queue to be set for program files. The routines GETPRIORITY and AIFPROCPUT can change the process queue attribute. Thus, the system manager cannot precisely control the queue attribute of a process.

However, while users can continue to influence scheduling subqueue assignments, the queue attribute of a process is just one factor that determines workgroup membership (others include the logon and the program file).

Workgroup membership is determined by the system manager and cannot be changed by the user. The queue attribute (which the user can influence) is one attribute that determines workgroup membership, but the system manager can also use the logon and program information to determine membership. Thus, the system manager has complete control over workgroup membership.

While the ability to create user-defined workgroups requires the Workload Manager, every system updated to Release 5.0 of MPE/iX will have five system-defined workgroups (AS_Default, BS_Default, CS_Default, DS_Default, and ES_Default). These workgroups each have a membership criterion specifying one of the five queue attributes, and their scheduling parameters match those of the five scheduling subqueues. Thus, complete backward compatibility is provided by the five default workgroups. All processes on the system have a queue attribute (AS, BS, CS, DS, ES) and therefore will belong to one of the five default workgroups. Figure 1 (on the following page) shows the five traditional MPE/iX scheduling subqueues, and their corresponding default workgroups.

FIGURE 1.



Effectively Manage Competing Workgroups

Needs

The system workload on an HP 3000 system is composed of many processes. These processes will most probably have different performance demands of the system. A telephone sales representative might require one-second response time in order to effectively process customer orders. Other users might simply require adequate response time to avoid noticeable delays in their editing sessions. A particular report might need to be completed by the next business day, while another report must be available within two hours. The processes supporting all of these different activities are running on the same HP 3000 system, competing for the same set of resources, including the competing for the system CPU(s).

The system manager understands the performance requirements of the various users of the system, and needs to be able to control the system in order to balance these competing performance needs.

Workload Manager Features

Once the system workload has been partitioned into user-defined workgroups, the system manager can control the scheduling behavior of processes within those workgroups. Each workgroup supports the full complement of scheduling parameters available with the traditional scheduling subqueues: base and limit priority, quantum bounds, boost property, and timeslice. In addition, new parameters of CPU percentage bounds have been introduced for the user-defined workgroups. The system manager can guarantee a minimum amount of CPU to a workgroup or restrict the workgroup to a maximum amount of CPU.

In addition to showing the five system-defined default workgroups, Figure 1 shows several user-defined workgroups (Order_Entry, Shipping, MRP, Reporting, High_Batch, and Low_Batch). Each workgroup might have different performance requirements. By customizing the scheduling parameters of the user-defined workgroups, the system manager controls the performance of processes within that workgroup.

For example, the Order_Entry and Shipping workgroups are given higher priority than the MRP workgroup. Processes with a queue attribute of CS that do not belong to one of the user-defined workgroups will be in the CS_Default workgroup. Through customizing the scheduling parameters of the workgroups, the system manager is able to control the CPU access of processes within those workgroups.

In addition to the traditional scheduling parameters, the Workload Manager introduces minimum and maximum CPU percentage bounds for the user-defined workgroups. Processes in a workgroup with a minimum CPU percentage of 20% will be guaranteed 20% of the CPU(s), provided they have enough demand to use the 20%. If the CPU demand of all processes within the workgroup is less than the minimum CPU percentage, the "extra" CPU time is made available to processes in other workgroups. If the processes within the workgroup need more than 20%, they can receive more, provided they do not violate the minimum values for other workgroups. Thus, the minimum is a true minimum and can be exceeded.

The maximum CPU percentage serves to restrict the CPU consumption of processes within a workgroup. Processes in a workgroup with a maximum CPU percentage of 50% will **never** receive more than 50% of the CPU(s). If no other workgroups require CPU, the system will idle rather than allow the workgroup to exceed its maximum. Thus, the maximum can be used as a throttle to limit the CPU consumption. This can be used to restrict the impact

of one workgroup on other workgroups. A workgroup containing CPU-bound database queries might be given a CPU maximum of 25%. A workgroup containing batch jobs might be held to only 20% during production hours to limit its impact on the response time of on-line users.

The combination of the traditional scheduling parameters and the new CPU percentage bounds allows the system manager to accommodate the different performance demands of the processes on their system. The Order_Entry workgroup might have priority bounds from 152-160 and a CPU minimum of 25% to ensure response time is appropriate. The workgroup containing certain critical reports might have higher priorities than another workgroup with less important reports. The workgroup containing CPU-bound processes with a history of impacting other processes might have a CPU maximum of 30%.

Thus, the Workload Manager allows the system manager to control levels of service by controlling the CPU access of the workgroups that have been defined. The precise controls used will depend on the goals of the system manager, but could result in maintaining a certain average response time, or providing a certain degree of throughput.

Effectively Manage Changing System Demands

Needs

The workload imposed on a system is rarely constant. The system manager needs to be able to respond to both unanticipated and anticipated changes. An unanticipated change might involve a user running a compile at high priority that is now impacting the response times of other users, or the need to expedite the processing of a particular job or database query so that the marketing department has the information they need for a 2 p.m. deadline. Thus, the system manager needs reactive controls that enable them to handle situations as they arise.

Anticipated changes reflect the system manager's knowledge of their environment. For example, the data entry clerks in the accounting department may all start work at 8:30 a.m., causing a rapid increase in system activity. Or perhaps after 6 p.m. most interactive users have left and the system is primarily used for batch processing. Based on an understanding of the changing demands placed on the system, the system manager needs proactive controls that enable the system to change scheduling policy in anticipation of changing system demands.

Workload Manager Features

The Workload Manager provides both the reactive and proactive controls motivated in the above discussion. Reactive controls are available via the ALTWG and ALTPROC commands, while proactive control is provided primarily through the NEWWG command.

Reactive Changes

While the ideal is to proactively manage the system, often problems arise unexpectedly and reactive changes are necessary. The Workload Manager can be used to make reactive changes such as increasing or decreasing the amount of CPU available for a particular user or set of users.

Increase CPU Access: Once the Workload Manager has been used to group processes into workgroups, the entire workgroup can be given increased access to the CPU. Perhaps it is a busy day for orders and the telephone sales reps need faster response time and therefore you want to increase their CPU access. Perhaps there was a problem with a batch run the night before and you want to give the batch jobs limited CPU access during the day so they can complete.

The ALTWG command can be used to adjust the scheduling characteristics of the workgroup to give it increased CPU access. The base and limits of the workgroup might be increased to higher priorities. Since the MPE/iX Dispatcher is priority-driven, this will give the members of this workgroup preference over lower-priority processes. If the workgroup has been given a minimum CPU percentage, and is using that percentage, the ALTWG command can be used to increase the percentage of this workgroup (and decrease the minimums of other workgroups). Note that this is **only** effective if the workgroup is using its minimum; if there is not enough CPU demand within the workgroup to consume 20% of the CPU, raising the minimum to 25% will not improve performance. Also note that if the workgroup is constrained by a maximum and is hitting that maximum, raising the maximum will serve to give the workgroup greater access to the CPU.

The above scenario assumed that the entire workgroup (all member processes) was to be given improved CPU access. If there is a specific process that requires increased access to the CPU (a particular user who has called with a problem, a batch job that must complete quickly, etc.), the ALTPROC command can be used to adjust that process. The ;WG option can be used to move the process to a workgroup with better CPU access (higher priority or a larger CPU minimum value). For example, the system manager might define a High_Priority workgroup that runs with base and limit set to

152. Processes requiring fast CPU access could be placed in this workgroup via `:ALTPROC pin;WG=High_Priority`.

Decrease CPU Access: The opposite case involves a need to decrease the priority of a set of users. Perhaps a batch run did not complete the night before and is now impacting the response time of interactive users. Perhaps a group of users is receiving 0.25-second response time when 0.5-second response time would be sufficient. Decreasing that group's CPU access can allow the "extra" CPU to be given to other processes, without resulting in noticeable change to the original group of users. The `ALTWG` command can be used to alter the scheduling characteristics of the appropriate workgroup. The base and limit priorities might be lowered (thereby reducing the priorities of all member processes). If the workgroup has been assigned a minimum amount of CPU and is using that minimum, the value might be lowered. As before, this will only have an effect if the workgroup is using its minimum. Lowering the CPU minimum from 25% to 20% will have no effect if the workgroup is only consuming 15%. Alternatively, a maximum CPU percentage might be imposed on the workgroup. A workgroup might be constrained to run within 20% of the system.

As before, it might be the case that a single process requires adjustment rather than an entire workgroup. There may be a process in an infinite loop that is critical and cannot be killed, or a CPU-intensive process which may be performing a non-time-critical task. The `ALTPROC` command can be used to move the process to a workgroup running at lower priority.

Proactive Changes

In addition to supporting the reactive changes that are often required, the Workload Manager allows the system manager to make proactive changes. The mere act of configuring user-defined workgroups is actually a form of proactive change. The system manager might know that their programming staff has a tendency to sneak in high-priority compiles, and therefore creates a `Compiles` workgroup that will capture processes running the compilers and route them to lower priorities where they don't disrupt the interactive users. Similarly, certain users or programs might belong to user-defined workgroups running with specific scheduling characteristics.

The Workload Manager provides additional controls that build on the concept of using user-defined workgroups to proactively control system performance. Often a system manager can predict the behavior of their system workload. They know the performance requirements at 9 a.m. differ from those at 6

p.m. and on the weekends. The NEWWG command supports an indirect file format that can be used to make proactive changes.

The system manager can create configuration files that represent the desired workgroup configurations for different system performance needs. The files need not be created from scratch; the SHOWWG command with the WGFILF format option can be routed to a file. This file can then be modified as appropriate. The NEWWG command supports a ;VALIDATE option that will merely check the validity of the indirect file, not invoke the changes. In this way, the system manager can be assured subsequent NEWWG commands using the file will not fail. The NEWWG command with the appropriate configuration file can be invoked via timed job streams or at the completion of certain activities.

Provide More Consistent Response Times

Needs

End-user complaints often center around response time. In some instances, the increased response time is indicative of a problem with the operation or the system (for example, a database deadlock, a looping application program, or a network problem), but frequently the problem is more one of perception. A particular job took 30 minutes last week, and now it has run for an hour and hasn't completed; the time to process a screen of data is usually 1-2 seconds, and now is taking 3-5 seconds.

In some cases, the need for consistent response time goes beyond mere user perception and satisfaction; some MIS organizations negotiate specific Service Level Agreements (SLAs) with the users they serve. The SLAs may specify a particular level of service that is to be provided (e.g., 90% of all transactions complete in less than 2 seconds).

Workload Manager Features

The Workload Manager features can be used to ensure more consistent response times for users. The need might be to meet a specific Service Level Agreement (SLA) with users, to minimize performance complaints, or to facilitate capacity planning. The definition of workgroups allows the system manager to partition processes into groups with similar needs. Customizing the scheduling characteristics of those workgroups provides the control over CPU access, which in turn helps determine response time.

It is critical to understand that CPU access is just one component of response time. The Workload Manager can help handle this aspect, but cannot handle

problems with disk access speeds, memory constraints, network latency and availability, or the other components of response time.

In controlling CPU access, the system manager can either control the workgroup needing the consistent response times, or identify the other workgroup(s) whose behavior leads to inconsistent response times and control those workgroups.

Single Workgroup: In handling a single workgroup, the system manager can manipulate priorities (changing the base and limit priorities), the rate of priority decay (changing the quantum), or the CPU percentages. Which control is most effective depends on the characteristics of the processes being controlled. It may be the case that placing the workgroup at priorities 160-170 gives consistent one-second response time to those users. Alternatively, it may require that a minimum CPU percentage of 20% be given to the workgroup in order to ensure one-second response time.

If the workgroup is constrained by a maximum CPU percentage and is using that percentage, raising the CPU maximum may allow the processes to receive more CPU (providing the minimum guarantees of other workgroups are not violated), and the response time may drop. If the response time of the users within the workgroup is too fast (e.g., half-second when one-second would be sufficient), imposing a CPU maximum might serve to even out the response times to one second.

Other Workgroup(s): If processes in Workgroup_1 are having inconsistent response times, it may be due to the influences of other workgroups. Perhaps Workgroup_2 is at higher priority and contains processes who perform long transactions, leading to increased response times for the processes in Workgroup_1. Those processes might be removed from Workgroup_2 and placed in a workgroup at lower priority. Workgroup_2 might be moved to a lower priority. Alternatively, a CPU maximum might be set for Workgroup_2, restricting the amount of CPU it can consume. If it is a single process that is disrupting the response times of others, it might be moved to a lower-priority workgroup.

System Consolidations

Needs

The process of consolidating multiple systems onto a single HP 3000 system brings forth several unique needs. One typical concern regarding consolidation is the limited amount of control available on the target system

for the consolidation. Five scheduling subqueues were available on *each* of the source systems, and *only* five scheduling subqueues are available on the target system. Thus, the target system will have the workloads from all source systems, but will have only the same control capabilities.

This "loss" of control relates to another concern, that of the impact of combining the various workloads. The workloads of the source systems had been physically separated prior to the consolidation, and will be placed on a single system. The need to partition the workload and manage the interactions and performance is clear.

The final need centers around user expectations. Consider a consolidation situation where systems A, B, and C are being consolidated onto system D. Often the consolidations are spread over time, perhaps bringing over System A on weekend 1, System B on weekend 2, and System C on the following weekend. One problem that can result from this deals with the performance expectations of the users of System A. While they are running alone on System D, the performance (response time and throughput) of the users from System A is excellent. When they are joined by the users from System B, their performance may degrade. Once all three systems are combined on System D, the System A users may actually complain about their performance.

Workload Manager Features

The Workload Manager can provide value when consolidating multiple source systems onto a single target HP 3000 system. The Workload Manager features can address concerns when planning for the consolidation, during the consolidation itself, and when managing the final consolidated system.

Partitioning: When planning the consolidation, the concern regarding combining the various workloads and still providing acceptable performance must be addressed. The Workload Manager can be used to define multiple workgroups that represent the users of the various systems. If desired, workgroups can be created to represent the CS, DS, and ES processes from each of the source systems. This would serve to preserve the partitioning that had been available with the physical separation of the source systems.

Alternatively, the Workload Manager can be used to define workgroups that more naturally reflect the needs of the combined user population. Perhaps data entry clerks had been in the CS subqueue of several source systems and can be combined into a single workgroup on the target system. Similar batch jobs might be collected into a common workgroup. Users from one system

who were forced to share the CS subqueue can now be broken into individual workgroups. The scheduling characteristics of the workgroups on the target system can be adjusted to result in the CPU access that the system manager requires to achieve desired performance.

User Expectations: As demonstrated in the earlier example, there may be a need to manage user performance expectations during the consolidation itself. How can the Workload Manager be used to help this situation? Consider the root cause of the dissatisfaction of the users from System A. They had grown accustomed to the better performance when alone on System D, and felt the hit when it settled to steady state after the consolidations were complete. The CPU maximum feature of the Workload Manager can be used to restrict the amount of CPU available to users. The users from System A could be constrained to use only 30% of System D. Thus, they will experience from the onset the performance that will result when all consolidations are complete. Their expectations will not be set artificially high, and thus they will not be disappointed by the results after the consolidations are complete.

This example is obviously simplified. The system manager may not wish to divide the target system up evenly among the users from the three source systems. Perhaps one set of users is more important and requires more of the CPU. Alternatively, the issue with the consolidation may be a concern of how to ensure that competing workloads from the various source systems are able to co-exist on the target system. The partitioning into workgroups addresses this concern.

The benefit in this case is to provide consistent performance throughout the consolidation; the Workload Manager can also add value when a system is in its steady state (as discussed in the previous sections).

Conclusion

The preceding discussions have outlined various workload management needs and shown how the Workload Manager features can be used to address those needs, ranging from improved granularity and control, to proactively managing changes in system demand, to consolidating systems. Specifically, the Workload Manager provides the system manager with the ability to partition their system workload into user-defined workgroups and establish scheduling parameters to govern the performance of processes within those workgroups. In addition to this improved granularity and control over workload partitioning, the Workload Manager provides new scheduling

parameters for the user-defined workgroups. Minimum CPU percentages can be used to ensure a workgroup receives a guaranteed percentage of the CPU, while maximum CPU percentages can be used to constrain the processes within a workgroup to only consume a certain percentage of the CPU. With these features, the Workload Manager gives system managers the control necessary to ensure their HP 3000 systems can effectively meet the performance demands of their environment.

Acknowledgments

We would like to extend special thanks to our customer partners and their field teams who helped ensure we had an accurate understanding of the challenges they were facing in tuning their systems for desired performance, such that we could design controls to meet their needs. Also, we appreciate all of those within and outside of H-P who have taken the time to provide us with insight into the problems they or their customers face with workload balancing. Obviously, particular thanks are due to those who invested so much of their time and energy in designing, developing, and delivering the Workload Manager (formerly known as High-End Scheduling): Pat Alvarez, Amy Arnold, Amy Blocher, Wing Chan, Daren Connor, Jim Curtis, Greg Gloss, Patrick Goddi, Raymonde Guindon, Joy Hansen, Todd Hirozawa, Mark Hoerth, Michael Kahn, Larry Matteucig, Pam Miller, Phuong Nguyen, Shelley Nelson, Bellos Nisan, Jyothy Reddy, Chuck Samson, Narinder Sandhu, Bob Togasaki, Jeff Vance, Meadow Walker, Robert Winter, and JoAnn Yuki. Their contributions have helped the Workload Manager meet the needs of HP 3000 customers.

Managing a 5.0 POSIX HP 3000 System

Jeff Vance, Hewlett-Packard

(6 July 1994)

Abstract

Release 5.0 marks a significant change for MPE-- POSIX system calls are now part of the operating system. But POSIX is more complex than additional C library intrinsics. To support the POSIX APIs, major enhancements were made to the file system, directory, process management, Store/Restore, C library and the Command Interpreter. The result is that for the first time UNIX and POSIX applications can run on MPE/iX with little or no changes. It also means that system managers need to know how to manage the POSIX extensions to MPE.

This paper will first define POSIX and describe how POSIX has been implemented in MPE/iX.

Next, some systems management issues related to POSIX are addressed. Topics covered include: system backup, disk space management, security, system logging, file transport, etc. A glossary contains the definition of the terms used throughout the paper.

This paper was first presented at INTEREX '93 in San Francisco and has been updated to reflect the newest 5.0 POSIX enhancements.

What Is POSIX?

POSIX (Portable Operating System Interface for uniX), a standard set of interfaces developed by IEEE designed to facilitate application portability, covers a broad range of operating system functionality, but currently only two focus areas have been ratified:

- 1003.1 -- "System Application Program Interface for the C language" (.1, spoken "dot one")
- 1003.2 -- "Shell and Utilities" (.2, pronounced "dot two")

POSIX.1 defines C programming interfaces for directory and file creation, file security, process management, interprocess communication (IPC), file sharing among users, file reading, file writing, and general terminal I/O. Essentially, all operating system support needed by a C application is covered by POSIX.1. MPE programmers can view POSIX .1 as a new set of intrinsics for C programs. MPE provides many of the same features (with different names), but POSIX standardizes the "intrinsic" name, behavior, and error returns. For example, the

POSIX *rename()* function changes the name of a file, whereas the MPE *FRENAME* intrinsic accomplishes the same task. However, certain behaviors of *rename()* differ from *FRENAME*. Note: IEEE is evaluating a similar standard for FORTRAN and COBOL.

POSIX.2 (often referred to as the ".2 shell") defines commands and utility programs that invoke the POSIX.1 APIs. This is analogous to the MPE Command Interpreter providing access to intrinsics. For example the POSIX.2 command to rename a file is *mv*, which calls the POSIX *rename()* function; whereas, the MPE equivalent is the *:RENAME* command, which invokes the *FRENAME* intrinsic. The .2 shell contains many popular UNIX commands such as *grep*, *awk*, *diff*, *ls*, *sed*, *rm*, etc.

It is important to note that POSIX is only one piece of the application portability puzzle. Database, networking, client/server, and user interface standards must be adopted to obtain true application portability.

Structure of POSIX and MPE

Release 5.0 of MPE/iX contains the POSIX.1 APIs and the .2 shell and utilities. The POSIX.1 APIs are implemented in NL.PUB.SYS, but a separate RL, available with the MPE/iX Developer's Kit, is required in order to link a C program. Note: the MPE/iX Developer's Kit is not necessary to run a POSIX application; it is needed to link a POSIX program.

The .2 shell and utilities reside in a new group named HPBIN.SYS. The shell uses files located in the hierarchical directory (e.g., /usr/terminfo/h/hp2648a). In Release 5.0, all POSIX directories under root and the HPBIN group can be purged without adversely affecting the rest of the operating system. Although this frees approximately 73 megabytes of disk space, purging these files is discouraged since third party solutions may rely on services provided by these files and directories. It is likely that future enhancements to the MPE/iX operating system will also depend on POSIX.

The POSIX.1 APIs and the .2 shell are part of the Fundamental Operating System (FOS) and are included in the System Load Tape (SLT).

OSIX /iX Primer

If you are already familiar with the POSIX implementation on MPE/iX then you may wish to skip to the section titled "POSIX/iX System Management" where specific system management issues are discussed.

OSIX has been designed to integrate well with MPE. It is implemented at the intrinsic layer and lower, so performance is par with MPE intrinsics. POSIX is not an isolated subsystem. A POSIX application views MPE as part of the POSIX world. The converse is normally not true: existing MPE applications are typically blind to the POSIX extensions. The next several pages summarize the enhancements in MPE to support POSIX.

- **Hierarchical File System and Directory (HFS)** - The POSIX.1 standard defines a hierarchical directory structure of unlimited depth originating from the root directory. This concept should be familiar to UNIX and MS-DOS users. POSIX applications see MPE groups and accounts as directories. Figure 1 shows a POSIX view of a typical MPE accounting structure. Note that MPE accounts and groups are a subset of the generic POSIX directory structure. Also the root directory, which has existed since Release 1.0, is visible for the first time. Figure 2 shows the HFS extensions to the same basic directory organization.

The directory that you are logged in to is called your current working directory (CWD). Your CWD is usually your logon group, which can be changed via the :CHGROUP command. However, the new :CHDIR command also allows you to position your CWD to any directory on the system that you have access to, independent of your logon group. In fact, with the appropriate access, you can move your CWD to a group in another account without re-logging on. The location of your CWD has no bearing on file access security--it simply provides a shorthand way to name files.

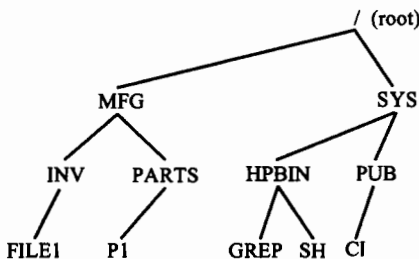


Figure 1. POSIX view of MPE directory.

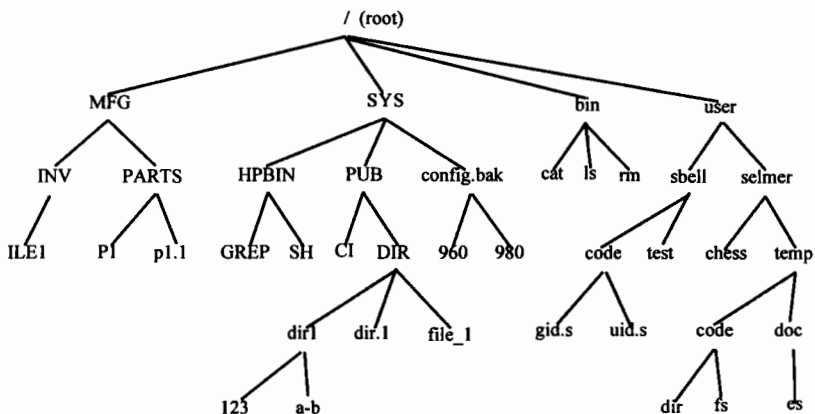


Figure 2. Mixed MPE and POSIX hierarchical directory.

- File Name Syntaxes** - A POSIX filename is usually referred to as a "pathname", or in MPE parlance, as an HFS name. A pathname may begin from your current working directory, which is called a relative pathname, or can start at root, known as an absolute pathname. MPE has the same concept: unqualified filenames (relative to your logon group) versus qualified names where the group and account are specified. POSIX pathnames are "top down" names, contrasted with MPE filenames which are "bottom up". For example, these two filenames designate the same file:

 CI.PUB.SYS = /SYS/PUB/CI

By default, all MPE intrinsics and commands expect the familiar MPE syntax. For example, `:BUILD X` creates file X (uppercase) in your CWD. (Remember your CWD is your logon group unless you have CHDIR'd somewhere else.) Also, `FOPEN("x")` opens X in your CWD. POSIX applications, including the .2 shell, assume POSIX syntax, which is case sensitive, supports long pathnames, permits special characters, etc. See Figure 3.

MPE commands and intrinsics can name POSIX files by using a new syntax called MPE- escaped syntax. If an MPE filename begins with a dot (".") or a slash ("/") then the name is assumed to be a POSIX name, and all POSIX syntax rules apply. For example, `:BUILD /x` creates file x (lowercase) in

your CWD, which may be your logon group, and FOPEN("./x") opens file x. The MPE escape syntax trigger (". " or "/") works because absolute pathnames begin at root (/) and relative pathnames begin at your CWD, which POSIX defines as the file dot (". "). So, the name "./x" is interpreted as "starting at your CWD (dot) locate the file below it named x". Figure 3 summarizes the differences between MPE syntax and POSIX syntax.

FEATURE	MPE	POSIX
max number of components	3	unlimited. MPE has a depth limit of 512 for wildcarded CI operations.
component delimiter	"." ("/" introduces lockword)	"/"
max component length	8	255 *
max overall file name length (non-networked name)	35	1023 *
legal characters	A - Z, 0 - 9, cannot start with a number.	A - Z, a - z, 0 - 9, "-", "_", ".", cannot start with "-".
lockword?	yes	no
back reference to file equation?	yes	no
temp file?	yes	yes
program file?	yes	yes
database file?	yes	no
stream job file?	yes	no
UDC file?	yes	no
command file?	yes	no

The CI has further length restrictions due to its command buffer size and limits of the MYCOMMAND intrinsic. The maximum size for a compatibility mode command is 253 for relative and 255 for absolute pathnames. The maximum size for native mode commands is 511 (in 5.0) or 279 (in 4.5) less the length of the command name and other parameters. Also if the POSIX object is directly under root, an account or an MPE group its name length is reduced to a maximum of 16 characters.

Figure 3. Comparison of MPE vs. POSIX name rules.

- Security - Access Control Definitions (ACDs)**, a major component of POSIX security on MPE/iX, were introduced to MPE XL in Release 3.0 and have been enhanced for POSIX. ACDs differ from "matrix" security in several ways. Matrix security allows file access to be controlled at the account and group level; ACDs define all security rules at the file level. Matrix security limits subjects to account users (AC), group users (GU), creator (CR), and others (ANY); whereas, ACD security is much more

expressive, permitting different accesses for individual users. In Release 5.0, ACDs are better integrated with the operating system. For instance, the ACD of a file gets copied, stored or renamed with the file. Also, in 5.0, EDITOR leaves the ACD intact when keeping a file.

A few POSIX concepts must be defined before discussing the new ACD features. Each user of a POSIX system is assigned a user ID (UID), which uniquely identifies this user to the system. POSIX defines this identification by a number, but on MPE/iX it is currently implemented as a "user.account" string. There is also a number associated with this string, which is seen by POSIX applications as the UID, and is kept in the new user database file (HPUID.PUB.SYS).

Each user is also assigned an ID which associates them with a group of users. This ID is called a group ID (GID) and all members of the same POSIX group share the same GID. A POSIX group is just a collection of users without regard to directory structure. In MPE, all users in an account form a "group" for the purpose of file sharing. Therefore, the current MPE/iX implementation of the POSIX group ID is the string name of the logon account. Of course, POSIX applications see the numeric form (GID) of this "group", which is stored in the new group database file (HPGID.PUB.SYS).

MPE keeps track of a file's creator, which is slightly different from the POSIX concept of a file's owner. The creator is static for the life of the file; whereas, the owner of a file can change over time (assuming there is a way to assign a new owner to an object). MPE/iX has adopted the owner model, meaning that the original creator ID will be lost once ownership is altered. POSIX determines the owner of a file via the file's owner ID, which is a number. MPE/iX identifies the owner of a file by the "user.account" string. If a file's owner ID matches the UID of a user then that user is considered to be the owner of the file. In MPE terminology, if the creator of the file is the same as your logon user and account then you own the file. The new :ALTFILE command changes file ownership. To use this command you must be the current owner, possess SM capability, or be the GID manager (see below) of the file. However, only the System Manager can alter file ownership to any user on the system.

MPE facilitates file sharing via the account, group and file security matrices. MPE also overrides matrix security for all released files and all files with an ACD. POSIX assigns a file group ID to a file, and all users whose GID matches the file's group ID can participate in file sharing. By default, when you logon to an account the account name is your GID and all files in that account are stamped with the same group ID, so you are considered a POSIX group user of the file. If you also have Account

Manager (AM) capability you are deemed the GID manager, meaning you can manage all files matching your GID. The :ALTFILE command can also be used to change a file's group ID.

Since POSIX allows the owner of a file to be altered during the life of the file, owner specific security is necessary. Also, POSIX lets a file's group ID (GID) change, and hence a group ID security rule is essential. To satisfy these requirements three new ACD subjects have been created:

- \$OWNER - signifies the file's current owner,
- \$GROUP - identifies the file's current group ID (GID),
- \$GROUP_MASK - used for POSIX compliance so that ACDs are a viable alternative security mechanism¹.

At the same time, the existence of HFS directories motivated the creation of four new access permissions:

CD - Create Directory entries - allows the insertion of names into the target directory entry. Lack of CD access implies the user cannot create files or directories under the target directory. The user must have Save File capability in order to obtain CD access². CD is analogous to POSIX Write access.

DD - Delete Directory entries - allows the deletion of names from the target directory entry. Lack of DD access implies that users cannot purge files or directories under the target directory³. DD is analogous to POSIX Write access.

The access defined by \$GROUP_MASK is bit ANDed with the access defined for the matching file group subject to determine the final access. This is necessary because a file's group class contains both MPE and POSIX elements. For instance, *user.account* and *@.account* are MPE subjects, whereas, \$GROUP is a POSIX subject. In order for the POSIX *chmod()* function to be compliant, \$GROUP_MASK is set when any of the MPE forms of a file's group class already exist in the ACD. For example, assume the ACD for FILE1 is:

```
R, X      : @.myacct
R, W, X   : you.myacct
R, L, X   : $group_mask
```

The three ACD pairs above contain subjects that make up a file's group class of accessors. In this example, the file owner and "other" classes have not been defined. "Other" refers to all potential accessors who are not the file's owner and do not belong to the file's group class. "Other" is the "@.@" ACD subject. Any user logged on to MYACCT is given read and execute access to FILE1-- the \$GROUP_MASK does not restrict access. However, if the user *you.myacct* tries to access FILE1 he will not get write access because \$GROUP_MASK denies write access. The result of the specific *you.myacct* access ANDed with \$GROUP_MASK is read and execute access only. So, \$GROUP_MASK acts as a filter that can further restrict access for the file's group class of users.

If the target directory is an MPE group, CD access is defined as the user possessing SF capability and having save access to the group. MPE groups do not support ACDs and thus cannot directly grant CD access.

If the target directory is an MPE group, DD access to the group is defined as having write access to a particular file within the group. MPE groups do not support ACDs and thus cannot directly grant DD access.

RD - Read Directory entries - allows the names in the target directory entry to be read. RD is necessary for all wildcard expansion and recursive operations rooted at the target directory. Lack of RD access implies the user cannot :PURGEDIR or :DISKUSE the target directory, and cannot :LISTFILE the contents of the directory (i.e., cannot see the objects under the target directory). This discussion of RD access applies to the last component of a pathname. A directory component in the middle of a pathname may deny the user RD access; however, as long as the user has TD access to that component, it is processed and the next component is evaluated. RD is analogous to POSIX Read access.

TD - Traverse Directory entries - allows the traversal of the target directory. TD access to each directory component in a pathname is necessary to "reach" or "traverse to the last component in the name. Lack of TD access implies the user cannot reach the next component in a pathname, i.e., cannot reach the target directory. TD is analogous to POSIX eXecute access.

Below are some examples using the new ACD features:

:ALTSEC myfile:repacd=(R,W,L,A,X:\$owner;R,L,X:\$group;X:@.@)
Changes the ACD for (or adds an ACD to) "MYFILE" such that the owner has all access (which is the default), members of the file's group have read, lock and execute access, and everyone else has only execute access. The *repacd* keyword is new for Release 4.5.

:ALTSEC ./mydir:repacd=(RACD,RD,CD,TD:\$owner)
Changes the ACD to "mydir" to prevent the owner from accidentally purging objects under it. Note the lack of DD access.

All files built outside of the MPE group and account structure, all directories, and all files created by a POSIX.1 API have an ACD automatically created. This ACD is required⁴ and allows everyone to read the ACD, but nobody has additional access. Therefore, if a user creates a directory under her logon group no one else can list or :CHDIR to her directory. No users will be able to create additional files or directories nor purge objects under her directory. However, as owner she can issue the :ALTSEC command and give away access to other users on the system. Remember, as creator, she can also :RELEASE her files.

MPE groups, accounts and root do not support ACDs but implicitly provide TD and RD access to all users. No other accesses are allowed, so only special users (e.g., SM or AM capability) are permitted to add new accounts, groups, or files under root. An ordinary user cannot build files or

⁴ In general, an ACD is required whenever a file's GID does not match the GID of the account the file is located under. If a file is not under an account an ACD is always necessary.

directories under root. Ordinary users are allowed to create directories under MPE groups as long as they have Save File (SF) capability, are a member of the account that the group belongs to⁵, have save access to that group, and have not exceeded their disk space limit. Save access to an MPE group grants implicit CD access to that group and DD access to the group for all non-file objects, such as directories and symbolic links. For file objects, e.g. flat file, MSG file, database, DD access to an MPE group is granted if the user has write permission to the target file⁶. In summary, to create a file directly under an MPE group still requires save access to that group; and to remove a file directly under an MPE group still requires write access to that file. And finally, save access to a group is sufficient to delete non-file objects. The above rules are necessary for MPE groups and accounts to conform to the generic POSIX security model controlled by write and execute directory permissions.

Users with SM capability and a file's owner are always granted all permissions; however, a file's owner can limit their access via an explicit \$OWNER ACD specification. No such artificial restriction is available to System Managers.

Once an ACD is attached to a file it remains with the file until it is explicitly deleted. ACD protection defeats lockword and matrix security. A file can be built outside of the MPE account structure with a lockword, but the lockword is ignored: CI and shell commands will not prompt for the lockword. If this same file is renamed into an MPE group the ACD remains but can be deleted via :ALTSEC ...;DELACD. The following rules apply to a file with a lockword and lacking an ACD:

- MPE interfaces will prompt for the lockword or it can be supplied inline with the filename.
- POSIX interfaces (including the .2 Shell and all HFS syntax) will not prompt for the lockword, and of course, a lockword cannot be specified directly in the pathname. Therefore a lockworded file cannot be accessed via any POSIX interface or using the CI's MPE-escaped syntax.
- The file can be renamed outside of the MPE account domain and an ACD will be attached and the lockword will be meaningless. This file can later be renamed back to its original name (location) and the ACD remains, but can be deleted. Note that the lockword is not automatically saved across renames.

Technically, the user's GID must match the GID of the MPE group's account.

The distinction between directories and symbolic links versus files for granting DD access to an MPE group is for backwards compatibility. We would prefer a rule that states that save access to a group grants both CD and DD access to the group for all objects contained within the group. However, in all MPE releases, files within a group cannot be deleted without the user having write access to the target file. So to ensure that the write access rule is not broken, the distinction is made between "old" file types (including stream files) and the new POSIX files such as directories and links.

- **Enhanced and New CI Commands** - Several new commands have been added to MPE/iX to support the POSIX features, and many commands have been modified to exploit the MPE-escaped syntax mentioned earlier. All new commands support MPE and MPE-escaped syntax.

New CI commands:

ALTFILE- alters a file's owner (UID) and/or POSIX group ID (GID).

CHDIR - changes your current working directory.

DISKUSE - displays disk space usage in sectors.

NEWDIR - creates an empty directory.

NEWLINK - creates a symbolic link.

PURGEDIR - purges one or more directories and optionally all files underneath.

PURGELINK - purges a file (ordinary, KSAMXL, symbolic link, device link, empty directory)

SETCLOCK - not related to POSIX but much needed - changes the system date and time with the system running and preserves time stamp sequence in log files.

SHOWCLOCK - displays the current date, time, time zone, and remaining time correction.

Old CI commands that accept MPE-escaped syntax:

ALTSEC, BUILD, COPY, FILE (right hand side only), LISTFILE, PRINT, RENAME, PURGE, RELEASE, RESTORE, RUN, SAVE, SECURE, STORE.

Other modifications to CI commands:

ALTACCT / ALTGROUP / ALTUSER - *cap*= + or - capabilities, e.g. ;cap=+op.

ALTSEC - allows \$OWNER, \$GROUP in an ACD pair; works with directories; allows ACDs to be replaced via "repacd=" keyword.

ALTUSER - can change the UID.

LISTACCT - *format=detail* displays UID and GID, *format=brief* shows just the account names.

LISTF - supports formats 5 and 7; lists MPE named directories and symbolic links.

LISTFILE - lists accounts, groups, directories, symbolic links, in addition to files; new output for all HFS objects.

LISTFTEMP - supports formats -3, -2, 4, 5, 6, 7.

LISTGROUP - *format=brief* shows only the group.account names.

LISTUSER - *format=detail* displays UID, *format=brief* shows only the user.account names.

NEWACCT - can specify a UID and GID.
NEWUSER - can specify a UID.
PURGE - supports wildcarding with three levels of confirmation.
STORE - supports storing all objects by internally translating "@.@.@" to "/".

Miscellaneous CI enhancements:

CI Variables:

There are three new pre-defined, read-only CI variables: HPCWD, HPFILE and HPPIN. HPCWD contains your current working directory name in POSIX syntax. Since your CWD is not shown with the :SHOWME command, you may want to display it in your prompt-- :SETVAR hpprompt "!!hpcwd:". Note, like all CI string variables, HPCWD is limited to 255 characters. If your current working directory pathname exceeds 255 characters CIERR 8159 is returned. HPFILE holds the fully qualified MPE filename of the currently executing command file or UDC file. It is null (i.e., len(hpfile) = 0) if you are not executing a user command. HPPIN is an integer variable that contains the current process's Process Identification Number (PIN).

Finfo Enhancements:

The FINFO evaluator function recognizes HFS names and supports the new FLABELINFO intrinsic items related to POSIX. Examples include: time of last file access, file owner (full user.account name and numeric ID), file type (directory, symbolic link, pipe, FIFO, device link, etc.), record type (fixed, root, spool, byte stream, directory, etc.), file size in bytes, KSAM XL version, device type (disk, tape, port, streams, sockets, etc.), and whether or not the file has been released. Some of this information is only available in Release 5.0.

CIOR:

Command I/O redirection supports HFS names. For example, :ECHO Hi there! >> /a-file, appends to the file named /a-file. Release 5.0 is needed for the above example to work. In Release 4.5 a file equation is necessary to create /a-file in the permanent domain, since the NEW and TEMP domains do not support HFS names in 4.5

- **Symbolic Links** - A symbolic link is a POSIX concept similar to the notion of a permanent file equation, such as :FILE A=B. Essentially, a symbolic link is a permanent file which contains the name of the target file or directory, which can, itself, be another symbolic link. Like the right hand side of a file equation, the target object may or may not exist. The symbolic link filename and the target filename can be either MPE or MPE-escaped syntax. If a symbolic link is found in a pathname, the target of the link is

inserted into the pathname in place of the symbolic link name, and the resulting pathname is evaluated. The CI establishes symbolic links via the :NEWLINK command. The :PURGELINK command deletes symbolic links (as well as any non-database file). :LISTFILE formats 5 and 7 show the target of a symbolic link. Symbolic links are available in Release 5.0 and later.

If a symbolic link is encountered anywhere in a pathname, other than the last component, it is always replaced by its target name. When the last component of a pathname is a symbolic link, it is replaced for all "data" operations (e.g., file open, copy, print, etc.). However, the actual filename of the symbolic link is used for "directory" functions (e.g., POSIX *rename()*, *unlink()*). STORE doesn't quite fit either model; however, if a symbolic link file is selected, it will be stored but its target will not be stored.

Here's how a few CI commands operate when the last component of the filename is a symbolic link:

```
:NEWLINK s1, f1    { creates symlink "S1" in CWD, points to "F1" }
:NEWLINK s2, f2    { creates symlink "S2" in CWD, points to "F2" }
:BUILD s1          { creates the file F1 in CWD }
:ECHO Hi there! >s1 { writes "Hi there!" to F1 }
:PRINT s1         { prints contents of F1 }
:COPY s1,f2       { copies F1 to F2 }
:RENAME s1,f3     { renames F1 to F3 }
:PURGE s2         { deletes F2 }
:STORE s2        { stores S2 only}
:NEWDIR s2       { creates directory F2 }
:ALTSEC s2; repacd=(RD,TD:$group, @.test; TD:@.@)
                  { replaces the ACD on directory F2 }
:PURGEDIR s2     { deletes directory F2 }
:PURGELINK s1    { deletes the symlink S1 }
```

- **System Logging Changes** - Release 5.0 has seven new system logging events (127, 128, 129, 142, 145, 148, 155) and new formats for all old events that log one or more filenames. The existing 1xx series formats (e.g. 105, file close) contain fully qualified MPE filenames. The new 2xx series formats contain HFS pathnames at the end of each record. The 2xx formats are variable record width so that a short filename consumes less disk space than required for a full 1023 character pathname. If an existing 4.0 event causes a filename to be logged, and the name is a valid MPE name residing directly under a group, then the traditional 1xx format is used. If a logged filename is outside of the MPE name space, or a new Release 5.0 event is logged then the new 2xx formats are used. LOGTOOL supports the both formats. The "*System Logging Changes*" Communicator article should be

read for more details.

It is convenient for log file analysis programs to have absolute pathnames (MPE or HFS) logged for all formats. As mentioned above, all 1xx formats contain fully qualified MPE filenames; however some 2xx formats log the HFS pathname exactly as it was specified by the user or application. For example, if file open logging is enabled and a user enters `:print ./my-file` then only *my-file* is logged, not *CWD/my-file*. In this case, if the absolute pathname of *my-file* is desired then it needs to be reconstructed by looking for logon, chgroup and chdir information in prior log records. The table below summarizes the new logging events:

LOG EVENT (#)	FORMAT NUMBER	FILENAME LOGGED
File close (105)	105 for MPE names 205 for HFS names	fully qualified MPE name absolute pathname ¹
File dir (127)	227 for MPE and HFS names	as user specified ²
Process adoption (128)	228 for MPE and HFS process names	absolute pathname ³
File owner change shown (129)	229 for MPE and HFS names	as user specified ⁴
Password change (134)	134 for MPE process names 234 for HFS process names	fully qualified MPE name absolute pathname
File store (136)	136 for MPE names 236 for HFS names	fully qualified MPE name absolute pathname ⁵
File CD change (138)	138 if all names are MPE 238 if any of the 3 names are HFS	fully qualified MPE name as user specified ⁶
User logging (140)	140 for MPE process names 240 for HFS process names	fully qualified MPE name absolute pathname
Process creation (141)	141 for MPE process names 241 for HFS process names	fully qualified MPE name absolute pathname
Security configuration change (142)	242 no filenames logged	not applicable
File open (144)	144 for MPE names 244 for HFS names	fully qualified MPE name as user specified
File command logging (145)	245 for MPE and HFS process names	absolute pathname ⁷
File name directory logging (155)	155, no log record generated	not applicable

¹ if the filename exceeds 1023 bytes the overflow bit is set and no name is logged.

² MPE directory names are converted to HFS names without qualification.

³ MPE process names are converted to HFS names.

4	MPE filenames are converted to HFS names without qualification
5	if the filename exceeds 1023 bytes the overflow bit is set and the leftmost 1023 characters are
logged.	
6	if any of the 3 names logged are MPE names they remain in MPE syntax.
7	MPE process names are converted to HFS names, except CI.PUB.SYS is logged as "CI".

- **Byte Stream Files** - POSIX defines a byte stream file as simply one or more bytes of data in a file. Unlike MPE fixed or variable records files there is no formal record structure, instead, the logical end-of-record is the newline character (ASCII code 10, linefeed). Many POSIX applications read a byte at a time until the newline is reached and then process the "record".

To MPE, byte stream files have a record width of one byte and a blocking factor of one. They can be created using the :BUILD command, e.g., :BUILD bytef, rec=,b: disc=100. This command creates a byte stream file that can contain up to 100 bytes of data. If the record width, blocking factor, or record type (ASCII vs. binary) are specified they will all be ignored. :LISTFILE .1 shows a "B" under the "TYP" column:

FILENAME	CODE	-----LOGICAL RECORD-----			
		SIZE	TYP	EOF	LIMIT
BYTEF		1B	BA	0	100

Most MPE applications would not be able to handle byte stream files because the application calls FREAD to read, say, 80 bytes, but FREAD would only return one byte of data. To solve this problem, whenever a byte stream file is FOPENed, the file system binds to a type manager responsible for making the byte stream file appear as if it is a variable record file. Without this type of emulation, the :PRINT command would display a byte stream file as a single column one character wide, with most of the contents missing. With byte stream emulation the majority of MPE applications work correctly on byte stream data. Of course, POSIX applications see byte stream files without any emulation because the POSIX *open()* function calls the HPFOPEN intrinsic so that the raw byte stream type manager is invoked.

Users of the shell, and users of tools that work in mixed MPE and POSIX environments, sometimes need to operate on MPE record oriented files. However, the POSIX.1 functions for file reading and writing don't perform with record structured files. Release 5.0 provides another type of byte stream emulation, where a classic MPE file is viewed by POSIX functions as a byte stream file. This emulation occurs whenever the POSIX *open()* function is called for a record structured file.

POSIX/iX System Management

The evolution of POSIX on MPE/iX places new demands on System Managers. It has become more complex and consequently more difficult to manage. Today, System Managers have a full workload without extra time to learn the new POSIX concepts. I hope this paper will help accelerate the learning curve. Also, there is a self-paced tutorial on POSIX named POSIXCBT.LSN.SYS which covers the topics addressed in the previous chapter. The prior chapter discussed general features of POSIX/iX, while this section covers some of the System Management topics related to running a production system on MPE/iX Release

It should first be mentioned that the topics below mostly apply to sites that will be running both MPE and POSIX applications. If your shop has no immediate interest in POSIX you have several options:

1. You can update to Release 5.0 and take advantage of the new `:SETCLOCK` command or wildcarded `:PURGE`, but not use the POSIX extensions. If most of your users are trapped into applications at logon, they will not notice any difference. More importantly, they cannot create any HFS objects, which means the System Manager doesn't really need to be POSIX knowledgeable. You can take comfort in knowing that HFS named files are not created automatically by the operating system (other than those on the FOS tape), and users need to take deliberate action to build a POSIX named object. Remember that `:BUILD` still creates the file uppercase "A" in your logon group (as long as you have not changed your CWD, which would not likely be accidental). Production jobs and programs will behave the same. You can restrict use of the .2 shell by adding a lockword; however, a better approach to limit access to the shell and utilities is to add ACDs to all program files in the HPBIN.SYS group.
2. You can stay on Release 4.0.
3. You can update to 5.0 and purge all of the FOS HFS files and the HPBIN.SYS group. This will save you some disk space, but will not prevent a user from creating an HFS object.

System Backup - The most popular concern shared by System Managers is "what if someone accidentally creates an HFS file or directory?". They are usually afraid that it won't get backed up by their standard backup procedures. On Release 4.5, the STORE program reported a warning and required an operator reply if it determined that a system backup was requested but MPE syntax was specified. The easiest case to understand is

`:STORE @.@.@`. Clearly a full backup is desired, however the "@.@.@" signifies (to 4.5) only MPE legal filenames. This excluded all HFS objects on the system, including the files and directories supplied by HP. In this situation, STORE tried to warn the operator that not all files on the system

would be backed up. The 4.5 System Manager was encouraged to change all of her backup jobs to use `:STORE/` rather than "`@.@.@`", where "/" means root and all objects below it. For customers that really intended to only backup MPE named files, the 4.5 version of STORE also contained a new option called NOHFSWARN to suppress the warning and operator reply.

In Release 5.0 STORE was changed to recognize that "`@.@.@`" means *all* files, including directories, lowercase named files, symbolic links, etc. Production store jobs do not need to be modified to account for POSIX filenames or directories. Of course, HFS syntax is fully supported by STORE too. The table below shows the name translations done by STORE in Release 5.0.

<code>@.@.@</code>	---->	/
<code>@.@.account</code>	---->	/ACCOUNT/
<code>@.@</code>	---->	/LOGON_ACCT/
<code>@.group</code>	---->	/LOGON_ACCT/GROUP/
<code>@.c@v</code>	---->	/LOGON_ACCT/C@V/
<code>@</code>	---->	./@/
<code>j@</code>	---->	J@ (not translated)

The translation rule is simple: if the first component of an MPE name is "`@`" then the full MPE name is converted to an HFS name, with logon account and group filled in if necessary. The trailing slash ("/") at the end of the HFS pathname is significant-- it indicates a recursive operation. In the case of STORE, a trailing slash means to store the target object and *all* objects below it. This behavior is consistent with the `:DISKUSE`, `:LISTFILE` and `:PURGEDIR` commands. The 4.5 NOHFSWARN option is recognized but ignored in Release 5.0. This option should no longer be used and will be phased out.

What if you want to backup all of your MPE named files? "`@.@.@`" will be translated to "/", and "`./@/@/@`" will match directories and lowercase files-- so neither works. The trick is to cause the first component of the MPE filename to not equal "@", but still match all files. This is accomplished by: `:STORE ?@.@.@.`

STORE tapes created on 4.5 and 5.0 can be RESTORED to earlier systems. In Release 4.5 the STORE tape format changed such that as soon as the first HFS named object is encountered, a file named HFSMAP is written to tape. This file contains the mapping of all hierarchical names to the simple, three tier MPE name space. For example, the HFS file `"/users/selmer/bin/temp.a"` might map to `"F0000001._HFSGRP._HFSACCT"`. There is one entry in the HFSMAP file for each HFS object selected by STORE. If a 5.0 store

tape is restored to a 4.0 system via `:RESTORE :@.@.@`, the F0000001 through F00n files are not restored because their group ("`_HFSGRP`") and account ("`_HFSACCT`") names are illegal. However, the HFSMAP file can be restored locally and the desired HFS files can be restored by wildcarding the group and account names. For example, suppose F0003729 is the MPE name for the desired HFS file, `:RESTORE :f0003729.@hfsgrp.@hfsacct` will restore the file. Similarly, all HFS files can be restored to earlier systems via `:RESTORE :f#####.@hfsgrp.@hfsacct:local`. The HFSMAP file can be used to see the actual HFS filename that was restored.

When HFS named files are restored to pre-POSIX systems the data is intact, with one exception. Most likely the HFS named files were protected by ACDs, and STORE has preserved the ACDs on tape. However, systems prior to 4.5 do not understand these ACDs since the internal ACD version number has rolled in 4.5. These files are restored with their 4.5 ACDs, but these ACDs are not recognized on earlier systems, and therefore are considered corrupt. `:ALTSEC :DELACD` can be used to remove the ACDs.

The above discussion pertains to the unexpected moving of HFS files to pre-POSIX systems. If you are planning on transporting POSIX named files to Release 4.0 or earlier then you should use the new `TRANSPORT=MPEXL` parameter. When MPEXL transport method is specified, STORE converts the ACD (which may contain \$OWNER and \$GROUP subjects) to a pre-POSIX ACD version. The HFS name mapping described above is still relevant, but the ACDs will be functional after the file is restored.

- **Disk Space Management** - As with most new versions of the MPE operating system, the POSIX release requires additional disk space. The new HPBIN.SYS group, which contains the .2 shell and utilities, uses approximately 105,000 sectors (27 Mb) of disk space. The hierarchical directories and files supplied with FOS consume an additional 179,000 sectors (46 Mb) of space. There is an effort being made to reduce the disk space requirements for future releases, but this is the current situation⁷. If all files in the HPBIN.SYS group and all objects in the hierarchical directory are purged, approximately 73 Mb of disk space can be reclaimed. HP does not recommend this action in the long run because business applications may rely on the services provided by these files; however, currently the operating system has no dependencies on any of these files. Release 5.0 requires approximately 500 Mb of disk space on LDEV 1. If you are using HP 7933 or 7935 disk drives for LDEV 1 they will need to be replaced with newer drives. The SCSI disks, HP-FL C22x drives, HP C2202(3)A drives and HP 7937s are all supported. Some HP 3000 Series 950s, 955s, 960s and 980s manufactured prior to June 1991, may not

⁷These measurements were taken after installing the Alpha version of 5.0 (FOS SLT only).

support SCSI disk drives without an Input/Output Dependent Code (IODC) firmware upgrade.

To see all of the new HFS objects use the :LISTFILE command:

```
:LISTFILE /[a-z]@/,6
```

This displays all lowercase file and directory names under root, and for directories, it will list all files and subdirectories. The trailing "/" induces this recursive behavior and is equivalent to specifying the TREE command line option. This :LISTFILE fileset shows all FOS HFS files because there are currently no FOS HFS files with uppercase names, and account names don't match the lowercase pattern. (Remember MPE-escaped syntax is case sensitive.) An alternate method, which does not rely on upper versus lower case names, is:

```
:LISTFILE /,6; seleq=[object=hfsdir]
```

However, this command only lists HFS directory names-- the files in the directories do not match the selection equation.

The :REPORT command is a familiar way to determine how much disk space has been consumed by each MPE group. The current :REPORT command could not be enhanced to support POSIX directories, and since these directories have no user connection (e.g., no CPU limits), a new command was designed to report disk space used for all directories, including MPE accounts and groups. The new :DISKUSE command accepts a directory name in either MPE or MPE-escaped syntax, and reports the number of sectors of disk space used by that directory. The name can be wildcarded, and the output is always in HFS format. For example:

```
:NEWACCT a,mgr
:NEWGROUP g1.a
:NEWGROUP g2.a
:COPY editor.pub.sys, .g1.a
:COPY catalog.pub.sys, .g2.a
:DISKUSE /A
SECTORS
TREE      LEVEL   DIRECTORY
          BELOW
4720      96     /A/
```

This shows the A account using 4,720 sectors of disk space, of which, 96 sectors are consumed by the three groups under A. :REPORT @.A shows 4,592 sectors. The difference of 128 sectors is the space used by the directory nodes themselves-- the 96 sectors used by the three groups and 32 sectors consumed by the account entry.

```
:DISKUSE /A ;tree
```

SECTORS		
TREE	LEVEL	DIRECTORY
	BELOW	
336 +	304	/A/G1/
4320 +	4288	/A/G2/
32 +	0	/A/PUB/
4720	96	/A/ (32 +)

The TREE option (which is equivalent to supplying a trailing "/" in the directory name) displays information for all directories under the A account. The first line shows that 304 sectors are used for all files and directories under G1 (namely EDITOR). A `:LISTFILE /A/G1,2` reveals that G1 uses 32 sectors itself, and so the total amount of space used by G1 and all objects under it is 336 sectors. The second line of output can be similarly analyzed. Line three shows that the PUB group has no objects under it, but still contributes 32 sectors to the total. The final line reports the same totals as the previous `:DISKUSE` command, and shows that the directory entry A consumes 32 sectors itself. The "+" indicates a subtotal for each directory one level below the target name. Also all "+" values add up to the total under the TREE column.

MPE System Managers are accustomed to setting limits on the amount of disk space that a particular account can consume. Each account can also partition disk space among its groups. POSIX does not define any disk space limits, and thus this concept is not part of the POSIX hierarchical directory specifications. For technical reasons, MPE/iX has also not implemented directory based disk space limits. There are significant performance implications if limits are not carefully designed. Meanwhile, what is the impact of having directories without limits?

First, all directories created under MPE groups are bounded by the group limit, so this space is easily controlled. Second, only users with SM capability can create directories directly under root. So the exposure is reduced but not eliminated for two reasons: 1) the System Manager can alter the ACD on a directory under root, giving CD access to one or more users, and there is no way to restrict the space used by these users. 2) POSIX, by convention, defines a directory named /tmp which needs to grant all accesses to all users. /tmp is supposed to be managed as a temporary work space.

The first problem is solved by taking care when altering the ACDs on files and directories. Perhaps products like SECURITY/3000 from VESoft will become ACD aware and report all directories that give away CD access. Command files can be written to parse the output of a `:LISTFILE -2` command looking for access violations. The second problem is probably easier to solve. Simply stream a job each night that purges all files under

/tmp. For example, the commands `:CHDIR /tmp` followed by `:PURGEDIR /tmp/ :noconfirm` deletes all objects under, but not including, /tmp. A job can also be streamed at system start up that shows the amount of disk space in /tmp, e.g., use the `:DISKUSE /tmp/` command. It is also possible to alter the ACD on /tmp to restrict access to everyone except the users of POSIX applications that rely on /tmp.

If the above ideas are not suitable for limiting disk space consumption, and your site will be running POSIX applications, symbolic links may be an attractive work around. The essence of this approach is to trick POSIX applications into seeing all files as residing in the hierarchical file system, but, in fact, these files actually live in MPE groups with disk space limits. The first step is to backup your HFS files, e.g., `:STORE /[a-z]@/`. Next, purge these files via the `:PURGEDIR /[a-z]@` and `:PURGE /@` commands, which remove all HFS objects. Now, create an account and some groups to partition disk space for your POSIX applications. Assume the account is named PXACCT and the groups are APP1 and APP2. For each directory directly under root create a symbolic link of the same name that points to one of the APPx groups. For example, `:NEWLINK /tmp./PXACCT/APP1` and `:NEWLINK /usr./PXACCT/APP2`, etc. Symbolic links are discussed in the "*POSIX/iX Primer*" chapter. Now, restore all files and directories one level below root, e.g., `:RESTORE : /@/@/:create`. These files will actually be restore to one of the MPE groups defined above. One problem is that a `:LISTFILE /[a-z]@ :tree` only shows the symbolic link files; whereas, previously it listed all HFS files. Also, if HFSDIR objects are selected via the SELEQ= parameter the symbolic links under root will not match. However, `:LISTFILE /[a-z]@/@/` does display all the HFS objects with their original names.

User volumes are recommended as a safe and speedy way to perform backup and recovery operations. Unfortunately user volume sets are not yet fully supported in POSIX/iX. The good news is that all files and directories created at any level under an MPE group which is linked (HOMEVS'd) to a user volume are built on the user volume set, not the system volume set. However, all objects created directly under an account or under root reside on the MPEXL_SYSTEM_VOLUME_SET and cannot be directed to a user volume set. POSIX.1 mentions the concept of a mountable file system as a mountable subset of a complete file hierarchy, but leaves its full definition implementation specific. MPE has deferred the design of mountable file systems, but we envision it as a super set of today's user volume sets.

In the meantime, symbolic links can be used to force all files under an arbitrary directory node to reside on a user volume set in the same manner as symbolic links were used to apply limits to directories in the hierarchical name space. The idea is to create MPE groups that are HOMEVS'd to user

volume sets. Then, rather than building a directory named */usr/app1*, create a symbolic link of the same name which "points" to a group linked to a user volume set. For example:

```
:.NEWGROUP g1.a1:onvs=user_vs1  
:.NEWGROUP g1.a1:homevs=user_vs1  
:.NEWLINK /usr/app1, /A1/G1
```

Now when a POSIX application creates a file named */usr/app1/xyz* it will actually be created on the USER_VS1 volume set as */A1/G1/xyz*. One caveat is that the filename component cannot exceed 16 characters since the name is actually directly under a group (see Figure 3).

- **Security** - The two biggest concerns for System Managers regarding POSIX security are: 1) lack of a comfortable understanding of ACDs, and 2) the inability of ACDs to provide mandatory control vis-à-vis matrix security. The MPE/iX ACD enhancements are discussed in the "POSIX/iX Primer" chapter.

The second problem already exists since a released file is exempt from matrix security policies. However, security conscious sites have worked around this "feature" by disabling the :RELEASE command (including programmatic calls via the COMMAND intrinsic), and monitoring the system for released files. Similar solutions may be needed for files and directories protected by ACDs.

The first problem is equally challenging to solve. Even though ACDs have been available since Release 3.0, few customers use them. Part of the reluctance is the sense that ACDs are separate from the core operating system and not well integrated with subsystems. Release 5.0 marks the beginning of better ACD integration-- :COPY, :RENAME, :STORE and EDIT/3000 are ACD aware. Another reason that ACDs are not popular is that most 3rd party software does not support ACDs. Since ACDs are seldom used, most System Managers have little knowledge of or experience with ACDs, and this represents a hurdle for System Managers who need to run a mixed MPE and POSIX shop. The following HP manuals contain information on ACDs: "*Security Monitor/iX User's Guide*", part number 32650-90454, and "*Security Monitor/iX Manager's Guide*", part number 32650-90455. These two manuals are new for Release 5.0. The 5.0 manual "*New Features of MPE/iX: Using the Hierarchical File System*", part number 32650-90351 also describes ACDs and the POSIX security extensions. Additionally, the 5.0 *Communicator* contains a detailed article titled "*MPE/iX File Access and Security Enhancements*".

The rules for purging files have changed because MPE now supports directories and the POSIX *unlink()* function. :PURGE is the most secure

way to delete a file. If that fails you can try `:PURGELINK` or the Shell's `rm` command. If the file still cannot be purged then the directory contents (minus the file in question) can be copied or stored, followed by a `:PURGEDIR`. If your CWD is the target directory then it will not be purged, but all other objects under it will be deleted. This is analogous to changing your group to `G1` then issuing a `:PURGEGROUP G1` command - all files in the group are purged but the group remains intact. More details about `:PURGE`, `:PURGEGROUP`, `:PURGEDIR`, and `:PURGELINK` are provided in the following paragraphs, including cases where files cannot be purged.

The MPE `:PURGE` command *FOPENs* the target file (exclusive, append or write access) and then removes it via *FCLOSE(,4)*. All security rules apply such that if the file is already opened, is being stored, is allocated, the user lacks write access, etc., it will not be purged. If the file has a lockword then the lockword must be supplied with the filename or it will be prompted for. Because `:PURGE` calls *FOPEN*, individual file access rules must be obeyed before the file can be deleted, regardless of the access permissions to the file's parent directory. Of course, the System Manager can `:PURGE` any non-protected, non-privileged file on the system; however an Account Manager may no longer be able to `:PURGE` all files in their account. For example: if the GID of `FILE1` does not match an AM's GID then they cannot `:PURGE FILE1`. A `:LISTFILE FILE1,4` will reveal that the AM does not have write access to `FILE1`. `:PURGE` is the most secure way to delete a file-- it will not delete a directory or a privileged file. In Release 5.0, `:PURGE` has been enhanced to accept wildcarded filenames.

Traditionally, `:PURGEGROUP` allows an Account Manager to remove all files under a group, including privileged files and files with lockwords. As mentioned above, it is now possible in Release 5.0 to have files under an MPE group that cannot be purged by the Account Manager, and therefore, the group in which these files reside cannot be deleted. An MPE group can be purged as long as the following conditions are met:

- 1) the AM user has DD access to all subdirectories under the group, or
- 2) the AM user's GID matches the GID of all subdirectories under the group (even without explicit DD access to a subdirectory), or
- 3) the AM user is the owner of all subdirectories under the group, and
- 4) the AM user's GID matches the GID of all files immediately under the group, and matches the GID of all privileged files at all levels under the group.

These rules read as "(1 or 2 or 3) and 4". Of course, if there are no directories under the group then just rule 4 applies.

For example, suppose `GROUP1` contains files and directories under it, and those directories contain subdirectories and files, etc. Assume `AMUSER` is

the Account Manager for GROUP1. If the GID for every directory and file under GROUP1 matches *AMUSER's* GID then *AMUSER* can successfully **:PURGEGROUP GROUP1**. (This is identical to Release 4.0 security.) The fact that *AMUSER's* GID matches each subdirectory's GID grants *AMUSER* DD access regardless of the ACD on a particular directory. If, however, one of the subdirectory's GID does not match *AMUSER's* GID, and the ACD for that subdirectory does not grant *AMUSER* DD access, then the objects under that subdirectory cannot be deleted, and thus the **PURGEGROUP** operation fails.

Another example is when the System Manager alters the GID of a file to be different from the Account Manager's GID, in which case, the AM would not be able to purge the group that the file resides in:

```
:hello manager.sys          (user's GID="SYS")
:build file1.grp.acct       (FILE1's GID="ACCT")
:altfile file1.grp.acct; groupid=sys  (FILE1's GID changes from "ACCT"
to "SYS",                      also FILE1 is
automatically assigned an ACD)
:hello amuser.acct,pub      (user's GID="ACCT")
:purgegroup grp             (fails since FILE1 cannot be deleted)
```

So, by default, **:PURGEGROUP** works the same as on 4.0, except that POSIX security rules allow an object's GID to be altered. On Release 5.0 an Account Manager is really a "GID" Manager.

Ordinary users cannot execute the **:PURGEACCT** or **:PURGEGROUP** commands, but are allowed to invoke the **:PURGEDIR** command. **:PURGEDIR** is the CI command to delete one or more directories. In fact, **:PURGEDIR**, **:PURGEGROUP** and **:PURGEACCT** all behave the same with respect to deleting files under the target directory, group, or account respectively. Since none of these files are individually opened each file will be deleted, regardless of its individual file security, as long as all of the following conditions are true:

- 1) the user has DD access to the file's parent directory, and
- 2) the file is not opened by another process, and
- 3) the file is not being STORED, and
- 4) the file is not a privileged file*.

If any of the above rules are violated the file will not be deleted, and hence the directory, group or account will not be purged.

* SM users are exempt from the privileged file exception. An AM user can successfully purge a group or directory containing a privileged file only if the file's GID matches the AM user's GID.

There are situations where an ordinary user cannot **:PURGE** a file, but can delete the file via **:PURGEDIR**. For example, if

/ACCT/GROUP1/DIR1/DIR2/F1 has an ACD that prevents Mike from accessing it (which it does by default assuming he is not the owner) then he will get CIERR 384 if he attempts to :PURGE F1. On the other hand, if DIR1 and DIR2 grant TD and DD access to Mike then he can :PURGEDIR /ACCT/GROUP1/DIR1/DIR2 :TREE successfully and in the process delete F1. Note that even though Mike has DD permission for DIR1 he cannot :PURGEDIR /ACCT/GROUP1/DIR1 :tree since, in this example, the parent to DIR1 is an MPE group, and he does not have DD access to GROUP1, unless he has save access to the group.

The POSIX *unlink()* function, which is called by :PURGELINK and the Shell's rm command, is somewhere in between :PURGE and :PURGEDIR/ACCT/GROUP in terms of file security. Since *unlink()* does not open the target object, the object's individual access rules are ignored and thus *unlink()* is potentially less restrictive than :PURGE. Also, since the file is not opened, *unlink()* will fail if the file has a lockword-- there is no lockword prompting. To be able to unlink a file requires TD access to the entire pathname and DD access to the file's parent directory, or the equivalent if the file's parent is an MPE group (see the Security section in the "POSIX/iX Primer" chapter). System Managers are granted DD access to all directories. Account Managers are granted DD access if their GID matches the directory's GID. There are situations where a file cannot be unlinked but can be deleted via :PURGEGROUP. For instance, a file with a lockword in an MPE group cannot be unlinked (i.e., the POSIX *unlink()* function will fail, meaning the :PURGELINK and rm commands will fail); however, since :PURGEGROUP ignores lockwords, the file is deleted via :PURGEGROUP. Lockworded files, files with negative file codes, and protected files can never be unlinked, even with SM capability. One last point, if you just want to delete a file and you don't know whether it is an empty directory, symbolic link, or an ordinary file, simply use :PURGELINK.

- ◆ **File Transport** - Network Services (NS) is not POSIX aware in Release 5.0. Therefore, to transfer POSIX named files from one machine to another requires a little creativity.

First, the HFS named files can be copied to legal MPE filenames in MPE groups, and then your site's standard procedures can be followed. Second, a file equation can be created for each HFS named file to be transferred. For example:

```
:FILE x=/test/filex
:DSCOPY *x; filex:nodename
```

Third, FTP.ARPA.SYS can be used to transfer files. FTP is HFS aware on MPE/iX, but using FTP may change current system management

procedures. Fourth, STORE can be used to transfer files via tape.

- **System UDCs** - A UDC file named HPPXUDC.PUB.SYS is part of the Release 5.0 FOS tape. This file is *not* automatically cataloged. It contains several useful UDCs such as: FINDFILE which locates a filename at any depth in the directory. FINDDIR is similar to FINDFILE but finds directories rather than files. LISTDIR shows information about a directory. The PLISTF UDC supports the MPE-escaped syntax for both the *fileset* and output *listfile* parameters. The four UDCs mentioned above were implemented by invoking the :LISTFILE command, and are worth reading to learn more about :LISTFILE. For customers habituated to typing "disk" as "disc", there is a DISCUSE UDC that executes the :DISKUSE command.

System Managers may choose to catalog HPPXUDC at the system, account or user level. The file may be modified to better support your specific requirements.

- **System Logging**- The new system logging events mentioned in the "POSIX/iX Primer" chapter are enabled via SYSGEN and are available only in Release 5.0.

Event 155 controls whether or not to recognize the file open (144) and file close (105) events when the target file is an HFS directory. If the file is root, an account, or an MPE group no logging occurs regardless of the setting for event 155. If event 155, 105 and 144 are all enabled then every time a directory is opened and closed two log records are generated. The shell's *ls* command, :LISTF and :LISTFILE open and close each directory listed, so there is potential for large log files. Note: not all directories in the pathname are logged, only directories that need to be "officially" opened. For example, :LISTFILE /a/b/c ;tree only has to open directory "c" in order to read its entries (object names); whereas, :LISTFILE /a/b@c ;tree must open directory "a" (to find if a has any "b@" entries), in addition to directory "c".

Event 127 enables *chdir* logging. This may be important since the filenames logged for the other events may not be absolute pathnames. With *chdir* logging operative, these relative pathnames can be "qualified" with the user's CWD. *Chdir* logging takes place for the POSIX *chdir()* function and the :CHDIR command.

Event 129 enables *chown* logging. This event is triggered whenever the POSIX *chown()* function or the :ALTFILE command are successful. Event 129 should be enabled for all systems that currently log other security related events. The POSIX *chmod()* function and the :ALTSEC command are captured via the ACD change logging event (138).

As mentioned in the "*POSIX/iX Primer*" chapter, some HFS events do not log fully qualified (absolute) filenames. Specifically, ACD change (138), file open (144), chdir (127) and chown (129) do not qualify any of the logged filenames (even though the name may be converted from an MPE to a HFS name). If your site will be using HFS names, and will enable any of the above logging events, and will require absolute pathnames, independent of how the user specified the names, then you will need to interrogate the log files for job/session logon (102), and chgroup (143) events. This is necessary in order to construct absolute pathnames for any relative filename logged by events 138, 144, 127 or 129.

- ♦ **3rd-Party Software** - A concern of System Managers is that the third party tools that they rely on for their production work will not function correctly in an HFS environment. HP has been working with many software vendors to teach them about our POSIX release, and to help them make their products POSIX aware. Obviously, many of the tools will not be fully operative in a POSIX environment in the 5.0 time frame. The important question to ask is "Does this tool need to handle HFS objects for my business to be successful?". For most customers the answer is "no", since most sites do not have immediate POSIX requirements. Those customers dependent on POSIX need to ensure that their corresponding business tools work satisfactorily in a POSIX atmosphere. HP will be able to provide information about third party products that are POSIX aware.

Conclusion

The POSIX release of MPE/iX is of significant value to customers that desire to further protect future software investments by developing and purchasing open applications. For MPE users who have always wanted another group under a group, or have complained that they need just a few more characters for a meaningful filename, or need to prevent ordinary users from being able to do a LISTF and see every file on the system, the POSIX enhancements to MPE should be welcomed.

POSIX on MPE/iX benefits from the strengths of MPE: excellent price/performance, high availability, superior OLTP, and number one in support..

POSIX doesn't come for free though. System Managers need to understand the fundamental concepts discussed in this paper. Additional disk space is required. Third party software suppliers need to be told your expectations for POSIX support. The transition to taking advantage of POSIX and the hierarchical file system may not be easy; however, for most customers it will be worth the effort.

Glossary

\$GROUP - A new ACD subject for which access control can be specified. \$GROUP differs from an *@account* subject because a file's group ID (GID) can be altered, and thus \$GROUP access control can refer to a different collection of users over time.

\$GROUP_MASK - A new ACD subject needed to make ACDs a viable POSIX alternate security mechanism. \$GROUP_MASK is an additional filter for determining a file's group class access. It may be set explicitly via `:ALTSEC`, or indirectly via the POSIX `chmod()` function.

\$OWNER - A new ACD subject for which access control can be specified. \$OWNER differs from a *user.account* subject because a file's owner ID can be altered, and thus \$OWNER access control can refer to a different owner over time.

ACD - Access Control Definition - A security mechanism whereby all access control to an object is defined as part of that object. ACDs are more expressive than matrix security since certain accesses can be granted to individual users or groups of users. If a file is protected by both an ACD and a lockword, the lockword is ignored. ACD is a proprietary name for Access Control Lists (ACLs) which will be the key security component of POSIX.

account - An MPE directory directly under root. All account names follow traditional MPE name rules (e.g., 8 character maximum length, begins with a letter, no special characters, etc.). Accounts still contain capabilities, matrix access and limits.

AM capability - Account Manager - Users with AM capability can create user and groups within their logon account. They can also access all files in their account. In POSIX terms, the Account Manager can access all files whose file group ID matches their user GID.

API - Application Programming Interface - An external, documented procedure.

byte stream file - A file without any formal record structure. Each logical "line" in the file is terminated by the newline (linefeed) character.

chmod - A POSIX.1 function to change read, write and execute access for a file or directory.

chown - A POSIX.1 function to change the ownership and group ID of a file.

CI - Command Interpreter - A program (CI.PUB.SYS) created at logon time which is the MPE equivalent of the POSIX shell. The CI is responsible for prompting, reading command input, command execution, servicing break and error handling.

compatibility mode (CM) - A CM program or procedure that emits classic 3000 instructions which are emulated or translated to the native instructions set.

component (of a pathname) - A name delimited by a '/'. Typically a component is a directory name, except when it is the last component, where it could also be a filename. "Last" refers to the rightmost component of a POSIX pathname.

CWD - Current Working Directory - The directory, which often is your logon prompt, where you are currently located. Moving your CWD has no effect on your file access-- it is only a naming shortcut.

Device link - A file that is linked to an LDEV number such that opening the device link is identical to opening a device via its LDEV number.

Directory - A file that has a directory record structure. Generically, MPE groups, accounts and root are directories. In fact these directories have special name length restrictions (16 characters vs. 255) since these objects are part of the additional directory structure. HFS directories must be protected by an ACDs, but in 5.0, root, accounts and MPE groups cannot have ACDs.

IFO - A type of file with the property that data is always read and written in a first-in-first-out manner.

File - File can be used to describe a directory, byte stream, symbolic link, device link, etc. Basically all objects in the HFS are implemented as files.

File group - The class of users who are not a file's owner, but match one of the *GROUP*, *user.account*, or *@.account* ACD subjects. These user's GID match the GID of the file. In MPE terms they are in the same account as the file.

File owner - The class of users whose UID match the owner ID of the file. In MPE terms they are the creator of the file.

File other - The collection of users who are not file owners nor members of a file's group class. In an ACD pair "*@.@*" is the file other subject.

OS - Fundamental Operating System - The core operating system without any optional subsystems.

GID - Group IDentification - A GID identifies users as members of a file's group class. These users can have unique file access defined for them. POSIX defines a GID as a number. It is simulated as a number in MPE, but the user's account name is currently the basis for security.

Group - An MPE directory directly under an account. MPE groups have access rules and capabilities associated to them. For the POSIX group definition see **file group** and **GID**.

HFS - Hierarchical File System - The MPE/iX directory and file system which allows files and directories to be at an arbitrary level under the root directory.

HFS is often used synonymously with "POSIX-named" to indicate that the object is not part of the traditional MPE file, group and account structure.

HFS syntax - see **POSIX syntax**.

HPGID.PUB.SYS - The name of the POSIX group database file. This is where all account names and their associated GID are stored. This file is automatically created when updating to 5.0. In 4.5 a utility program (PXUTIL.PUB.SYS) is used to synchronize the databases with the actual directory. This synchronization is not needed in 5.0.

HPUID.PUB.SYS - The name of the POSIX user database, where the *user.account* names and associated UID numbers are stored.

lockword - A password for a file. Must begin with a letter and cannot exceed 8 characters in length. *:BUILD file/lock* creates FILE with the lockword LOCK. File access via MPE syntax supports inline lockwords (e.g. *:PRINT file/lock*), or

will prompt for the lockword if it is omitted. POSIX and MPE-escaped syntax don't support lockwords and cannot access a lockword protected file. Lockwords are ignored if the file has an ACD.

matrix security - A mandatory security mechanism where access is established at the account, MPE group and file levels. Typically, access is more restrictive as you move down from account to group to file. Matrix security allows a System Manager to shut off a certain access to all users by disallowing it at the account level. A released file is exempt from matrix security.

MPE syntax - Traditional filename rules: must begin with a letter, no special characters, less than or equal to 8 characters in length, etc. See Figure 3.

MPE-escaped syntax - Either MPE syntax or POSIX syntax if the filename begins with a "." or a "/". Many intrinsics and CI commands accept MPE-escaped syntax.

native mode (NM) - Native mode execution means that a program or procedure directly calls the machine's native instruction set.

NL.PUB.SYS - The operating system's XL (executable library). The NL is the final point for binding external procedures.

object - A generic term for files, directories, root, MPE groups and accounts.

owner - See **file owner** and **UID**.

pathname - The POSIX equivalent to a filename. Pathname can refer to the complete, "fully qualified" name (absolute pathname), or the name relative to your CWD (relative pathname).

pipe - A pipe consists of two file descriptors connected such that data written to one can be read by the other in a first-in-first-out manner.

POSIX syntax - A top-down, case sensitive name that can contain "_", "-", "." characters, numbers, and upper and lower case letters. A POSIX name cannot start with a "-". See Figure 3.

privileged file - A file that either has a negative file code or was created with a privilege level less than ring level 3 via HPFOPEN item 38. Privileged files typically cannot be purged by the CI or POSIX shell.

protected file - A file that cannot be purged or cannot be written to.

FPROTECT.PUB.SYS lists all protected files on your system. A file that cannot be written to also cannot be purged. A file designated "purge protected" may be modified. FPROTECT, itself, is "write protected".

released - a file is released via the :RELEASE command, which disables group and account level security. The inverse operation is performed by the :SECURE command.

root (/) - The origin of the directory structure. Root cannot be protected by an ACD. Object names under root cannot exceed 16 characters in length. Only SM can create objects under root. If a pathname begins at root it is an absolute pathname.

shell (.2 Shell) - A program that serves the purpose of the Command Interpreter, but is POSIX compliant. Currently the shell must be run from the CI.

SM capability - System Manager - Users with SM capability are in charge of the file system. They can create accounts and directories under root. They can access all files on the system.

Streams - A streams device is a bi-directional, character oriented connection between a file and typically a device driver.

Subject (of an ACD) - The target of an ACD rule. For example, `:ALTSEC file1; pacd=(R:mgr.test)` restricts all users logged on as *mgr.test* to being granted only read access to FILE1. *Mgr.test* is the ACD subject.

Symbolic link - A file that "points" to another object (e.g., file, group, account, directory, symbolic link). When a symbolic link name is encountered in a filename it is substituted with its target name.

Type manager - A file system module responsible for handling all file system operations for a particular type of file. Operations include: read, write, control, delete, etc.

UDC - User Defined Command - A collection of zero or more CI commands given a name, which must begin the first line of the UDC. One or more individual UDCs are placed in the same file, which is "cataloged" by the SETCATALOG command. The CI searches for UDCs before built-in commands and prior to command files.

UID - User IDentification - A unique identification for every user on the system. POSIX implements this as a number. MPE/iX currently maintains both a number in the HPUID.PUB.SYS database for use by POSIX applications (and process signals), and a string ID in the form of *user.account* for all other needs.

User id - see **UID**.

Wildcard - Traditional MPE wildcard characters are: "@" - match zero or more of any legal character, "?" - match a single legal character, and "#" - match a single numeric character. POSIX syntax expands the range of legal characters to include lowercase, "_", ".", and "-". A range or group of characters is expressed as "[abc]" or "[a-c]", which both indicate to match the letters "a", "b" or "c".

Paper #8006
Unix Boot Disasters--Planning your Alternatives
Interex Conference, September, 1994

Dennis McClure
Hewlett Packard Response Center
2000 South Park Place
Atlanta, GA 30339
(404) 850-2742

Introduction

Most of the time, we don't think about the complexities of booting an HP9000. It just always works, time after time. But if you are a typical system administrator, you worry about the unexpected. It's your job to plan for the worst, and to know how to deal with it.

What are you going to do on the day your HP9000 fails to boot? This paper will discuss the possibilities that are available. Your choice will depend on what went wrong with the normal boot.

Here are some faults that make a boot fail, and what we can do:

- * Bad/missing kernel file--use a different kernel.
- * Bad/missing programs, scripts, or configuration files on the root disk drive--use different boot options.
- * Fatal error in boot area or file system--use a different boot device.

The above faults are all similar in that they are matters of data that is not correct on the disk. The methodology of the fix is always to gain access to the disk, then rewrite or correct the data, or at least back up critical files. This paper assumes that the fault is not in hardware. When hardware may be the cause, this methodology will give us ample opportunity to observe what works and what doesn't, see hardware error messages that occur, and make a clear diagnosis of the real fault. At that point we would detour out of this methodology until the hardware is fixed.

BAD/MISSING KERNEL FILE

The kernel is a file in the root directory, called /hp-ux by convention. It is assumed that an alternate kernel will always be present in a file called /SYSBCKUP. When utilities like sam create a new /hp-ux file, the old one is moved to /SYSBCKUP. In this case, /SYSBCKUP will always be the next most recent version of your kernel. You can manage this better yourself. For example, two bad kernel generations in a row could leave you with no good copy. It's a good idea to make your own copy with a name of your own choice. Also, if you know that /hp-ux is the best possible kernel for your system, then /SYSBCKUP would serve you better if it were an exact copy of /hp-ux.

The kernel file can be removed or changed while you are up and running. You won't know there is a problem until the next attempt to boot.

When there is something wrong with /hp-ux, you can boot a different kernel. See below, after the next topic, for instructions how to do it.

BAD/MISSING PROGRAMS, SCRIPTS, OR CONFIGURATION FILES

The HPUX boot procedure includes many complicated steps besides loading the kernel. The program /etc/init runs. It reads /etc/inittab, which indicates what run level to initiate, and what things to launch for that run level. For a multiuser boot, many scripts run from the /etc directory: bcheckrc checks file systems and then runs fsck(1M) if necessary, brc cleans up flags and the mnttab file, and rc initializes a long list of system processes and resources. File systems are mounted and network software starts. Then gettys are spawned and the console login is processed. Many of the files used during the boot process are idle between boots, making it possible for changes or deletions to happen while the system is up and running. The next time you try to boot, you might find it fails due to problems with these functions.

You can work around many of these problems by booting into single user mode, and then fixing the things that are wrong.

Booting an alternate run level or kernel file from the root disk drive

Series 700's:

- 1) Initiate a normal boot (usually by turning the power off and on).
- 2) You will see "Selecting a system to boot. To stop selection process, press and hold the ESCAPE key." Press ESCAPE briefly rather than holding the key down. Allow the search for potential boot devices to complete. This is a poll of every device on the scsi bus.
- 3) Boot from the root disk to the ISL> prompt, using the ipl parameter:

```
b p0 ipl (assuming that the root disk is
listed as p0 in the search for potential boot
devices)
```

If you need help identifying which device contains bootable software, use the "search" option from the menu, then enter the correct device:

```
p0 ipl
```

- 4) The system will boot to the ISL prompt. This is where you can specify what run level to boot into, and what file to use for the kernel:

```
ISL> hpux [-i<run-level>] <kernel file>
```

Examples:

For booting /SYSBCKUP:

```
ISL> hpux /SYSBCKUP
```

For booting /hp-ux into single user mode:

```
ISL> hpux -is /hp-ux
```

For booting /hp-ux.old into run-level 3:

```
ISL> hpux -i3 /hp-ux.old
```

Series 800's:

- 1) Initiate a normal boot (use "<ctrl>b rs" on newer systems to avoid having to turn off and on the power).
- 2) Interrupt the boot when you see "press any key within 10 seconds".

"Boot from primary boot path (Y or N)?" Yes

"Interact with IPL (Y or N)?" Yes

- 3) At the ISL> prompt, display the normal autoexecute command:

```
ISL> lsautofl (or the abbreviation "lsa")
```

You will see something like one of the following, depending on your 800 model and the use of LVM:

```
hpux (;0)/hp-ux
hpux disc1(4.0.2;13)/hp-ux
```

- 4) Enter the command essentially as you see it, changing or adding as needed. To change the kernel file name, change what appears after the "/". To indicate a run-level, insert "-i<run-level>" after the hpux command.

Examples:

For booting /SYSBCKUP:

```
ISL> hpux (;0)/SYSBCKUP
```

For booting /hp-ux into single user mode:

```
hpux -is (;0)/hp-ux
```

For booting /hp-ux.old into run-level 3:

```
ISL> hpux -i3 (;0)/hp-ux.old
```

Listing files from the ISL> prompt

When there is doubt about the kernel file in the root directory, you can list the files that are present:

```
ISL> hpux ls /
```

Look for evidence that this is the root file system. You should see the typical directories, like /bin, /dev, and /etc, and you should see the files /hp-ux and /SYSBCKUP. You can look for other files that you suspect are missing, such as:

```
ISL> hpux ls /etc/init*
```

Problems in Logical Volume Manager

Series 800's can use Logical Volume Manager (LVM) as the method of disk organization and access. When LVM structures on the disk are incorrect, or when disks are not responding properly, the system can fail to boot. LVM error messages will indicate what type of problem is happening.

One requirement of LVM is that more than half of the physical volumes must be present and responding. If not enough are available, the boot will fail this "quorum" test. There is a special boot option, "-lq", that allows the system to ignore the LVM quorum. This boot option is used at the ISL prompt, like the "-is" option for single user mode.

Boot from the primary path and interact with IPL.

```
ISL> hpux -lq (;0)/hp-ux
```

Once the system is up, you can pursue problems on individual disks.

Another possible problem involves the Boot Data Reserved Area (BDRA) and the LABEL file. Each time you change the configuration of the root volume group, both should be updated. You can update them with the following command, assuming default names:

```
# lvlnboot -R /dev/vg00
```

If there was a change to the root volume group, and the BDRA or LABEL file was not updated, the next boot may fail with "could not configure root VG". There is a special boot option, "-lm", that allows the system to boot in LVM maintenance mode.

Boot from the primary path and interact with IPL.

```
ISL> hpux -lm (;0)/hp-ux
```

When the system comes up, you will have to activate the root volume group, update the BDRA and LABEL file, and reboot:

```
# vgchange -a y /dev/vg00  
# lvinboot -R /dev/vg00  
# reboot
```

There is an important LVM backup function that is often overlooked--the `vgcfgbackup(1M)` command. It backs up a volume group configuration from the physical volume to a file `/etc/lvmconf/<vg-name>.conf`. If you lost one or more logical volumes, this file could be used as input for the command `vgcfgrestore(1M)`. It would recreate the LVM structures on the disk. Next you would recreate the file systems and restore files. Lacking this backup file, you would have to recreate the volume group from scratch, and the only way to do that for `vg00` is to reinstall! The backup command should be done for each volume group after any change to the volume group.

```
# vgcfgbackup /dev/vg00
```

FATAL ERROR IN BOOT AREA OR FILE SYSTEM

The HP support tape

When the root drive is unbootable, there are still many options left to try. The one we use most often when giving help from the Response Center is the HP support tape (or CDROM). This tape was originally developed by the HP Customer Engineer Organization as a hardware support tool. It was not intended to be used by customers, even during Response Center telephone help. On the other hand, we think it is the best tool available, so we use it.

The support tape is available for 700's and 800's. It is shipped with each order of software media. But, with "Instant Ignition" systems (software already installed on the disks), many are sold these days without software media. It is critical that software media is also purchased, so that it is possible to reinstall or recover from unexpected disasters.

The support tape is bootable. It contains a minimal kernel and a number of useful utilities. While booted from the support tape, "/" refers to the support tape's root directory, which is managed in main memory.

Although there are many files available on the tape, only a few are loaded into this memory-based root file system, due to limited space. You may load additional files as needed, subject to the space limitations. The "ls" command, for example, is not present unless you load it, because it is very large. We use "echo *" as a substitute. There are some automatic recovery options available, but I generally do not use them. I prefer to use manual techniques to change only those things that need to be fixed.

The general approach is to boot, then escape to a shell. From there, we attempt to mount the root disk under a mount point such as /mnt1. Sometimes the disk file system has to be fixed with fsck(1M) before it can be mounted. It takes some expertise and experience to know what device file name to use in referring to the root disk. Remember, at this time the /dev directory we are using is the one that came from the support tape, and the device file names may not match the names you are accustomed to using.

Once mounted, we may need to change our point of view to the root disk using the chroot(1M) command (change root). Then it is possible to fix a number of faults that may be responsible for the failure to boot. We can fix a defective boot area, where the loader utility resides, with the mkboot(1M) command. We can gen a new kernel if needed using normal procedures. Sometimes we have to fix permissions, create device files, edit scripts or configurations, or restore files from a backup. If this can be done successfully, the final step is to reboot with default options from the root disk.

Before beginning any repair, it is good to verify that the damage does not extend further than is feasible to repair. It is terribly disappointing to hack through the repair of one file only to discover that all the other files are missing.

Support tape examples

The following is an example of the steps to boot the support tape on a 700 at 9.01 and repair the boot area of the root disk:

- 1) Boot from the tape device to the ISL> prompt, using the ipl parameter, then boot the support system:

b p2 ipl (assuming that the tape drive is listed as p2 in the search for potential boot devices)

ISL> support

- 2) When the support tape menu is displayed, choose "load a file". When prompted for what files to load:

```
ls chroot
```

- 3) When the files are loaded, return to the menu and select "exit to shell" to mount the root disk (/mnt1 is already provided as a mount point).

```
# mount /dev/dsk/c201d6s0 /mnt1 (assuming the
root disk is on single-ended scsi, address 6)
```

If fsck is needed, run it; then redo the mount command:

```
# fsck -y /dev/rdisk/c201d6s0
# mount /dev/dsk/c201d6s0 /mnt1
```

Inspect to see if it looks like a root file system:

```
# ls /mnt1/*
```

- 4) Switch to the disk file system and rebuild the boot area:

```
# chroot /mnt1 /bin/sh
# ./etc/mkboot /dev/rdisk/c201d6s0
```

- 5) Switch back to the memory-based support menu:

```
# exit
# menu
```

- 6) Choose reboot from the menu, and let it boot as normal.

The following is an example of the steps to boot the support CDROM on an 800 at 9.0 and gen a new kernel:

- 1) Boot from the CDROM drive and interact with ipl, to the ISL> prompt.

```
ISL> supported
```


When the boot is nearly completed, watch for it to display the device file name of "rootdev". It scans disk devices from low scsi addresses to high, assigning lu numbers starting with zero. If you have three scsi disks, use lvm, and the root disk drive is at address 6, the device file name will be /dev/dsk/c2d0lvm.

- 2) When the support tape menu is displayed, choose "exit to a shell", and make a mount point. (Command files will be available from the CDROM without loading them, because the CDROM is mounted automatically.)

```
# mkdir /mnt1
```

- 3) Fsck if necessary, mount the root disk and inspect as above.
- 4) Chroot as above. Use normal commands to gen a kernel and move it to /hp-ux.
- 5) Run exit, menu, and reboot as above.

The recovery tape

The recovery tape is the other kind of bootable tape that can be used. It must be created by the user on series 300's, 400's and 700's, using the mkrs command. Depending on the fault that prevents a normal boot, it is possible that the recovery tape would be useful fixing the fault, particularly on versions 9.0 and later. On the other hand, we find that the auto-recovery feature requires some manual steps that are not clearly documented. It copies a pre-defined set of files from the tape back to the disk, putting them in the /tmp directory, in a subdirectory named recovery.<mmdd>. It is necessary for you to copy these files back to the regular places they belong. It is possible to escape to a shell and execute manual procedures, but there are fewer capabilities than on the support tape. Also, it is critical that you create a new recovery tape each time the system is modified in any way, including each time the kernel or critical configurations are changed. If you are depending on the recovery tape to save you from a disaster, it is highly recommended that you find a way to test it first in a non-critical environment.

Based on preliminary information, HPUX 10.0 will have improvements in the recovery tape and support tape

strategies. It will be worthwhile to check the details when they become available.

Diskless clusters, a special bonus

There may be one more alternate boot possibility that already exists and is ready for you to use--a diskless cluster server. In HPUX 9.0, diskless cluster software is available for 300's, 400's and 700's. It provides a capability for client systems to boot over the network from the server's root file system. The original idea was to save the costs of disk drives for the clients. This advantage is disappearing these days as disk costs are shrinking. Another standard advantage is that system administration can be centralized on the server, and this is probably the biggest reason for using diskless clusters today. A third advantage is often overlooked: cluster servers can provide redundancy for booting.

In many ways, booting a problem system from a cluster server is better than using the support tape. You have the advantage of running a full-sized kernel and a complete file system, with all Unix capabilities available. Once booted from the server's disk, you would use the same repair procedures as in the support tape section above.

If a cluster client has trouble booting and the server is ok, then it is possible to login on the server and manipulate the client's files. If it is the server that will not boot, redundancy depends on having a least one more server. You can configure each server as a client on a different server. If the problem system is a standalone system, and you have a diskless server available, you could temporarily configure the problem system as a client.

All members of the cluster must be on one "local lan segment". Repeaters, hubs, and bridges between members are ok, but gateways are not. This is due to the fact that communication is based on link level addresses rather than IP addresses.

It is very easy to change a cluster server's configuration to add a new client. It is a bit more effort if the problem system has to be moved so that it can be attached to the server's local lan segment. You might physically move the problem system next to an existing client, and then temporarily borrow that lan connection.

It would take considerably more time if the cluster server was not already in place. You would have to install cluster server software using update(1M), then configure both the server and the client. If you are not already familiar with clusters, you may not want to undertake this task in an emergency. However, it is a feasible technique, and sam does a good job with the configuration.

Based on preliminary information, there will be significant changes in HP-UX 10.0. Instead of HP's proprietary diskless cluster software, diskless capability will be provided as part of nfs. Support will be added for series 800's as servers, but not as clients. IP addressing will be used, so a cluster will not be confined to a local lan segment. One hop over a gateway or a router will be permissible.

Spare root disk drive

The larger your organization, the easier it may be to plan your boot alternatives. If you have a lot of systems, it may be cost effective to prepare a spare root disk drive. It should be an external disk with the operating system fully installed. It would be handy if you could set the disk's address to one not used on any of the systems. It could sit on the shelf like an insurance policy, ready to attach to whichever system might need it. Boot from it, and fix damaged files as above.

A good choice for an external drive is a magneto-optical read/write disk. Since the media is removable, you could have several versions of the operating system, each on its own platter. You could also use the drive for multiple other purposes.

At the Response Center, we often need to test situations on any one of many currently supported HP-UX versions, so we install different versions to different disks on the same system. The leftover space on each disk is available for user data. We boot whichever version is needed, and mount all the other disks as secondaries. It was not our original purpose, but we also get boot redundancy this way. You can plan this for yourself by installing the smallest possible system to a secondary disk, to be used in case of disaster on the root disk.

In your plan, the "spare" root disk drive might be a root drive you can borrow from a different system. It would be easiest if the drive is in an external cabinet, but

this is not necessary if you have the hardware expertise to rearrange internal disk drives. Alternatively, you could take the bad drive to a different system, where it could be attached as a secondary drive.

Before physically rearranging disks, there are some important cautions about the rules of scsi buses. No changes should be made while the system is powered up. Shutdown and power off the system first, then power off all peripherals on the scsi bus, and only then make any cabling changes. Reverse the sequence to power up. If these rules are not followed, it is possible to corrupt data on the disks. Also, be sure to avoid any duplicate scsi addresses, which are another possible cause of data corruption.

Reinstalling software

Doing a reinstallation might be part of your plan, rather than the last resort. You could install a minimal system to a disk other than the root disk, boot the minimal system, and attempt to save the root disk. You might use any available disk for this purpose, even one that has necessary data, if you first back it up before using it for the install.

You could organize your disk environment so that reinstalling on the root drive is quick and harmless. Considerations might include:

- * Keep only HP-supplied installed directories on the root drive, and all user data on secondary drives.
- * Be prepared to recreate or restore every file that was customized or configured for your operation.
- * Be prepared to restore or reinstall all patches applicable to software in the reinstalled directories.

In the case of customized files, you could store copies in a special directory on another disk, or back them up separately, for easy restoring. The difficulty is being sure that you have identified all the pertinent files that various products modify. It would be good to test this procedure, to be sure all customized files are identified. In the case of patches, some program files and shared libraries may not be replaceable by restoring, due to being in use. You can replace them if booted from another source, but that defeats the purpose of this approach. Reinstalling the patches may be easier.

Unsupported boot devices

HP will support systems that were installed from HP media using the standard install procedures. You may use other handy procedures, like the ones following, if you do not require HP's support for it.

Creating a bootable backup (Series 700's)

A bootable backup tape, like they use on an HP3000, would be a very handy thing. Unfortunately, the only technique I have ever heard about is considered an unsupported one. On the other hand, it is listed in the man pages under `hpux_700(1M)`, restore option. Also, it was supported to the degree that patch PHCO_3927 was issued to make it work in 9.03.

To copy a boot area to a tape, and then copy the root file system (assuming standard tape and disk configuration):

```
# dd if=/usr/lib/uxbootlf.700 of=/dev/rmt/0mn bs=2k
# sync
# sync
# dd if=/dev/rdisk/c201d6s0 of=/dev/rmt/0m bs=64k
```

To boot from the tape and restore the entire root disk after a disaster:

Boot from the tape device to the ISL> prompt, using the `ipl` parameter, then boot the tape system:

```
b p2 ipl (assuming that the tape drive is
listed as p2 in the search for potential boot
devices)
```

```
ISL> hpux restore disk(scsi.6;0)
```

A number of cautions are required. The disk you copy to should be the same model disk that you copied from. A fresh backup has to be made any time there are changes affecting this disk. The `dd` command does not handle I/O errors, so if I/O errors occur the results will be unpredictable and incomplete. The restore is not capable of spanning two different I/O controllers, such as a differential scsi disk and a single-ended scsi tape.

Creating a bootable disk

The following is an unsupported technique for creating a bootable disk. It could be useful on hardware such as a magneto-optical disk drive. This procedure was written for 700's by an engineer of the Belgium Response Center, with minor changes by me. It could be adapted for series 800's.

Initialize the disk:

```
# mediainit -v /dev/rdisk/mo
```

Create a file system with 32 MB of swap:

```
# newfs /dev/rdisk/mo hpS6300.650A_32MB
```

Make the MO disk bootable:

```
# /etc/mkboot -v -u /dev/rdisk/mo
```

Mount the disk and copy the root file system:

```
# mount /dev/dsk/mo /mo
```

```
# find / -depth -hidden -xdev |  
> cpio -pvdxulm /mo &
```

Modify /mo/etc/checklist so it properly describes the disk drives that will be present when the mo is booted.

Conclusion

The lengths to which you are willing to go to repair a damaged root drive will depend on how valuable the data is you can save. Regular backups are the usual indicator of valuable data, but that value should also influence the investment you make in disaster planning.

###

EFFECTIVE NETWORK MANAGEMENT ON MPE V

ROSS MARTIN / GEORGE CLARK

HEWLETT-PACKARD COMPANY
SOFTWARE TECHNOLOGY DIVISION
ROSEVILLE, CALIFORNIA
916-786-8000

SEGMENTING NETWORKS USING SUBNET MASKS

IP addresses consist of four bytes of information. These are typically expressed as aaa.bbb.ccc.ddd, and the byte values are shown in decimal. For example, an IP address of 15.135.6.241 would be equivalent to:

00001111 10000111 00000110 11110001

To manage communication among nodes, it must be possible to determine whether the nodes are on the same network. To accomplish this, IP addresses are divided into two parts: a network portion and a node portion. Three classes of IP addresses are available to accommodate both users with large numbers of networks and few nodes as well as users with few networks and large numbers of nodes. For the generic address aaa.bbb.ccc.ddd, the capabilities of the three classes are defined as follows:

	<u>NETWORK PORTION</u>	<u>NODE PORTION</u>
Class "A"	aaa	bbb.ccc.ddd
Class "B"	aaa.bbb	ccc.ddd
Class "C"	aaa.bbb.ccc	ddd

To make the classification self-contained (no "A", "B", or "C" necessary) the first three (left-most) bits are restricted:

PERMISSIBLE BIT VALUES (x = DON'T CARE)

Class "A"	0xxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
Class "B"	10xxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
Class "C"	110xxxxx xxxxxxxx xxxxxxxx xxxxxxxx

Thus, a class "B" address can only have network values ranging from 128.000 through 191.255, for a total of 16,383 distinct networks.

Users often desire network or node addresses which extend beyond the limits of the three classes. For example, a network with many personal computers quickly exhausts the available number of class "C" nodes. Other situations which can

exceed the capabilities of the class definitions include:

- > allowing more nodes than Class "B" or Class "C" permit
- > allowing more networks than Class "A" or Class "B" permit
- > partitioning nodes on a network into groups for security reasons.
- > routing traffic onto specific links for throughput reasons
- > managing gateway access to remote networks

These requirements are met with subnetwork masks.

Subnet masks are 4-byte words which are used to redefine the network portion of an IP address within the local machine. A bit-wise logical OR is performed on the address and the mask. The result is the new network number in the eyes of all machines using the same mask. Consider this address and mask combination:

IP address: 17.001.002.003
subnet mask: 240.000.255.000

A bit-wise OR results in the logical network number 16.0.2.0:

IP address: 00010001 00000001 00000010 00000011
subnet mask: 11110000 00000000 11111111 00000000

logical net: 00010000 00000000 00000010 00000000

If a node with this address/mask pair receives packets from an initiator having the source address 17.1.3.3, those packets will NOT be accepted; they come from a different logical network. Likewise, packets coming from 20.1.2.3 WILL be accepted. Notice that since masks are defined locally, the source mask does not matter at the destination end. However, when a reply is sent back, the original initiator must view the respondent as being on the same logical network, and the initiator's mask is then crucial.

MPE nodes always use subnet masks. If a mask is not configured by the Network Manager, a default is assumed. The defaults are obvious:

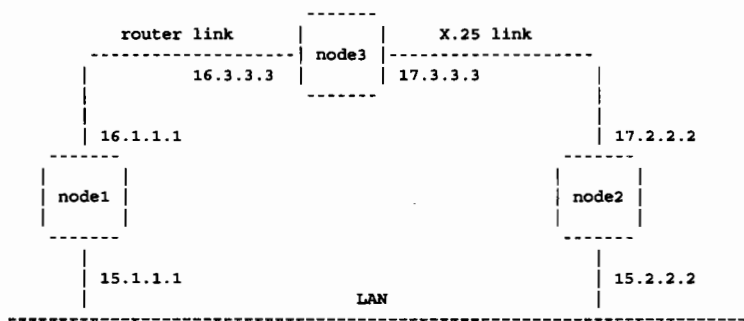
DEFAULT SUBNET MASK

Class "A"	255.000.000.000
Class "B"	255.255.000.000
Class "C"	255.255.255.000

MPE places no restrictions on the masks. So to double the number of nodes available on a class "C" network, 255.000.255.000 and 255.255.000.000 would both be acceptable (as would with many other possibilities). Note that a mask can cause the left-most bits of the logical network number to be partially altered, destroying the class information. This is one reason that NMMGR forces the user to specify an "A", "B", or "C" along with the numeric addresses.

Using masks in a particular way could violate external requirements even if MPE permits such use. For example, suppose a corporation has been assigned the class "C" network 192.18.42.0 by a governing organization. Within that corporation, there is a need to attach about 500 nodes, so the network manager defines a subnet mask of 255.255.0.0 to use the third byte for node addresses. The node address 192.18.50.1 would seem legal, but would actually collide with an address assigned to a different corporation. It would be critical that this node's traffic be confined within the local corporation's environment.

The next example shows an academic situation in which subnets are used to alter the normal flow of nodal communication.



Assume at node2 a mask of 255.255.000.000 is configured for its LAN interface but node1 has only default masks defined. Now suppose node1 initiates a connection with node2 on the LAN. The TCP SYN packet will arrive at node2 and be accepted because the IP header will have a destination address of 15.2.2.2, which is one of node2's aliases. But when node2 tries to resolve the reply path, it will decide that 15.1.1.1 is not on the same network (15.2.0.0) as the LAN.

Address calculation at node2:

local address:	015.002.002.002	
mask:	255.255.000.000	

logical net:	015.002.000.000	
remote address:	015.001.001.001	==> different networks
mask:	255.255.000.000	

logical net:	015.001.000.000	

Since node1 does not appear to be directly-connected to node2, node2 searches its internal routing tables for a gateway. If 15.1.0.0 is configured as a reachable network with mask of 255.255.000.000 in the X.25 Network Interface

at node2, the ACK for node1 will be sent to node3 as a gateway. If node3 has been configured to reach 15.1.0.0 (same mask) through its router, the packet will eventually reach node1 and be accepted. Afterwards, all packets from node1 to node2 will use the LAN and all packets from node2 to node1 will take the X.25/router path.

Notice that subnetting which involves gateways can be complicated and sometimes produce unexpected results. In the example above, if node3 did not have its masks correctly defined, the packet from node2 to node1 could be echoed back to node2. The connection attempt would timeout at node1, and packets could bounce between node2 and node3 indefinitely.

INCREASE NUMBER OF GATEWAYS SUPPORTED BY MPE V TRANSPORT

OVERVIEW

Certain complex catenets require significant numbers of gateways for interconnection of the various networks. The routes among these networks must be configured using NMMGR to specify the reachable networks and the directly-connected gateways.

Currently, the maximum number of gateways is set at 64 on MPE V. Furthermore, the number of gateways multiplied by the number of remote networks reachable from the local node must be less than 2000:

$$(\text{NUMBER OF GATEWAYS}) * (\text{NUMBER OF REMOTE NETWORKS}) \leq 2000$$

Finally, the total number of remote networks which can be reached from any specific node must be fewer than 256.

These limits are determined by the sizes on certain internal data structures in the NS Transport. These data structures contain the routing information required to select a gateway for a specific destination network and to update this information upon certain network events such as ICMP redirect messages.

There are three cases which can create restrictions in network configuration:

1. There are only a few gateways on the network, but they collectively reach a large number of remote networks. In this situation, the the gateway limit of 64 would probably not be hit before the remote network limit of 256. Although configuration of the remote networks is somewhat tedious, the limit of 256 is seldom reached in practice.
2. There are about as many gateways as there are remote networks. For example, if each gateway reaches one and only one remote network, the 2000 rule quickly limits the number of gateways to about 40 (the square root of 2000).

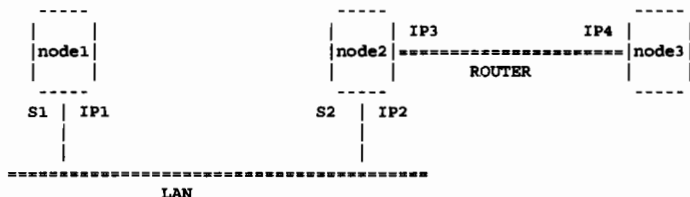
3. There are a few remote networks and several gateways which can reach each one. This indicates a desire for redundancy or alternate routes to the remote nodes. In these situations, the limitation of 64 gateways could be a factor.

Case 2 and case 3 represent the typical configuration challenges, although most sites actually have case number 1.

HOW GATEWAYS FORWARD PACKETS TO REMOTE NETWORKS

When a packet is to be sent to a remote node through a gateway, the NS Transport first determines the IP address of an appropriate gateway. This is accomplished using the internet information for each Network Interface that is configured using NMMGR. The gateway's IP address is then used to resolve the level-2 address (i.e., the station address on a LAN, the phone number on a router, or the SVC information on an X.25 network). On an MPE LAN, this is commonly done with Probe or ARP requests to the IP address of the gateway, and would look something like 08-00-09-2C-48-B6 for an HP machine. Once the gateway's level-2 address is known, its IP address is no longer needed. The originating node packages an IP header containing the IP address of the destination node into a network frame using the level-2 address of the gateway. For example, on a LAN, the destination station address in the IEEE802.3 header would be the gateway's, but the IP address in the IP header would be the remote node's. The link at the gateway will accept the incoming packet because its station address matches. The IP information will be examined by the gateway to see whether the packet is destined for an IP address owned by the gateway. If not, the next hop to the destination will be resolved, and the IP/TCP information forwarded unchanged.

As an example, consider this catenet:



Node1 wants to send a packet to node3. The IP address of node3, IP4, is first determined using Probe Proxy or node1's Network Directory. Node1 examines IP4 and decides that it is not on the LAN or any other network which node1 occupies. Thus, a gateway is necessary. Node1 looks in its internet tables for the LAN (and any other networks attached to node1) to see whether a gateway to the IP4 network has been defined. In this example, node1 will find that node2 is such a gateway and that its address is IP2. Note that IP2 must be on the same network as IP1; IP3 would not be accepted as the gateway's address. Node1 then determines where to send packets so they will reach IP2. This destination will be the station address of node2, S2. Node1 will discover S2 by sending a Probe and/or ARP request to IP2. When IP2 answers, node1 captures the source

address (S2) for the LAN packet. Now, node1 associates S2 with the remote IP address, IP4. Once this association is made, node1 no longer needs to know about IP2 in order to reach IP4 (unless something happens to change the routing).

The packets from node1 will have an IEEE802.3 header with the source address S1 and the destination address S2. The IP header within these packets will have a source address of IP1 and a destination address of IP4. Node2 will accept the packets from the LAN because S2 is its address. But the IP software in node2 will examine the packets (with the 802.3 header removed) and decide that the destination address is not one of node2's. Node2 will then undertake the same resolution process which node1 executed. That is, node2 must answer the questions: "Is IP4 on a network which I occupy? If not, can I find a gateway to reach that network?" In this example, node2 will send the original IP header and imbedded information to node3's level-2 address because node3 is directly-connected.

In summary, the only work which an originating or forwarding node must do is to determine the level-2 address of the next hop. The same IP/TCP information is sent to this address as if the destination were directly-connected.

CREATING PSEUDO-LOCAL NODES

To circumvent the limits on the maximum number of gateways or the maximum number of reachable networks, we need only find a means for specifying the level-2 address without involving the routing algorithms normally used by the Transport. Since these algorithms are invoked only for networks which are not directly-connected, the solution is to make a remote network look as if it is directly-connected when, in fact, it is not. This is accomplished with subnet masks.

The Transport always uses subnet masks to determine whether two addresses belong to the same network. If the Network Manager does not specify these masks, default values are used. For class "A" addresses, the default mask would be 255.000.000.000, for class "B" it is 255.255.000.000, and it is 255.255.255.000 for class "C".

Consider two class "A" addresses such as:

```
A 013 aaa.bbb.ccc  
A 015 xxx.yyy.zzz
```

The default mask would cause these to be treated as different networks because the masking would produce different logical network numbers;

IP address	A 013 aaa.bbb.ccc	A 015 xxx.yyy.zzz
subnet mask	255.000.000.000	255.000.000.000
logical network	013.000.000.000	015.000.000.000

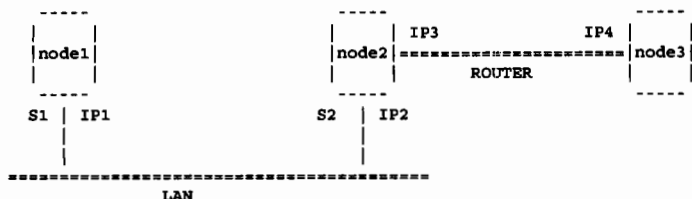
The nodes could not directly communicate, even if they shared the same

hardware LAN. But if a mask of 252.000.000.000 were used, the addresses would suddenly seem to be on the same network, and no routing algorithms would be needed. We shall call such nodes "pseudo-local".

IP address	A 013 aaa.bbb.ccc	A 015 xxx.yyy.zzz
subnet mask	252.000.000.000	252.000.000.000
logical network	012.000.000.000	012.000.000.000

Half of the trick is now in hand. But the level-2 address of the gateway to the pseudo-local node must still be resolved. We will not know the IP address of the gateway because the Transport routing algorithms don't get used. Also, Probe and ARP messages don't cross network boundaries, so those requests would be rejected by a gateway if they are actually destined for a remote node. The answer is to configure the level-2 information in the Network Directory at the source node. That is, configure the level-2 address for the gateway as if it belonged to the remote IP address. The result will be a packet sent to the gateway's level-2 address which contains the IP information for the pseudo-local node.

Consider the previous example again:



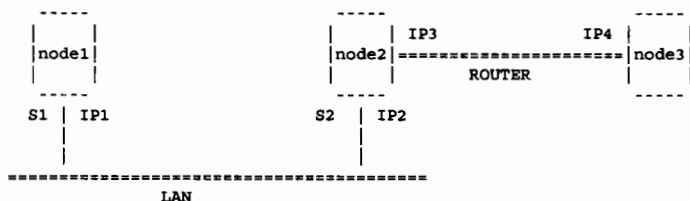
Assume IP1 = 015.001.001.001
 IP2 = 015.001.001.002
 IP3 = 013.001.001.003
 IP4 = 013.001.001.004

The gateway must view these two networks as distinct. So no subnet masks are used at node2. At node1, a mask of 252.000.000.000 is applied. Node3 will look directly-connected. In the Network Directory at node1, IP4 is configured for node3 with S2 specified as "additional address" of type 2 or 4. The search order for node1 must include the Network Directory, but Probe and Probe Proxy may also be enabled. Thus, packets with IP headers containing IP4 as the destination will be sent to S2. Node2 will then forward these packets to node3. Node3 does not necessarily need subnet masks. It will see the inbound packets as destined for its address IP4, and it can reply to IP1 using the normal gateway routing algorithms.

The level-2 address is resolved when the user issues a DSLINE/REMOTE sequence or when an IPC program calls IPCDEST to connect by name. This works just fine

if node1 is the originating node. If node1 is not the originating node but rather it is a gateway along the path to node3, node1 does not know about node3 by name, and no search of the Network Directory will be done. In this case, Network Directory mappings at node1 must be loaded upon a NETCONTROL START. This is specified in the NETXPORT.NI.<ni_name> screen of NMMGR.

As a second example, assume that it is not feasible to choose the network portions of the addresses such that an appropriate subnet mask is possible. With a bit more effort, node portions of the addresses can be used to split the networks. Consider the same configuration, but with both networks having the same class "A" network.



Assume IP1 = 015.001.100.001
 IP2 = 015.001.100.002
 IP3 = 015.001.001.003
 IP4 = 015.001.001.004

At node1 no subnet masks are configured. At node2 and node3, a subnet mask of 255.000.255.000 is used. So node1 thinks that all three IP addresses are on the same network, but node2 and node3 think there are two different networks:

	NODE1	NODE3
IP address	A 015 001.100.001	A 015 001.001.003
subnet mask at node2	255.000.255.000	255.000.255.000
logical network (as seen by node2)	015.000.100.000	015.000.001.000

Now assume that the network directory at node1 has an entry for node3 with station address S2. When node1 resolves the path to IP4, it will think that IP4 is directly-connected and has node2's station address. So packets for IP4 will be sent to node2. Node2 will grab these packets because they come in on S2. But examining the IP address, 015.001.001.004, node2 decides that the packet is destined for the router node. To node2, it looks exactly as if node1 had intended to talk with node3 through a gateway. At node3, the return path is resolved realizing that node2 must act as a gateway.

Now imagine replacing the LAN in this example with an X.25 network. The same trick will work if the address keys in the Network Directory are chosen so

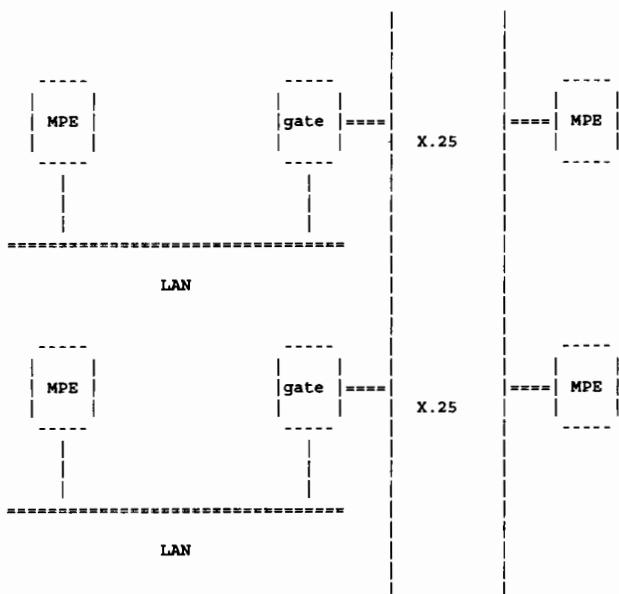
that the SVC or PVC information of the gateway is used to connect to the remote ip address.

DISADVANTAGES

1. ICMP redirect messages have no effect. The Network Manager must take steps to be certain that the local network and the gateway will not be flooded with unnecessary traffic.
2. The Network Manager must know and configure the station address(es) for the gateways in the Network Directory entries which describe the pseudo-local nodes. It is relatively easy to determine these addresses with a tool such as NODESTAT on MPE V or NETTOOL on MPE/iX. However, if the addresses change so, must the Network Directory Entries. Usually, it is possible to have one set of Network Directory entries which can be maintained and distributed to all nodes which must talk to pseudo-locals.
3. The remote networks must be defined such that subnet masks can make them appear to be directly-connected. Retrofitting an existing catenet with such subnets is often difficult or impossible.

CASE STUDY

This is an actual situation in which the user requested about 200 gateways. They originally had several nodes which communicated through a single X.25 network. Since all nodes were on the same network, no gateways were needed. They wanted to increase the number of nodes to about 400, and also change the configuration such that half the nodes have their own LAN that is a unique IP network. The other half would remain on the original X.25 network. In order for a particular X.25 MPE node to communicate with the LAN nodes, gateways seemed necessary. Their initial approach was to place a router between each LAN and the X.25 network and to configure the router as a gateway. Ultimately they would have 200 MPE machines on LANs and 200 gateways to remote MPE nodes.



Each of the X.25 nodes must be able to communicate with each of the LAN networks, but the LAN nodes do not communicate with each other. Looking outward from any particular LAN, there is a single gateway which acts as an interface to a single reachable network. This obviously obeys the limits of the Transport mentioned above. But for the X.25 nodes, 200 gateways and 200 reachable networks are required. This exceeds the legal number of gateways.

The solution was to have all networks be class "B" with subnets configured at the X.25 MPE nodes such that all the remote LANs appeared to be directly-connected. The address keys and SVC information for the gateways were specified in the Network Directory at each X.25 node. Because all the X.25 MPE nodes were on the same network, every Network Directory could contain exactly the same information. The strategy was to maintain a single Network Directory and distribute any future address changes identically to all the X.25 nodes.

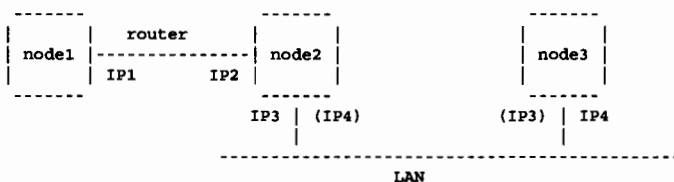
DUPLICATE IP ADDRESS DETECTION ADDED TO NODESTAT

As most Network Managers realize, having the same IP address configured on two different nodes creates connectivity problems which are difficult to isolate.

What might be less obvious, however, is that the duplicate addresses don't need to be on the same physical network nor even active to cause such problems. An MPE node will accept inbound packets (including TCP SYNs) if the destination address in the IP header matches any of the configured address for that node, EVEN IF THOSE ADDRESS ARE ON A NETWORK WHICH IS INACTIVE!

All IP addresses configured with NMMGR become part of an internal Alias List at the time the Transport is started. This alias list is created once and is not affected by which particular network interfaces have been started with a NETCONTROL START;NET=<ni_name> command. It is not an uncommon practice for the configuration file (nsconf.net.sys on MPE V) to have network interfaces are never started. Some users view this as an easy way to manage several nodes with one version of the configuration file existing on all. The idea is that if only certain NIs are started on each node, there is no harm in having several others present and inactive. While this is usually true, networks with gateways can experience connectivity and routing problems if the IP addresses are not managed carefully.

Consider a simple example of a three-node catenet in which a router node (node1) wishes to communicate with a LAN node (node3) using intermediate node2 as a gateway:



Node2 and node3 have duplicate addresses configured, but only IP3 is actually started on node2, and only IP4 is started on node3. If node1 sends a packet to IP4 with the intent of having it forwarded to node3, the packet will actually be accepted by node2. To uncover such corner-case situations, a duplicate address sleuth operating at one of the LAN nodes would need to examine all IP addresses configured for all nodes on that network, not just those addresses which are active. New functionality has been added to the MPE V NODESTAT tool to accomplish this.

At each MPE node, a data construct called a Path Report is created and contains all the configured IP addresses for that node. The Path Report is made available to other nodes through Probe messages on a LAN. As these Path Reports are received, they are stored in the local node's name cache for possible use during future connections. Connection establishment causes the Transport to select only one of the several IP addresses available for the destination node. But when NODESTAT searches for duplicate addresses, it examines all the addresses in all the Path Reports known to the local node,

not just the address actually used in direct connections. Thus, situations such as previously described would be announced by NODESTAT if the local node has been alive long enough to accumulate a significant portion of the LAN Path Reports.

DUPLICATE IP ADDRESS SEARCH

The "G" command has been added to detect duplicate IP addresses and optionally display all IP addresses known to the local node. "G" initiates a search through the name cache and the Network Directory. Only address information which is contained in one of these two areas can be examined. For a LAN, cache entries come primarily from Probe messages, and the number of cache entries will generally increase with time. Therefore, detection of duplicate IP addresses is more likely if NODESTAT is run on a node whose Transport has been active for a long time rather than a node which was recently started with a NETCONTROL START. If a specific remote node is suspected of having a duplicate IP address but it is not in the cache, exit NODESTAT and do a DSLINE/REMOTE to that node. This will place the node's name and address into the cache.

The name cache entries are temporary, because a remote node's information will be removed if it stops gracefully and issues a PROBE NODE DOWN message on the LAN. This means that a duplicate IP address found by NODESTAT could appear and disappear if NODESTAT is run several times. Also, it is possible for NODESTAT to report duplicate IP addresses which are actually multiple instances of the same node. This would occur if the run time for NODESTAT happens to span the time when a node is active, then goes down, then becomes active again. In these transient cases, carefully examine the domain and organization portions of the node names. If the duplicate addresses come from nodes with exactly the same full name and if the duplication goes away when "G" is used again, then the IP address was probably not a real duplicate. (See exception in next paragraph.) But if the node names are different then the duplicates are probably real, even if NODESTAT reports them intermittently.

Note that MPE V supports the "shared NI" capability for X.25 networks. That is, a single node can legally have the same IP address configured twice. In this situation, NODESTAT would detect the duplicate and show two nodes with the same name. The user is given the option of not displaying such duplicates. This option will also eliminate the transient situation mentioned above in which a node goes down and comes back up while NODESTAT is checking the cache.

A cache entry usually contains all addresses configured on a remote node, not just its LAN address. This means that NODESTAT can often detect duplicate addresses on first-hop networks. For example, suppose the local LAN has only class "A" addresses, but some LAN nodes are also attached to an X.25 network using class "C" addresses. Those "C" addresses could be in the name cache even though they are never used on the LAN. NODESTAT will search for duplicates on the "C" network as well as on the directly-attached LAN. Note that this first-hop search is not a feature of NETTOOL on the MPE/iX Transport; NODESTAT will occasionally report duplicate address which NETTOOL will not find.

NODESTAT cannot detect duplicate addresses in the following situations:

1. The remote node(s) use only inbound connections to the 3000 and do not support PROBE. These nodes have no entry in the name cache.
2. The remote node(s) use DDN names. Since only NS names are typically cached using Probe, NODESTAT cannot detect duplicate IP addresses which come from a DDN name server.
3. A remote node has a duplicate IP address, but its addressing information comes from a PROBE PROXY sequence, and the duplicate address is not configured at the PROXY server.
4. A remote node has a duplicate IP address, but its addressing information comes from the local NETWORK DIRECTORY, and the duplicate address is not configured in the NETWORK DIRECTORY. This would include router nodes such as node1 in the example given above. NODESTAT running on node2 would detect the LAN duplicates, but NODESTAT running on node1 could not.

PATH CACHE DELETION

The "P" command has been added to delete entries from the local name cache. There are three primary situations which make this function useful:

1. The remote node's IP address has changed, and PROBE has not been able to detect this change. For example, the node did not send a PROBE NODE DOWN/PROBE UNSOLICITED sequence or the address came from a PROXY server. The old entry in the name cache must be deleted before the local node will select the new IP address.
2. There are multiple paths from the local node to the remote, and the original path can no longer be used. For example, suppose the remote node can be accessed over the local LAN or over a router link. If the LAN network at the remote node goes down after the local node has once used it, connectivity will not shift to the router unless the entry in the name cache is removed.
3. Two or more nodes were incorrectly configured with duplicate IP addresses. Even if the problem is fixed at the remote sites, the local node's name cache could remain corrupted until the cache entries are deleted.

On MPE V, there are two caches of interest for outbound connections. One is the name cache, which stores the IP addresses for a particular node name. The other is the mapping table, which stores the path information (MAC address, etc.) for each IP address. The "P" command will not always solve connectivity problems, because it does not update the mapping table. For example, if an IP address is reassigned from remote node "A" to remote node "B", the mapping table will still contain the MAC address and protocol information (Ethernet, IEEE802.3) for node "A". Therefore, anytime the "P" command is used, the "W" command should also be used. This allows some manual verification of the information for a specific IP address. If the "W" command reveals errors, the

local node's network must be shutdown to clear the problem. Note that the name cache is cleared only when the entire Transport is stopped, but the mapping table is cleared when the associated network is stopped. So a NETCONTROL STOP;NET=lan1 is sufficient to clear mapping table entries for lan1; if a router network is active, it is not necessary to also stop it.

Note that on MPE/iX, the mapping table has an aging algorithm which will automatically remove an entry after a period of non-use.

"W" COMMAND DISPLAY ENHANCEMENT

The "W" command lists entries in the mapping table (not the name cache). For each IP address in the mapping table, NODESTAT tries to find a back link to the node name, so some of the displayed addresses have a name while others do not. The name, if found, could come from the name cache or from the NETWORK DIRECTORY. The "W" command can cause confusion because NETWORK DIRECTORY names are displayed even if a different name (or no name at all) is in the cache. In debugging connectivity problems, the important issue is whether a node has an entry in the cache. This is true because on MPE V, it is impossible to connect to a remote node using its name without an entry in the name cache. The "W" command's display has been enhanced to indicate whether there is a name cache entry for each IP address.

A System Manager's Look at MPE/iX 5.0

by

Kevin Cooper

Hewlett-Packard Company
Commercial Systems Division
19055 Pruneridge Avenue, #46TU
Cupertino, California, 95014
(408) 447-4004

Presented at

The 1994 INTEREX Conference
Denver, Colorado
September 18-22, 1994

Paper #8008

Introduction

Much has been written over the last few years about the challenges of adding POSIX support to MPE/iX in an upwardly-compatible way. For most HP 3000 system managers, this has remained theoretical up until now, because their systems are still running pre-POSIX releases such as 4.0. Now that release 5.0 will soon be shipped to the entire installed base, everyone who manages an MPE/iX system needs to understand the issues associated with updating their systems to an MPE/iX release which supports POSIX.

This presentation is a follow-on to the paper I presented last year, entitled "A Programmer Looks at MPE/iX." That paper described the first round of POSIX-related changes introduced in MPE/iX release 4.5. This year I will revisit some of those same issues and provide an update on how HP has enhanced many of the POSIX/iX features in release 5.0. Special emphasis will be given to the ways MPE/iX 5.0 resolves some HP 3000 system managers' earlier concerns about the POSIX/iX implementation in release 4.5.

Almost all of the features described in this paper are included in the first 5.0 release, called the "Limited Release" or "5.0 Pull". MPE/iX support customers can order this release today. A few of the new features (as noted later) will only be available in the second 5.0 release, known as the "General Release" or "5.0 Push". This release will be shipped to all MPE/iX support customers as the next platform release, with the first shipments planned for early 1995.

1. Key Enhancements to MPE XL

Before we look at the new POSIX/iX features, let's explore some of the most interesting non-POSIX enhancements made in MPE/iX 5.0.

:SETCLOCK

For the past several years, HP has been working hard to increase system availability on HP 3000 systems. One cause for downtime has been the need to restart the system twice a year for the time changes associated with daylight savings time. This downtime is eliminated as of release 5.0, with the new :SETCLOCK command.

There are several options to this command, but the easiest use for the upcoming change from Daylight Savings to standard time is:

```
:SETCLOCK TIMEZONE=W7:00
```

The characters "W7:00" in this command indicate the new time zone is seven hours west of GMT. This happens to be the time zone you would use this fall if your system was located here in Colorado (on Mountain Time), where we are attending this conference.

In the spring, when we return to Daylight Savings Time, system clocks in Colorado would be set ahead by this command:

```
:SETCLOCK TIMEZONE=W6:00
```

A change which moves the clock ahead will generally take place immediately. A change which sets the clock back may be phased in over a period of time, to avoid confusing processes which are dependent on sequenced timestamps. However, any change can be made to occur instantly if needed, by specifying the ;NOW option.

For further information on this and the new :SHOWCLOCK command, see page 3-35 of the 5.0 Communicator or the 5.0 on-line Help.

Wildcard :PURGE

Other operating systems (even MPE V) have allowed users to delete a whole set of files with a single command, but MPE/iX has just introduced this capability on release 5.0. By default, the system will ask you to confirm that you really want to purge all the files that match before it deletes them from the system.

```
:PURGE F@
```

```
19 FILEs matched
```

```
Continue PURGE ? (YES/NO)Y
```

```
19 selected. 19 succeeded. 0 failed.
```

Some of the options available on this new command include ";NOCONFIRM", which over-rides the prompt asking you whether to continue, or ";CONFIRMALL", which asks you to confirm each file before it is purged. You can see the name of each file which is

purged via the ";SHOW" option. You can also specify what to do if an error occurs during the :PURGE (default is to keep going) and the level of error messages you wish to see. Several other options allow you to specify how to handle files with lockwords. For further details, see page 3-57 of the 5.0 Communicator.

:SETCOUNTER

This new command will enable you to reset the counters for spool files, job numbers and session numbers without having to restart the system. So, if you wanted to limit the length of a job number to four digits, then restart counting at 1, you would enter:

```
:SETCOUNTER JOBNUM;BASE=1;MAX=9999
```

When the named counter reaches the maximum value specified, it resets to the specified base value and then uses the next available number without creating duplicates. So, if #J1 was still logged on, the next job would be #J2. This command can be placed in the SYSSTART file so your system will use a consistent range of numbers after each reboot. It is planned to be part of the 5.0 General Release early next year.

VOLUTIL CONTIGVOL

Also on the 5.0 General Release, a new CONTIGVOL command is planned for VOLUTIL. This will provide a way to condense disk space on a disk volume, mainly to gather all the contiguous space required for an MPE/iX operating system update. Of course, this won't help you update to 5.0, because you need to be on 5.0 to use it. But at least it will be there for future updates.

2. POSIX/iX Enhancements

The many new POSIX/iX features have been integrated into MPE/iX at low levels of the operating system for maximum performance and reliability. In addition, to integrate the POSIX/iX features into MPE/iX, many enhancements have been made to the familiar MPE XL functionality. Once you have updated to 5.0, there is an on-line tutorial available which can help explore many of these subjects. It is accessed by executing the command file POSIXCBT.LSN.SYS.

The POSIX.2 Shell and Utilities

The new POSIX.2 shell and utilities represent a significant addition to the MPE/iX Fundamental Operating System (FOS). The shell is an entirely new command interpreter on MPE/iX, which runs under the familiar CI. Its main purpose is to support the POSIX/iX application development and run-time environments. Users may choose either the CI, the shell, or some combination of both to accomplish various tasks on the system.

This paper cannot possibly discuss all the new capabilities of the shell software. For more information, refer to the HP two-volume manual set entitled "MPE/iX Shell and Utilities Reference Manual", which describes all the new shell functionality. You may also want to review the "MPE/iX Shell and Utilities User's Guide", which gives a more detailed look at many of the new utilities.

Hierarchical File System (HFS)

MPE files have followed the same naming convention since the beginning of MPE. File names contain 1-8 characters; the first must be a letter, while the others may be letters or digits. Letters are always shifted to upper case. File names may be qualified by group and account locations in the directory structure, with group and account names following the same naming convention as file names. In addition, there are three domains where files can be stored, called permanent, temporary, and new. All of this continues to work exactly the same way on MPE/iX 5.0.

The HFS on 5.0 adds multi-level directories to all three file domains. MPE account and group can be thought of as two levels in the directory structure - in fact, that is essentially how they are implemented in the new scheme. There is a new top layer called the root directory, specified by a slash ("/"). Users have the ability to add as many layers as they like below the root, an MPE account or an MPE group.

The HFS also adds a new file naming syntax called HFS syntax, used to represent HFS file and directory names. This new syntax is the default when you are running in the POSIX/iX environment, using either the POSIX.2 shell or a program linked with the POSIX.1 interfaces. The rest of MPE/iX (except for a few utilities which have not been modified) uses a new hybrid called "MPE-escaped" syntax, which defaults to the familiar MPE file name, unless the first character of the file name is a dot (".") or a slash ("/").

Here are some examples of how the new MPE-escaped syntax works:

```
:PRINT myfile
  prints the contents of MPE-named file "MYFILE" (upshifted)
:PRINT ./myfile
  prints the contents of HFS-named file "myfile" (lower case)
```

When you use HFS syntax (and only then), upper and lower case letters represent unique characters, so "MYFILE", "MyFile", and "myfile" name three different files. HFS names may also include three special characters: dot (.), hyphen (-), and underscore (_). An HFS name may even begin with a dot, underscore or digit. HFS names can be up to 255 characters in length, except for those residing directly under root, an MPE account, or an MPE group, which are limited to 16 characters.

There are two ways to reference an HFS file name: absolute and relative. Absolute file names start with the root directory and list every directory traversed to reach the file. Thus absolute file names always begin with a slash. For example, the HFS file name `"/bin/sh"` uses an absolute file name to describe the file `"sh"` in the `"bin"` directory under the root directory.

Relative file names use the new concept of the "current working directory" (CWD), which is introduced with the HFS. While this has some similarities to the familiar MPE concept of a logon group, there are also some important differences, which will be discussed later. You should think of your CWD as a short-hand for specifying file names, and no more. Relative file names use the CWD as a base and traverse directories from there. In HFS syntax, `"myfile"` without a leading slash refers to `"CWD/myfile"`. MPE-escaped syntax treats unqualified MPE file names as relative to the CWD rather than the logon group, so `"MYFILE"` refers to `"CWD/MYFILE"`. This will not impact users unless they explicitly execute a command to change their CWD.

Two special cases of relative file names are the dot (`.`), which specifies the CWD, and the dot-dot (`..`), which represents the parent directory one level above the CWD. The meaning of the dot character in file or directory names depends on its context:

1. A separator between the MPE file, group, and account names.
2. A valid character in an HFS file or directory name.
3. The current working directory (when it appears alone).
4. The parent directory (when two dots appear together alone).

Since MPE groups and accounts are special types of directories, all permanent MPE files can also be referenced with HFS syntax by specifying the name in the format `"/ACCOUNT/GROUP/FILE"` (upper case is mandatory). For example, `"/SYS/PUB/EDITOR"` is the HFS file name for the MPE-named file `EDITOR.PUB.SYS`. However, the converse is not true; HFS files outside MPE groups cannot be accessed using MPE file names. The file `"/bin/sh"` can only be referenced using its HFS name.

Restrictions with HFS Directories

With release 5.0, there are still several limitations on the types of files that can be located outside MPE groups. The most notable are privileged files (PRIV), including all data base files, and all Compatibility Mode (CM) file types (PROG, SL, KSAM, RIO, CIR). User-defined command (UDC) files, system and user log files, system configuration files, and files used by the mirrored disk and SPU switchover software are also limited to MPE groups. However, Native Mode KSAM files may reside outside MPE groups, as may message files, which are now handled in Native Mode. An HFS directory may only reside on a user (private) volume set if it is under an MPE group which is assigned to that volume set.

Native mode program files (NMPRG) and Executable Libraries (XL) may reside in HFS directories as of release 5.0. However, since directories do not have special capabilities associated with them like MPE groups do, any NMPRG file which was linked with the capabilities PM (Privileged Mode), MR (Multiple Resources) or DS (Extra Data Segments) will not run. An NMPRG file which was linked with PH (Process Handling) capability will execute from a directory only if the user running it has that capability.

Certain MPE/iX commands, such as :FCOPY, :STREAM, :XEQ (implied :RUN), and :WELCOME, still require MPE-only syntax for the file names given to them as parameters. One way to work around this is to use :FILE equations to refer to HFS file names, since :FILE equations allow HFS names on the right side of the equation. For example, to execute a command file with a name that can only be specified using HFS syntax, such as a file in a directory, try:

```
:FILE mycmd=./dir1/command_file
:XEQ *mycmd
```

DSCOPY is one command which does not work with this method. If a file specified resides in an HFS directory, it causes an error.

There is no way in the 5.0 CI to insert a directory into your path of locations to search for executable files, since the HPPATH variable has not been modified to accept HFS directories. If you use the shell, this is not a problem, because its PATH variable (which works like HPPATH) accesses files using HFS syntax.

:LISTFILE and the HFS

On release 5.0, the :LISTFILE command has been modified to handle MPE-escaped syntax, but :LISTF has not. Both commands, however, now list by default the files in the CWD rather than in the logon group. A new UDC named :PLISTF has been added on 5.0 as part of an HP-supplied UDC file called HPPXUDC.PUB.SYS, to simulate :LISTF functionality for the HFS by calling :LISTFILE.

The wild card character "@" used in :LISTFILE parameters generally still means all files. But the new POSIX/iX features have added some variations to the meaning, so it now depends on the context of the "@". Here are some examples of how it can be used:

```
:LISTFILE @ {lists all files in the CWD with valid MPE names}
FILENAME
```

```
MYFILE      YOURFILE
```

```
:LISTFILE ./@
      {lists all files in the CWD with valid HFS names}
      {note that upper case names sort before lower case names}
PATH= /COOPER/PUB/
```

```
MYFILE      YOURFILE      long_file_name      mydir/
```

```
:LISTFILE ./@;TREE {or ":LISTFILE ./", which implies ;TREE}
{recursively traverses directories under the CWD}
PATH= /COOPER/PUB/

MYFILE          YOURFILE          long_file_name    mydir/

PATH= /COOPER/PUB/mydir/

FILE2 file1
```

The "@" in a file set using MPE syntax means all MPE-named files. When it is used in MPE-escaped syntax, the "@" means all objects, including MPE-named files, HFS files and HFS directories.

Byte Stream Files

Byte stream files were introduced on release 4.5, and are similar to variable length record files without the concept of a record. Each file is simply a continuous stream of bytes, using the newline character (linefeed, ASCII 10, "\n" in C) to separate data. Byte stream files can be identified in a ":LISTFILE ,1" (SUMMARY) output by a size of "1B" (one byte) and a record type of "BA" (Byte Stream, ASCII) where you often see "FA" (Fixed, ASCII). To create and edit byte stream files, use the shell's "vi" editor.

Byte stream files work in both the MPE and HFS environments. The MPE/iX file system treats them as variable length record files, so most CI commands and utilities which operate on variable length files will also work with byte stream files. This includes commands like :PRINT and :COPY, but HP's editors (EDITOR, TDP, HPEDIT, HPSLATE) have not been updated to handle these files.

New Special File Types

MPE/iX release 5.0 supports several new special file types for the POSIX/iX environment. These are symbolic links, device links, FIFOs (First-In First-Out), and streams files. For more detail, refer to the 5.0 Communicator articles on pages 5-71 and 5-74.

Symbolic links can be thought of as permanent file equations. One file is simply a pointer to another file. For example, the program files of the POSIX shell actually reside in the group HPBIN.SYS, but have links which point to them in the directory "/bin". When you run the link, you actually run the file it points to. The CI commands to create and remove symbolic links will be discussed below, under "New CI Commands".

Device links work essentially the same way. You create a file, such as "DEVTAPE", and link it to an LDEV on your system, such as the tape drive LDEV 7. When you then read from or write to "DEVTAPE", you will actually access the tape file on LDEV 7.

FIFOs, or named pipes, are a POSIX standard for inter-process communication, and are conceptually like MPE/iX message files. They normally have one process writing to them and another process reading from them. A streams file is an interface to the streams facility, used in developing networking software.

Device links, FIFOs and streams files are created with the new program MKNOD.PUB.SYS. To create a device file "DEVTAPE" and link it to LDEV 7, you enter:

```
:MKNOD "DEVTAPE c 0 7" {the quotes are required}
```

where the lower case "c" indicates character mode, zero is the major number and means create a device link, and "7" is the LDEV. Entering a major number greater than zero creates a streams file.

To create a FIFO named "MYFIFO", you specify a lower case "p":

```
:MKNOD "MYFIFO p"
```

which indicates that the file is to be a FIFO, or Named Pipe.

Special files appear in ":LISTFILE ,1" (SUMMARY) output like this:

FILENAME	CODE	-----LOGICAL RECORD-----	SIZE	TYP	EOF	LIMIT
ASYMLINK			1B	BAL	13	1024
DEVTAPE			128W	BBd	0	1
MYFIFO			128W	BBf	0	1
STREAM			128W	BBS	0	1

The record type field (TYP) indicates each file is Byte stream ("B"), then ASCII or Binary ("A" or "B"), and then a special file of type Link, Device, FIFO, or Stream ("L", "d", "f", or "s").

You can use ":LISTFILE ,7" (UNIQUE) or ":LISTFILE ,5" (DATA) to see the unique information about special files:

```
*****
```

```
FILE: ASYMLINK.PUB.COOPER
```

```
SYMLINK TARGET: /SYS/HPBIN/SH
```

```
*****
```

```
FILE: DEVTAPE.PUB.COOPER
```

```
DEV TYPE : DEVICE LINK
```

```
LDEV : 7 IO CLASS : TAPE
```

```
*****
```

```
FILE: MYFIFO.PUB.COOPER
```

```
DEV TYPE : FIFO
```

```
*****
```

```
FILE: STREAM.PUB.COOPER
```

```
DEV TYPE : STREAMS
```

```
MAJOR # : 1 MINOR # : 7
```

```
LINK NAME :
```

New CI Commands

To manage HFS directories and symbolic links, six new MPE/iX CI commands and one new system variable were introduced in releases 4.5 or 5.0. Each of these has a similar command in the .2 shell, which may be familiar to users of other operating systems. While HFS directories will mainly be managed in the POSIX/iX environment by using shell commands, corresponding CI commands have been provided to make the system manager's job easier.

The new commands, and their related shell commands, are:

:NEWDIR	mkdir	to create a new directory
:CHDIR	cd	to change your Current Working Directory
:PURGEDIR	rmdir	to remove a directory
:NEWLINK	ln -s	to create a symbolic link
:PURGELINK	unlink	to remove a symbolic link
:DISKUSE	du	to show disk space used in HFS directories

The first parameter to all of these new commands is the directory or link name, but the CI and shell use different naming syntaxes. For compatibility with other MPE/iX commands, the CI expects the name to be in MPE-escaped syntax, and will upshift it unless it begins with a dot or a slash. The shell expects the name to be in HFS syntax, and does not upshift it. So ":NEWDIR ./dir1" and "mkdir dir1" both create a directory "dir1", but ":NEWDIR dir1" upshifts the directory name to "DIR1".

Creating an HFS directory with :NEWDIR or "mkdir" is similar in many ways to creating an MPE group. The new directory becomes a repository for files (and other directories). Another way to create directories is with the new ";CREATE=PATH" option on the :RESTORE command, if the user has the appropriate privileges.

Switching your CWD with :CHDIR or the "cd" shell command is quite different from changing MPE groups. Your file access privileges do not change like they do when you execute the :CHGROUP command. :CHDIR never requires a password, since it does not give the user access to any new files or capabilities. As stated earlier, the CWD should be thought of simply as a short-hand way for naming files. Changing it only impacts which directory you examine when you specify a relative file name in any file naming syntax.

The new system variable is HPCWD, which stores the value of your CWD. It is like the shell environment variable PWD. It is set by the operating system whenever you change directories, but cannot be set directly by the user since it is a read-only variable.

Directories are removed with the :PURGEDIR command, which is similar to the shell "rmdir" command. :PURGEDIR can be used to recursively remove directories by specifying the ;TREE option or by ending an HFS-syntax directory name with a slash. Any files in

the directories affected will be purged as well, the same as when you purge an MPE group. You have the chance to confirm the purge before it takes place, unless you specify the ;NOCONFIRM option or execute the :PURGEDIR from a job.

The :NEWLINK command creates a symbolic link, and has two required parameters: the link file name, and the file being linked to. The command ":NEWLINK ftp, ftp.arpa.sys" creates a link file named "FTP" in the CWD, which would point to the file "FTP.ARPA.SYS". From then on, whenever you refer to "FTP", you will access the file in ARPA.SYS, even if you logoff and log back on again.

:PURGELINK is used to remove a symbolic link from the system directory. It works conceptually like a :PURGE of the link file name, but there is an important difference between using :PURGE and :PURGELINK. If you just use :PURGE with a symbolic link file name, it will try to :PURGE the file which the link points to, and not the link itself. This works like a :PURGE command which back-references a :FILE equation (using a "*"), which removes the file pointed to by the :FILE equation.

:DISKUSE will be described later, under "Disk Space Management."

CI Error Handling

To accommodate the HFS, some CI error handling has been changed. Error numbers have not changed for existing errors, but new errors have been added where required to handle new situations. Let's look at an example which is fairly typical, a :LISTFILE where there are no files in the group. On release 4.0, users see:

```
:LISTFILE
NON-EXISTENT FILE @.PUB.COOPER. (CIWARN 920)
```

On 5.0, in this same situation, users now see:

```
:LISTFILE
Non-existent MPE named file "@.PUB.COOPER". (CIWARN 920)
```

New messages have been added for some HFS situations, such as:

```
:LISTFILE ./@ {with no files which match}
No match found for the pathname. (CIWARN 9043)
:LISTFILE /SYS/NOGROUP/@ {where the group does not exist}
A component of the pathname does not exist. (CIWARN 9053)
:LISTFILE /SYS/PUB/EDITOR/@
A component of the pathname is not a directory. (CIWARN 9055)
```

You may have noticed that these new CI error messages use both upper and lower case, unlike the old messages, which used only upper case. At the same time these new messages were added, the existing CI error messages were edited to use both cases for readability. The actual messages have changed very little, except where it was necessary for clarification, such as the addition of the words "MPE named" to the :LISTFILE messages above.

3. System Backup

:STORE and :RESTORE

Much has been written over the last couple of years about whether the ":STORE @.@@" command would back up your whole system when the HFS was added. On release 4.5, it did not, because MPE/iX kept the literal meaning of "@" in :STORE and :RESTORE commands, as described earlier for :LISTFILE. On 4.5, when you executed ":STORE @.@@", you only stored files with valid MPE file names which resided in MPE groups. This left out files in directories which were under root and therefore not in an MPE group.

With 5.0, all that has been changed. The "@" is translated into MPE-escaped syntax so that it takes on a broader meaning:

:STORE @.@.	All files on the system
:STORE @.@.[.ACCOUNT]	All files under the account
:STORE @.@.GROUP[.ACCOUNT]	All files under the MPE group
:STORE @	All files under the CWD

If you wish to select only MPE-named files for compatibility with earlier releases, you can specify "?@" as the file set in :STORE.

Obviously :RESTORE versions on MPE XL 4.0 and earlier (and all versions of MPE V) do not understand HFS directories or file names. But you can :RESTORE an HFS file from a 5.0 :STORE tape onto a system running an earlier release of MPE XL. Every 5.0 :STORE tape with HFS files on it contains a special file called HFSMAP, which maps each HFS file on the tape to a unique MPE file name, starting with "F0000000" and increasing sequentially. These HFS files and the HFSMAP file are placed in a special group and account, both of whose names begin with an underscore. This prevents you from directly restoring them onto a 4.0 system, but you may :RESTORE them by using the ;LOCAL option or specifying the ;GROUP= and ;ACCOUNT= options. You can then determine each file's original HFS name by looking at the HFSMAP file.

While byte stream files and other new file types can be restored onto a 4.0 system, they cannot be accessed properly. This is also true for HFS directory structures, which are copied to a 5.0 :STORE tape unless you specify the new ;TRANSPORT=MPEXL option. If you do :RESTORE one of these types of files or a directory onto a 4.0 system, you cannot use :PURGE to get rid of it. You must execute :PURGEGROUP or :PURGEACCT to remove it from the system.

One other subtlety with :STORE and :RESTORE is that a blank space is required before the hyphen when a negative file set appears after an HFS file set (such as "/ -@.PUB.SYS"). This prevents any ambiguity with the hyphen, which could otherwise be treated as part of a valid HFS file name. With MPE file sets, this trailing blank space remains optional.

4. System Security

As you may imagine, the new POSIX/iX features have a definite impact on system security. POSIX security requirements have been integrated into the MPE security system so that both work together. Whichever type of security features you use, MPE/iX protects your system and its data files from unauthorized access. This section explores how some of the new POSIX/iX security functions have been implemented within MPE/iX.

There are several new events related to POSIX/iX security, which can be logged by the system, if enabled by the system manager:

- :CHDIR Logging
- Process Adoption
- File Owner Change
- Directory Open/Close Logging

HP has also introduced a new security product on 5.0, called Security Monitor/iX, which MPE V users have had available for many years. This can enable the logging of two additional events. For more details, see the article on page 3-7 of the 5.0 Communicator.

Integrating POSIX/iX Security

Logon security remains basically the same. Everyone logs on with a USER.ACCOUNT, specifying the MPE group explicitly or allowing it to default to the user's home group. Two new concepts are POSIX User ID and Group ID, which you should be aware of even if you are not using other POSIX/iX features. On release 5.0, every MPE user/account combination has an entry in the POSIX User ID database, and every account has an entry in the POSIX Group ID database. These databases are created automatically when you update your system to release 5.0, and replace the directory user nodes in the MPE system directory. They are backed up with your system directory on a ":STORE ;;DIRECTORY" command, and can be restored to the system if needed with ":RESTORE ;;DIRECTORY".

The system handles user resource accounting with the HFS exactly as it always has. Users are always logged on to a group within an account. All CPU and connect time they consume accumulates to the logon group and account, as before. Changing directories has no impact on this; all resources still accumulate to the logon group.

The changes to file security are much more significant, but mostly affect files outside MPE groups. Security for files in MPE groups is still mainly based on the location of the file in the system directory. To access a file, a user needs permission to access the account, the group, and the file. If the file has a lockword, it must be specified as well. The exception to these requirements is when the file has an Access Control Definition (ACD) associated with it, but this feature has not been widely used.

POSIX/iX bases its file security on the concept of the ACD, so this capability will see much more use on release 5.0 and later. ACDs specify exactly which users may access a file. They can be much more selective than the MPE file access matrix, where the only way to allow even one user outside the file's account to access the file is to allow access to ANY users. To meet the POSIX security standards, all files which are outside MPE groups and all HFS directories (but not MPE accounts and groups) now have ACDs associated with them.

HFS directories use five new security modes in their ACDs, which may be used to restrict access to a directory:

- TD - Traverse Directory entries
- CD - Create Directory entries
- DD - Delete Directory entries
- RD - Read Directory entries
- RACD - Read the ACD for this directory (also used for files)

Here are some ways security checking is enforced for directories:

- :CHDIR needs TD to all HFS directories in the new path
- :BUILD needs CD to the directory where the file will reside
- :PURGE needs DD to the directory where the file resides
- :NEWDIR needs CD to the new directory's parent directory
- :PURGEDIR needs DD to the directory's parent directory
- :LISTFILE needs TD to all HFS directories in the path
and RD to the directory having its objects listed
- :LISTFILE, -2 (ACD) needs RACD to the directory having its ACD listed

Since MPE began, the :LISTF command and its successor, :LISTFILE, have allowed you to list files anywhere on the system. This is still true on release 5.0 for files in MPE groups. But new rules apply to objects in HFS directories, to comply with POSIX security standards. To list the contents of an HFS directory, any user without appropriate privilege (such as SM capability) must have TD access to all directories in the path being searched and RD access to the desired directory. So, to list any files in the directory "/dir1", a user without SM capability must have TD access to the directories root ("/") and "dir1", and RD access to "dir1".

Impact of ACD Format Changes

The internal format of the ACD has changed as of release 4.5. This can cause a problem if you :RESTORE a file with an ACD from a 5.0 :STORE tape onto a system running MPE XL 4.0. The new ;TRANSPORT=MPEXL option on the 5.0 :STORE command avoids this problem by placing ACDs on the tape in the pre-4.5 format. The existing ;TRANSPORT option, which moves files to MPE V systems, does not copy ACDs to tape unless the ;COPYACD option is specified in the :STORE command. In that case, the ACDs are translated to MPE V format ACDs.

If a 5.0 :STORE tape was created without the ";TRANSPORT=MPEXL" option, you can still :RESTORE its files onto a 4.0 system by specifying the ;NOACD option on the :RESTORE. You should reapply the ACD with :ALTSEC after the :RESTORE completes.

If neither the ":STORE ;TRANSPORT=MPEXL" nor the ":RESTORE ;NOACD" option was used, and you :RESTORE a file with a 5.0 ACD onto a 4.0 system, you may see some errors later. Only users with SM or AM (Account Manager) capability, or the file creator, will be able to access such a file. Those users will not notice any problems unless they try to access the ACD for the file with a command like ":LISTFILE, -2" (ACD), when they will see the following message:

```
ERROR ENCOUNTERED WHILE ACCESSING ACD
```

All other users will not be able to access the file at all, and will see error messages such as FSERR 140:

```
ACCESS DENIED DUE TO CORRUPT ACD, FILE OWNER MUST REAPPLY ACD.
```

5. Changes to Long-time MPE Security Restrictions

Several long-time security restrictions in MPE have been changed, to allow POSIX security standards to co-exist with MPE/iX security. It is important to realize that these changes apply whether you are using any POSIX/iX features or not. If you have programs, jobstreams, or command files that depend on some of these restrictions, you should test them before you go into production on release 5.0 or later.

Saving Files Outside the Logon Account

A user with the appropriate capabilities can now save a file across account boundaries. This is provided because of a POSIX requirement, but it is true whether you are using any POSIX/iX features or not. A file in one account might therefore have a creator in another account. Or, from a different perspective, a user from another account might be the creator of a file in your account. The creator field in the file label held only eight bytes for the user name on 4.0, but now has room for 16 bytes, to hold both the user and account names (without the dot separator).

On 4.0, a user logged on as MANAGER.SYS, with SM capability, saw:

```
:BUILD SMFILE.PUB.COOPER
SECURITY VIOLATION (FSERR 93)
BUILD OF FILE SMFILE.PUB.COOPER FAILED. (CIERR 279)
```

With 5.0, this same command will execute without error, and the resulting file will have "MANAGER.SYS" in the creator field. The file will not have an ACD assigned in this situation, but will use the standard MPE file security matrix for PUB.COOPER.

Since you can save across accounts, you can also :RENAME files across accounts. On 4.0, even MANAGER.SYS could not do so:

```
:BUILD SMFILE {in PUB.SYS}
:RENAME SMFILE,SMFILE.PUB.COOPER
SECURITY VIOLATION (FSERR 93)
RENAME FAILED DUE TO FILE SYSTEM ERROR, NOT RENAMED. (CIERR 373)
```

On 5.0, this :RENAME command works for an SM user. After the :RENAME, the file creator will still be MANAGER.SYS. In this case, the file is assigned an ACD during the :RENAME. The new ACD reflects the file security attributes of PUB.SYS, where the file was originally created.

Another way a user can save a file outside the logon account is when the ACD for an HFS directory allows Create Directory (CD) access to ALL users on the system ("@.@"). This is how MPE/iX implements the POSIX specification for granting "write" (w) access to users of class "other". Any user on the system can save a file into such a directory. This only works with HFS directories, not with MPE groups.

One situation where this may come up on any system is the special directory "/tmp", which is provided to implement the concept of temporary files on a POSIX system. This should not be confused with MPE/iX's temporary file domain, which is a totally different concept. By convention, files in /tmp are expected to exist for only a short time, and could be purged at any time. Since the default ACD for the directory /tmp allows CD access to "@.@", any user on the system (even with no special capabilities) can save a file there, by specifying:

```
:BUILD /tmp/kcfile
```

Since the ACD for /tmp also allows Delete Directory (DD) access to "@.@", any user who creates a file there can also purge the file. Other users must have SM capability to do so, since the ACD for the file itself restricts their access to it.

Renaming Files When not the Creator

One of the most requested enhancements to MPE over the years has been to remove the restriction that allowed only the file creator to :RENAME a file. Even if a user could validly copy the file and then purge it, MPE from the beginning has not allowed even the system manager to :RENAME a file without being its creator. Users would see:

```
:RENAME YOURFILE,MYFILE
USER IS NOT CREATOR (FSERR 94)
RENAME FAILED DUE TO FILE SYSTEM ERROR, NOT RENAMED. (CIERR 373)
```

With release 5.0, there are now several situations where you no longer need to be the file creator to :RENAME a file. In every

case, the user must have Save Files (SF) capability, and the file must not have a lockword. A user with SM capability can then :RENAME it to anywhere on the system, even if that user is not the creator. Other users may do so if they have TD access to all the directories involved, DD access to the file's current parent directory, and CD access to the proposed new directory. To maintain system security, an ACD reflecting the original MPE group security is automatically assigned to any file without an ACD, when that file is renamed from an MPE group to a directory that is not an MPE group in the same account.

This is an example of how trying to understand all the security ramifications of mixing the MPE and HFS environments can get quite complicated. But, remember that the purpose of all these security requirements is to protect your system and its data files from unauthorized access. The system accomplishes this goal quite well. As a system manager, it is worth your time to study how the enhanced system security features work, so you can design the best security strategy for the systems you are responsible for.

Managing Files with Lockwords

There are a few changes to watch for involving file lockwords. First, lockwords will only be enforced for MPE-named files which are located in MPE groups and have no ACDs. This is because security checking for a file with an ACD ignores any lockword, and all files outside MPE groups have ACDs associated with them.

Also, lockwords may only be specified when using MPE syntax. Without this restriction, what would QUERY/LOCK represent in HFS syntax? It could be either the file QUERY with a lockword LOCK, or the file LOCK in the directory QUERY.

The other change to lockwords involves an error condition that has been removed. On 4.0, if you specified a lockword for a file that did not have one, it caused an error:

```
:BUILD MYFILE
:PURGE MYFILE/LOCK
LOCKWORD VIOLATION (FSERR 92)
UNABLE TO PURGE FILE MYFILE.PUB.COOPER. (CIERR 384)
```

On 5.0, this no longer causes an error; the lockword is ignored.

Although the restriction that allowed only the file creator to :RENAME a file has been removed, it is still only the creator who can assign a lockword to a file. So, if you are not the creator of the file YOURFILE, you will still see the following error:

```
:RENAME YOURFILE,YOURFILE/LOCK
USER IS NOT CREATOR (FSERR 94)
RENAME failed due to file system error. Not renamed.(CIERR 373)
```

:RELEASE and :SECURE

On release 5.0, it is still true that only the file creator can turn file security checking off and on using :RELEASE and :SECURE. But these commands are only meaningful for MPE-named files without ACDs, since an ACD takes precedence over MPE file matrix security.

6. Disk Space Management

There are two disk space issues you must address before updating to 5.0. One is total disk space, which has increased. Another is LDEV 1 disk space, which has also increased. In fact, HP's older 7933/5 disk drives (404 MB) are no longer large enough to contain all the data that needs to reside on LDEV 1. That could force a full system INSTALL to replace LDEV 1 with a larger drive. For more details, see the Communicator articles on pages 2-1 and 2-6.

One change we must consider is how the system accounts for disk space with the HFS. The disk space limits and disk space usage for directories created under an MPE group are tracked as part of the group, so this presents no problem. But directories can also be created directly under the root directory, and these do not fall under any MPE group. System managers should be aware that disk space can be allocated in these directories without accumulating to the limit of any account or group, and without appearing in the :REPORT command totals.

The way to examine this use of disk space is with the new :DISKUSE command. :DISKUSE can be used to show the disk space in use for all directories directly under root, including all MPE accounts:

:DISKUSE /@

To recursively show all disk space in use by all directories:

:DISKUSE /

This will also list the disk space used by all files directly under the root directory ("/") just before the end of the listing, on the line that displays the count for "(files directly below specified directory)".

To see the total for the whole system without all the detailed line items, you should be able to specify:

:DISKUSE /;NOTREE

However, due to a bug in the 5.0 Limited Release, this command also prints all the detail. It should work as intended by the 5.0 General Release.

Any other directory can replace the root directory ("/") in these commands, to view just the disk usage for that specific directory. In this case, the ;NOTREE option does show you just the total for the requested directory.

The default system security settings limit the spread of disk use outside MPE accounts. Only users with the appropriate privilege (SM capability) may create files or directories under the root directory and all but one of its child directories, since the ACDs for these directories restrict CD access for all other users.

The one exception is the directory "/tmp", where POSIX/iX programs expect any user to be able to create a file. System managers need to establish a way to clean this directory out on a regular basis. One fairly safe way to accomplish this would be with the command:

```
:STORE /tmp/;;DATE<={some recent date};SHOW=OFFLINE;PURGE
```

Another way is to execute the following commands occasionally:

```
:CHDIR /tmp  
:PURGEDIR /tmp/;NOCONFIRM
```

This will remove the files in /tmp, but leave the directory structure in place, similar to what happens when you execute a :PURGEGROUP while logged in to the same MPE group you are purging.

Conclusion

There is no way anyone could possibly cover all the new features of MPE/iX release 5.0 in one presentation. But hopefully you have learned enough about MPE/iX 5.0 to feel comfortable with its new functionality. You should now be able to identify the key issues of updating to release 5.0 and understand how they apply at your site. So, whether you are planning to update to the 5.0 Limited Release now or wait for the General Release later, you should be much better prepared to plan for your update. Soon you will be taking advantage of many of these new features and wondering how you ever got along without them!

About the Author

Kevin Cooper has worked for Hewlett-Packard with the HP 3000 computer system since 1976, in software development, support and technical consulting. He currently assists HP software vendors with POSIX/iX implementations, as part of the Channel Partner Consulting group at Commercial Systems Division. He holds a bachelor's degree in Computer Science from the University of California, Berkeley. He has given several previous Interex presentations over the past eight years.

8009

De-centralizing Printing with the Associate Command
Royden Somerville
Office of University Computing
University of Notre Dame
Notre Dame, Indiana 46556
(219) 631-0581

At the University of Notre Dame, the administrative applications run on a Hewlett-Packard 3000 under MPE/iX. There are several dozen line printers and laser printers installed in the user office areas. (These are all serial printers since, because of distance requirements, they are all connected to the computer through Distributed Terminal Controllers (DTC's)). In addition to having direct access to printers, most users have control of the printers through the application of the Associate command. In what follows, we will look at how such an approach is implemented.

The original situation was that administrative functions were being transferred to the HP 3000 from another computer. It was decided that as much of the printing as possible should be done in the user areas rather than being entirely done in the central computer room.

This policy had several attractions. It was an opportune time to implement such an approach because the systems were going to be revised anyway. It would reduce the printing and forms handling in the computer room by distributing them to remote locations.

The policy would also turn out to have some indirect effects. Delivery effort and delay would be reduced by printing at the points of usage. Misdirection and loss of output would be reduced by minimizing handling of output. Privacy and security would be enhanced by reducing the handling of output. Some forms inventory and ordering could also be distributed.

There were some negative aspects, of course. User reluctance about something new or unfamiliar had to be overcome by training and experience.

First, let us look at what the user sees as a result of the software implementation. (We are here assuming that the hardware (printer, terminal, and DTC) is already installed and configured.)

De-centralizing Printing with the Associate Command
8009- 1

The user logs on with a special ID which is only used for the printer in his or her area. After the password is entered, a log-on, no-break User-Defined Command (UDC) is executed by the operating system. The UDC displays a menu of printer actions on the screen. This menu can be presented by an in-house program, by a third party program, or by a command file. In our case, it is presented by a third party program.

The user selects from the menu whichever choices are desired. The user has the ability: to list jobs; to list print files; to start, suspend, or resume printing; to alter the outfence; to alter output priorities; to delete print files; to turn headers off or on; to reply to requests; and to display printer status. In short, almost everything that the operator can do for that printer. Menu entries use parameters if they are necessary. When finished, the menu is exited, which causes the UDC to log off the session.

You may ask, what enables these things to happen? There are four elements which are required: the user ID, the UDC, the menu mechanism, and the association mechanism.

The user ID must be created. There is nothing special about the ID, but because of the way the Associate facility functions, it helps if the ID is dedicated to this usage.

The UDC file must be created and then linked to the user ID. The UDC file is really what holds the whole application together by executing automatically at log on, setting the environment, running the menu program, and logging the session off.

The menu must be constructed and then linked to the user ID. The purpose of the menu is to offer a simple list of choices to the user, and to hide the complexity of the choice implementation from the user. (In our case, there is a file in addition to the menu file, which is called the profile. This file serves as a "configuration" file by specifying user attributes to the menu program, and by linking the user ID to the menu.)

The printer device class and the user ID must be put into a system table by the Associate (ASOCIATE) program. This is what enables the operating system to allow operator functions to a non-operator user ID. It is a two stage process to establish the table entries.

First the printer class and the user ID must be put in list of similar data for the program, then the program must be run with the data as input.

Let us examine the details of these four implementation steps. Again, the first step is to create the user ID. This is done as the account manager, by executing the Newuser command:

```
:NEWUSER printa;CAP=IA;HOME=agroup;PASS=xyz
```

(where capitalized words are required and lower-case words are names which you supply). The user name and the password may be anything you want them to be--with-in the usual restrictions of length, character type, and uniqueness. The default capabilities are IA and BA; where BA is omitted because batch capability is not necessary, since everything is done online. The home group is whatever you want, provided it allows access to the UDC file (Read and Lock access) and to the menu program (and provided it exists by the time the user logs on). Changes to the user ID may be made with the ALTUSER command.

If you want to put the user into a new group, then use the NEWGROUP command to create the group: NEWGROUP agroup. The default capabilities are IA and BA, and the default access is group user (GU). Characteristics of the group can be changed with the ALTGROUP command.

The second step is to create the UDC file. This step has two parts. The first part is the actual creation of the file. A simple version would look something like this:

```
printudc  
OPTION LOGON, NOLIST, NOBREAK  
CONTINUE  
ASSOCIATE printera  
RUN menuprog  
BYE
```

The first line is the UDC name, which again can be anything you want within the restrictions, except in this case the first twelve characters are used. The second line is the options line, which is not required. Here it is used to declare that the UDC be executed when the user logs on, that the UDC not be listed when it is executed, and that break be disabled while the UDC is executing. The logon option is used to automatically

present the menu to the user, the no-list option is to keep lines off the screen, and the no-break option is to keep the user in the menu program as a security precaution. The CONTINUE allows the ASSOCIATE to cause an error and still have the RUN execute. (More about run-time errors shortly.) The RUN command executes the menu program; and the BYE logs off the session. In the case where this is not the first printer control ID to be created a previous UDC file can be copied and altered.

A fancier version of the UDC would look something like Figure 1. This version has three additional statements: a FILE statement and two IF statements. The FILE statement is optional; it would be for directing specific outputs to the user printer rather than letting them go by default to the LP printer. Other file equations may also be inserted here. The IF statements are intended to take care of two likely run-time errors in a user-friendly way. The first is the case where another user is already logged on with the printer control ID. This causes CIERROR 3021 (ONE OR

```
printudc
OPTION LOGON, NOLIST, NOBREAK
FILE list;DEV=printera
CONTINUE
ASSOCIATE printera
IF CIERROR=3021 THEN
    COMMENT Someone beat you to it.
    ECHO Warning:
    ECHO You may do displays, but you do not
    ECHO have control of the printer.
    ECHO (Press Return to clear.)
    INPUT response
ELSE
    IF CIERROR=3059 THEN
        COMMENT Associate error.
        ECHO There is a problem which prevents
        ECHO your menu from working; please
        ECHO notify your contact person.
        ECHO (Press Return to clear.)
        INPUT response
    ENDIF
ENDIF
RUN menuprog
BYE
```

Figure 1 -- A Better UDC

MORE DEVICES IN THAT DEVICE CLASS ARE ASSOCIATED BY OTHERS). The UDC displays a warning message and pauses until the message is acknowledged. It then executes the menu program. The second IF handles the case where the formal association has not been set up, or has not been set up properly. This condition causes CIERROR 3059 (ASSOCIATING THIS DEVICE REQUIRES SYSTEM MANAGER'S PERMISSION). The IF statement displays a message and pauses until the message is acknowledged. It then executes the menu program. These statements are friendly because they display a customized message which has time to be read, rather than allowing a system error message to appear fleetingly before the screen is cleared by the menu program.

After the UDC file is created, the second part of the UDC implementation is setting, or "turning on," the UDC file. This is accomplished by using the SETCATALOG command:

```
SETCATALOG udcprta;USER=printa
```

This format is the one which would be used by the account manager; it can be executed immediately following the user ID creation without another logon being required.

The third step is creating the menu and its link the user ID. The menu would look something like Figure 2. There are several things which streamline the appearance of the menu, such as keeping the entries to one line apiece if possible, and limiting the menu to one screen. The menu presents almost all the options which are available for manipulating the printer or the print files, plus a status display and some list displays. The intent is to allow the printer operator to function as independently as possible. Hidden behind these entries are the actual commands which enable them to be performed. If the menu file already exists, it may be copied and modified as necessary.

The specifics of this step will vary depending on your implementation. In our case, the menu and its commands go into a file of any name; and the profile is a file whose name is the same as the user name. Within the profile is a line which names the menu file. (As an alternate approach, the menu could be in a command file which would be run by the log-on UDC.)

The fourth step is activating the association. (The ASSOCIATE command, you will remember, is already

Payroll Printer Control Menu

1. Show the jobs
2. Show the print files assigned to the printer
(one line per file)
3. Show the status of the printer
4. Start spooling for the printer
5. Suspend printing on the printer
6. Suspend after the current file finishes printing
7. Resume printing on the printer
8. Resume printing back from where suspended
(you need to enter an approximate number of pgs)
9. Resume printing forward from where suspended
(you need to enter an approximate number of pgs)
10. Alter the output priority on a spool file
(you need a spool file # from the file display)
11. Alter the number of copies of a spool file
(you need a spool file # from the file display)
12. Delete a spool file
(you need a spool file # from the file display)
13. Change the Outfence for the printer
14. Turn Header pages off
15. Turn Header pages on
16. Recall a request
17. Reply to a request
(you need the number from the request display)
18. Show the print files and the Outfence
(two lines per file)
19. Suspend immediately, release file, & reset position
20. Release file w. position maintained (After suspend)
21. Undefer a file

Figure 2 -- Sample Menu

contained inside the log-on UDC for the user; but there has to be a mechanism to provide a validation check in order to prevent the command from being misused.) This step is done with the System Manager ID. This step also has two parts. The first part is to create a data file which is a list of user ID's and the printers to be associated to them. These entries have the form:

```
devclass=userid
```

Some examples would be:

```
PAYROLL1=user.acct  
PAYROLL1=user.@
```

PAYROLL1=@.acct

Notice that the wild-card character can be utilized in the user ID. This could lead to log-on complications, which we will examine in a moment.

The second part of this step is running the ASSOCIATE program. You need a file statement to equate your data file to the formal file designator INFILE:

FILE INFILE=datafile

Then you execute the program with:

RUN ASOCTBL.PUB.SYS

This program execution creates a file called ASSOCIATE.PUB.SYS. The file is a table containing one record for each Ldev on your system. This table structure limits you to one user ID per device class (keeping in mind that the user ID may be wildcarded).

Besides this structural limit, there is a practical limit which only allows one log-on per printer at a time. (This is a reasonable limit because you would only want one person at a time controlling the printer anyway.) What happens is that the first user to log on has the printer associated to the ID, but any other log-on which occurs before the first session logs off will not have the printer associated to it. The second session (or any other after the first) gets an error message when the ASSOCIATE command is executed at log on. (The CONTINUE command is inserted before the ASSOCIATE command to cover this situation by preventing the UDC from aborting at the ASSOCIATE error.) Otherwise, the situation appears normal: the log-on is successful, and the menu is displayed. The difference is that display commands work, but commands which alter the printer or the print files do not work. A "duplicate" session can be handled in several ways: the duplicate session could be logged off automatically by the log-on UDC, or the duplicate session could be left logged on to do displays only.

Security is covered by several factors. The menus limit users to pre-defined tasks. The log-on UDC steers users into the menus, prevents them from breaking out of the menus, and logs them off when they are done. CONTINUE's in the UDC prevent the UDC from aborting. These constructs keep the user either in a menu or logged off, thus preventing the user from doing

anything which is not specified in a menu.

There are additional security features in the menu program, which limits user ID's to particular Udevs, divides users into groupings, and logs off inactive sessions.

The control over a printer which the Associate mechanism provides enables users to: print special forms, filter output, reset jams, and reset time-outs.

Deferred output can be printed on special forms by raising the Outfence, taking the Headers off, changing forms, altering the file's output priority, printing the file, changing forms back, putting the Headers on, and lowering the Outfence.

Output can be filtered by keeping the Outfence above the default of 7, printing what is needed, and deleting the rest.

Jams can be handled by clearing the jam, resetting top of form, realigning the form, and putting the printer on-line. If more than one page of output was damaged, then printing can be suspended, moved back, and resumed.

Time-outs (spooling stopped) from jam, offline, or paper out conditions can be handled by correcting the original problem and then starting spooling.

Support from Operations is vital, especially in the initial phase. Often this can be handled by telephone. Simple problems can be handled by advising which menu selections to choose. Some problems may require an on-site visit to the printer, but these occurrences decrease as the users gain experience. Another form of support is file transfers. Large print-outs can be transferred by the operator from the remote printer to the central printer (which is usually faster --thus saving print time). Or, if the remote printer is down, files can be transferred by the operator from the remote printer to another remote printer or to the central printer.

Trouble-shooting the menu operation may be necessary occasionally. There are only a few main cases. If the menu does not appear, then either the UDC file does not exist, or it is not activated. This can be remedied by creating (or restoring) the UDC file if it does not exist, or by using the SETCATALOG command to

activate it if it does exist.

If the menu appears, but alter commands have no effect, then either there is another user logged on previously, or the printer Ldev has changed. If there is another user who logged on previous to the user with the problem, then the second user (the one with the problem must wait until the first user logs off so that the second user can log on again and obtain exclusive access to the printer, which will give the second user control of the printer. If the Ldev of the printer has changed, then the menu needs to be corrected. Also, the ASSOCIATE input data needs to be changed and the ASSOCIATE program re-run in order to rebuild the table with the correct Ldev in it.

The results of this whole printing conversion effort are easy to see. There are over thirty printers at Notre Dame which are associated with a special user ID. Some are little used and some are used every day. There is much printing which is done on the remote printers, and a corresponding drop in the amount printed in the computer room. Output which is printed remotely does not have to be delivered and does not have to have forms stocked in the computer room. The Payroll department in particular does not store paychecks at the central computer site any longer, but stores them at their department location. This is a more secure arrangement from the Payroll department's viewpoint because they keep the checks in their own area retain full control over them, rather than having to entrust them to someone else.

There are also some less obvious benefits. Output printed remotely incurs no delivery delay, and thus is available to the user in a more timely fashion. Reduced handling lowers the chances of loss or misdirection. There is increased privacy for the output because handling is reduced, since only the user department is involved in routing the output. The users have more control over the printing process since they can do the printing themselves and do not have to depend on someone else to do it. The users also acquire a better understanding of what is involved in printing because they experience it first hand.

Thus there are many benefits and few disadvantages to decentralizing printing with the ASSOCIATE command.

MPE/iX DEBUG & DUMP ANALYSIS

by

Michael Hornsby
Beechglen Development Inc.
5576 Glenway Avenue
Cincinnati, Ohio 45238
(513) 922-0509

Its 9:00, a Monday morning, the last day of the month, a concurrence of the busiest day of the week with the busiest day of the month. At once, all the phones in the department start to ring. You can see the turmoil in the machine room through the window, you instinctively hit the enter key, no colon prompt.

Scenario #1: You feel lucky, so you decide to restart right away, hoping that it was a one time fluke. It took only 40 minutes to get completely back up, a dump would have cost at least another hour. You start humming "You got to know when to hold them, know when to fold them ..."

Scenario #2: You place a support call, some time later they call back, they want you do a dump and restart. It is now 1 PM, you're thinking "Well, at least we only lost half a day, maybe they'll find something in the dump". At 2 PM, a support person dials up, spends about an hour signed on (users complained about poor response). At 3:30 you get a call back, something about the version of this and the version of that don't match, the bottom line is that you need the latest power patch. You start humming "Patches, I'm depending on you son ..."

Scenario #3: This is the fourth system abort in three days, You've reinstalled twice, and have replaced a disc drive and a SCSI controller. The meeting with the VP late yesterday was a real barn burner. You managed to get a couple of hours sleep, on a cot made of those manual updates conveniently stacked up in the corner. You had a strange dream. The system fell from the roof on the VP, all the users were very short, they tell you that only the wizard can help you to get home again. You start humming "Somewhere over the rainbow ..."

No matter what the platform HP, IBM, DEC, even PC servers, the system manager/administrator must make decisions that places him/her in the middle. The hardware/software support resources want memory dumps, traces, patches, etc., and the users simply want the system up ASAP. It's these times that you wish that you knew more about how things really work.

The purpose of this article is to provide MPE/iX system managers and system programmers with practical examples of how to explore a live system using DEBUG, and how to initially interpret a system memory dump.

The system DEBUG facility will allow a user with SM capability to view and MODIFY memory contents, therefore, before I continue, the following set of warnings and disclaimers must be stated. 1) The DEBUG facility bypasses the normal security and data protection mechanisms built into the operating system, so don't experiment on production systems and files. 2) Hewlett-Packard currently provides these macros as part of the operating system but at any time may choose to alter, password, or remove them. 3) The sole purpose of this material is for the information of the readers. Any implied use, warranty or support is expressly denied.

On each MPE/iX update a new set of DEBUG macros and symbol files are loaded into various groups in the TELESUP account. The macros are small debug command scripts, and the symbol files are data definition templates for operating system structures. There are two methods for using these macros and symbols; one is on a live system using DEBUG, and the other is on a DUMP file using DATPROG. The remainder of this article is devoted to examples on various system problem situations and how to use some of the more interesting and useful macros.

The key to reading the following debug and dump narration is that text appearing in lower case is a command that you enter. Text that is enclosed with leading greater than and less than signs is a comment. << THIS IS A COMMENT >>. And, three dots '...' indicates output that was displayed but not included to preserve brevity.

GETTING STARTED IN DEBUG:

```
:hello mgr.telesup,dat
...
:debug use datinit;macstart,"1" << THIS STARTS DEBUG AND LOADS THE OS MACROS >>
DEBUG/iX B.30.45
...

$10e ($33) nmdebug > mac1 << THIS COMMAND ALSO ACCEPTS WILD CARD CHARACTERS >>
macro add_pin_to_list          : STR/STR = ''
macro aif_detail1
...
macro xm_ui
macro xm_user_log
macro xm_val_lr                : INT/U16 = $0

$119 ($33) nmdebug > help config_console << THE HELP COMMAND WILL DISPLAY THE
MACRO >>
"config_console" is a MACRO.

macro config_console
  machelp = 'Prints the logical device number for the system.'
  mackey = 'HP Q_CQ_H HL CONFIGURATION CONSOLE DEVICE'
  macver = 'A.00.02'
{ loc save_error_action      : str = error_action;
  var error_action           = 'ps';
  loc ldev = [abs %1074];
  wl "SYSTEM CONSOLE AT LDEV ", ldev:'#';
  var error_action = save_error_action;
}
```

The full list of macros and symbolics would be too long to include in this article. Besides, a cross-reference is really required to make practical use of the list. I will submit a set of DEBUG listings, cross-references and sample programs to the Denver CSL Swap tape.

REGISTER ADDRESSING:

On MPE/iX systems the basic building block is the capacity for addressing into large virtual spaces. A virtual space consists of a space identifier and an offset into that space. Both are 32 bit values which would allow for over 4 billion spaces, each over 4 billion bytes long. The systems implemented to date only use half of the space identifier word, allowing for 65,636 spaces.

The basic process of debugging the system is finding a given virtual address pointer and translating the data at that location using a symbolic template. Before we continue, a brief discussion about the registers and virtual addressing is required.

At the hardware level MPE/iX defines 8 space, and 31 general registers. Two different methods of addressing are employed, depending on the desired maximum addressable range. Long pointer addressing requires that a space and a general register be loaded prior to a code or data memory reference. Long pointers use space registers 1 through 3.

Early in the development of MPE/iX, HP decided that loading two registers to address a virtual space of 4 billion bytes was overkill for most applications. So they developed a convention that would allow loading a single register value to act as a virtual pointer into a smaller one billion byte space. This is called short pointer addressing.

In short pointer addressing, space register 0 is loaded with a special form of the offset. The value of the high order two bits of the offset address is added to 4 to identify the space register 4 through 7 to be used. Space registers 4 through 7 are pre-loaded with commonly used space identifiers for the current process. Space register 4 is the space of the process' code, and register 5 is the space of the process' data. Can you guess what the other two are for?

The spaces and offsets referenced in short pointer addressing are called 'QUADS'. The name is derived from the fact that the use of the two high order bits as a space register semaphore also causes the pointer to range in one quarter of the 4 billion bytes of addressable space. Thus, a binary value of 00 (hex 0 - 3) in the high order bits means that the address is in QUAD #1 of space register 4, and a value of 01 (hex 4 - 7) in the high order bits means the address is in QUAD #2 of space register 5, and so on.

In DEBUG we can display the system registers with the display register (DR) command:

```
run discfree.pub.sys;debug
```

```
DEBUG/iX B.30.45
```

```
DEBUG Intrinsic at: 640.0000fe98 ?PROGRAM
$(338) nmdebug > dr
R0 =00000000 009b624c 0090b248 00000001 R4 =81c25200 849a2000 00000000 00000000
R8 =00000000 00000000 00000000 00000000 R12=00000000 00000000 00000000 00000000
R16=00000000 00000000 00000000 c0000000 R20=00001000 0000000a 81c25200 00000001
R24=40331008 0000fe9a 00000640 40331008 R28=0000000f 00000000 403330d0 00000002

IPSW=0006000f=jthlnxbCVmrQPD1 PRIV=2 SAR=0000 PCQF=640.fe9a 640.fe9e

SR0=0000000a 00000000 00000000 00000000 SR4=00000640 000001cc 0000000b 0000000a
TR0=007fe200 0083e200 000001cc 008182a0 TR4=849b0000 403333fc 40333238 849a2018
PID1=0682=0341(W) PID2=0148=00a4(W) PID3=0000=0000(W) PID4=0000=0000(W)

RCTR=00000000 ISR=00000640 ICR=00000000 IIR=0000400e IVA=00125000 ITMR=14ff4a3f
EIEM=fffffff EIRR=00000000 CCR=0080
```

Translating the registers: SR0's high order bits are 00 so this pointer is an offset into the space identifier in SR4. SR4 contains a \$640, this is the space identifier for the code that is executing. SR5 contains \$1CC, the space identifier for the data space of this process. R30 is the current stack pointer, notice it points to SR5.

In either addressing method a full 64 bit address is generated. The optimization of short pointers comes from having the four frequently used space identifiers preloaded.

SYSTEM TABLES AND KNOWN OBJECTS:

Every operating system has a set of tables that keep track of system resources and processes. On MPE/iX the tables are given their own space identifier and name. The tables form a hierarchy starting at the Known System Objects (KSO) table. This table has the space identifier of all of the other system tables. Exploring and understanding these system tables or objects is the best method to learn how the operating system functions.

The technique for mapping the KSO's to their virtual address comes in three parts: first we must know the base address of the KSO table. This is a fixed value and is set up for us in the macstart macro. The variable `hp_system_globals_ptr` is set to `$a.c0000000`. Second we will use the symbolic `'SYSTEM_GLOBALS_TYPE'` to format data starting at this address. Third, we will use the SYML command to map the name of the object to the pointer provided in step two.

```
$127 ($3b) nmdebug > fv hp__system_globals_ptr,'SYSTEM_GLOBALS_TYPE'
```

```
RECORD
KSO_TABLE
[ 0 ]: VA_TYPE( a.c0000000 )
[ 1 ]: VA_TYPE( 0.7fe200 )
[ 2 ]: VA_TYPE( 0.83e200 )
[ 3 ]: VA_TYPE( a.c0208800 )
[ 4 ]: VA_TYPE( a.c0208000 )
...
```

```
$129 ($3b) nmdebug > syml kso_a
KSO_ADOPT_SEMAPHORE      CONST  INTEGER    $a7
...
KSO_DISP_GLOBALS        CONST  INTEGER    $7f
...
KSO_PDIR                 CONST  INTEGER    $1
KSO_PDIRX                CONST  INTEGER    $3
KSO_PDIR_HEADER          CONST  INTEGER    $4
...
```

The symbolic constant gives the object number in the KSO table which has the virtual address of the desired object. Thus the KSO_PDIR is at virtual address 0.7fe200. The next question that occurs is, "How does one tell what all of these KSO's do?" The answer is that some of them have names that are self describing, others have cryptic names. The best method is to work with the macros that are of interest and decode them down to the KSOs that they extract data from.

Another method for extracting the virtual address is to use the KSO_POINTER macro using the KSO number. The following example extracts the virtual address of the dispatcher globals object using its number \$7f then uses the symbolic disp_globals_type to format the record.

```
$132 ($3b) nmdebug > wl kso_pointer($7f)
$a.cefbc000
```

```
$13d ($3b) nmdebug > fv $a.cefbc000,'disp_globals_type'
```

```
RECORD
...
DGL_NUM_SWAPINS          : 0
DGL_NUM_IDLE_PROCESSORS : 0
DGL_PAUSE_UPDATE_TIME   : 28f0a1877ef107
DGL_INIT_MEAS_DONE      : FALSE
DGL_SYS_BP_ENABLED      : FALSE
DGL_SYS_DATA_BP_ENABLED : FALSE
DGL_AS_PRI_BASE         : 70ff
DGL_AS_PRI_LIMIT        : 4e7f
DGL_BS_PRI_BASE         : 4dff
DGL_BS_PRI_LIMIT        : 34ff
DGL_CS_PRI_BASE         : 33ff
DGL_CS_PRI_LIMIT        : 1bff
DGL_CS_QUANTUM_MAX      : 1c9c380
DGL_CS_QUANTUM_MIN      : 3a98
...
```

RESETING THE SESSION LIMIT:

A more succinct method for setting the value of a pointer is to extract it directly from a separate table into a debug variable. In the next example, we are going to change the maximum number of sessions from 200 to 500. The maximum is set in SYSGEN and the only supported way to get it to take effect is to change it and reboot the system. The first step is to get the pointer to the JMAT KSO. The fv command formats the object using the symbolic definition jmat_header_type. Then we modify the session limit using the mv command.

```
$f8 ($24c) nmdebug > var jp =  
symval(!hp__system_globals_ptr,'system_globals_type.jmat_ptr')  
$f9 ($24c) nmdebug > wl jp  
$11c.0  
$fa ($24c) nmdebug > fv jp,'jmat_header_type'
```

CRUNCHED RECORD

```
...  
SESSION_TY      : 1  
SESSION_NUMBER  : 1afd  
JMAT_REPLACE_1  : 0  
JOB_TY         : 2  
JOB_NUMBER      : 10fc  
...  
SESSION_FENCE   : 7  
JOB_FENCE       : 7  
SESSION_LIMIT   : c8 << 200 DECIMAL >>  
SESSION_COUNT   : b4  
JOB_LIMIT       : 5  
JOB_COUNT       : 4  
...
```

```
$fb ($24c) nmdebug > dv jp+16,1 << THE +16 CAME FROM A DV jp,10  
VIRT $11c.14      $ 00c800b4 ....  
$fc ($24c) nmdebug > mv jp+16,1  
VIRT $11c.14      = "...." $c800b4 := $01f400b4 << ONLY CHANGE THE FIRST TWO  
BYTES! >>
```

```
:showjob status  
208 JOBS:  
  0 INTRO; 4 SCHEDULED  
 19 WAIT; INCL 19 DEFERRED  
184 EXEC; INCL 181 SESSIONS  
  1 SUSP  
JOBFENCE= 7; JLIMIT= 5; SLIMIT= 500
```

CHANGING THE SYSTEM MODEL:

In the following example, we will change a fixed memory variable, the system model. This was developed by using the logic from the `sys_spu` macro. First we set up a variable `R_P` as a real memory pointer based off of the interrupt vector address (IVA) which is stored in `CR14`. Then we add 8 to offset to the end part of the CPU name. The final step is to use the `mz` command to modify real memory. By the way, the reason that this command uses the `mz` command instead of `mv` is that the address translates into a native mode code space.

```
$111 ($3d) nmdebug > :showvar hpcpname
HPCPUNAME = SERIES 935
$112 ($3d) nmdebug >VAR R_P=[REAL[REAL[REAL[REAL CR14 -30]+8]+%C]+%5C)
$113 ($3d) nmdebug > wl r_p
$13bac
$114 ($3d) nmdebug >VAR R_P_M=R_P+8
$115 ($3d) nmdebug >MZ R_P_M,2,HEX,39390000
REAL $0013bac4 = "35.." $33350000 := $39390000
REAL $0013bac8 = "...." $0 :=
$116 ($3d) nmdebug > :showvar hpcpname
HPCPUNAME = SERIES 999
```

RESETTING THE SPOOL FILE COUNTER:

Many sites reboot frequently simply to reset the current spool file counter. It turns out that the counter is kept in the first word of a KSO named `KSO_SP_OUTPUT_SPFDIR`. The following set of commands resets this counter to zero. The next spool file created will start at one, and if a spool already exists the counter will skip to the next free number. Since this is such a frequently requested and repeated event I have developed a program called `SFRESET` for the CSL that insulates the user from `DEBUG`.

```
$11b ($46) nmdebug > :listspf
SPOOLID  JOBNUM  FILEDES  PRI COPIES DEV      STATE  RSPFN  OWNER
#01048576  S4109   LP       1    1 LP      READY  MGR.TELESUP
...
$11c ($46) nmdebug > var s_v = kso_pointer(symconst('kso_sp_output_spfdir'))
$11d ($46) nmdebug > wl s_v
$184.0
$11e ($46) nmdebug > mv s_v,1,hex,$0
VIRT $184.0 = "...." $100000 := $0
$11f ($46) nmdebug > :listf,2;"lp
$120 ($46) nmdebug > :listspf
SPOOLID  JOBNUM  FILEDES  PRI COPIES DEV      STATE  RSPFN  OWNER
#01048576  S4109   LP       1    1 LP      READY  MGR.TELESUP
#01       S4109   LP       1    1 LP      READY  MGR.TELESUP
...
```

CHANGING SYSTEM LOGGING EVENTS:

The system has the ability to log many different events to disc for later summary and analysis. The LOGTOOL utility can later be used to display individual records, and other programs such as FILERPT can be used to summarize usage by file. I also have written programs that can summarize usage by session, job, logical device, and day.

The FILERPT program reports the files by number of records accessed and by FCLOSE count. The most active files on a system are usually TurboIMAGE/XL datasets. This report prioritizes which sets should be evaluated for performance tuning. Every once in a while I'll find a file that is closed hundreds or thousands times more than the next highest file on the list. This would indicate a bug or application design flaw.

These more useful log events also generate a large number of records if left enabled permanently. What we would like to do is temporarily enable certain events for a day or a shift, then restore the events back to the disabled state. The only support method is to change the logging event configuration in SYSGEN and reboot the system. The following debug steps change the LOG_MASK dynamically and allow the user to enable and disable the events at will.

From SYSGEN we can get a listing of the current settings for the system logging events. The event number minus 100 is actually the bit number in the 64 bit binary value of the LOG_MASK.

system log events	event #	status
System logging enabled	100	ON
System up record	101	ON
Job initiation record	102	OFF
Job termination record	103	OFF
Process termination record	104	OFF
NM file close record	105	OFF
...		
CM file close record	160	OFF

```
$13d ($33) nmdebug > var lp = kso_pointer($f8)
$13e ($33) nmdebug > wl lp
$a.d2318000
$13f ($33) nmdebug > fv lp,'slog_glob_rec_t'
```

RECORD

```
...
LOG_MSG_PORT      : ffffffff51
LOG_MASK          : e391800010023800
BLK_SIZE_BYTE    : 800
FILE_SIZE_BLK    : 400
CUR_FNAME_NUM    : b2
...
```

```
$141 ($33) nmdebug > dv lp,10 << FIND THE OFFSET OF LOG_MASK >>
VIRT $a.d2318000 $ 427e7ffd 00000000 00000000 00000000
VIRT $a.d2318010 $ ffffffff51 e3918000 10023800 00000800
VIRT $a.d2318020 $ 00000400 000000b2 000002e8 00000000
VIRT $a.d2318030 $ 00000005 0000003a 00000005 000000de
```



```
$143 ($33) nmdebug > mv lp+14,2 << CHANGE THE BITS FOR THE EVENTS I WANT ON >>
VIRT $a.d2318014 = "...." $e3918000 := $d7918800
VIRT $a.d2318018 = "..8." $10023800 := $10023800
```

```
$144 ($33) nmdebug > fv lp,'slog_glob_rec_t'
```

```
RECORD
```

```
...
LOG_MSG_PORT      : ffffffff51
LOG_MASK          : d791880010023808
BLK_SIZE_BYTE     : 800
FILE_SIZE_BLK     : 400
CUR_FNAME_NUM     : b2
...
```

TRACING IMAGE INTRINSIC CALLS:

The following steps will produce an TurboIMAGE/XL intrinsic trace for a specific process. The trace will contain the intrinsic name, mode, and status words of the call. The macro formats the trace area of the DBU. Unfortunately, the dataset of the intrinsic is not displayed, but the status words usually can give enough data to reveal the dataset. In teaching over 100 classes on this data base manager, I'm convinced that these 10 status words are critical to a complete understanding of how TurboIMAGE/XL works.

```
:dbdriver << RUN DBDRIVER.PUB.SYS TO GET TURBOIMAGE/XL VERSION >>
DBDRIVER C.04.04 FRI, MAR 18, 1994, 12:47 PM (C) HEWLETT-PACKARD 1978
Command: /vers
...
Procedures:
HP30391C.04.09 TurboIMAGE/XL << VERSION ID IS C.04.09 >>
...
Command: exit
:debug << ENTER DEBUG FROM CI >>
DEBUG/iX B.30.45
HPDEBUG Intrinsic at: a.0098481c hxdebug+$144
$1 ($33) nmdebug > use datinit.dat.telesup << MACRO THAT INITIALIZES ENVIRONMENT>>
Type "macstart" to load Macros & Symbols.
$11 ($33) nmdebug > macstart,1 << MACRO THAT LOADS OTHER MACROS>>
Welcome to the DAT Macro facility.
RELEASE: B.40.00 MPE XL HP31900 B.30.45 USER VERSION: B.40.00
OS Symbol file SYMOS.OSB30.TELESUP is now open.
Next line maps VAMOS.OSB30.TELESUP
1 VAMOS.OSB30.TELESUP 238.0 Bytes = 1ef0
OS Macros restored from file MOS.OSB30.TELESUP.
OS DAT MACROS HP30357 A.40.71 Copyright Hewlett-Packard Co. 1987

$10f ($33) nmdebug > :showproc ;job=@ << GET PIN NUMBER IN DECIMAL >>
QPRI CPUTIME STATE JOBNUM PIN (PROGRAM) STEP
...
C152 0:18.683 WAIT $70 52 :RESUME
C200 00:15:20 READY $70 50 (DOCLOCUP.PUB.XXXXXXX) << DESIRED PROCESS >>
...
$111 ($33) nmdebug > pin #50 << SETS WORKING PIN TO DECIMAL 50 >>
$112 ($32) nmdebug > use utiinit.ti.telesup << LOADS TURBOIMAGE MACROS >>
TurboIMAGE/XL DAT Macros
DAT Macro Version: C.03.08.
...
Enter the Turbo VUF>> c.04.09 << THIS IS THE REASON WE NEEDED THE VERSION >>
...
```

Enter Turbo/XL pin # (press RTN for current pin) >>CR << RETURN, CURRENT PIN IS #50
 >>
 DBUX virtual address: \$452e0000

ID	Data Base Name		Open Count	DBU	DBG
	DBB				
#1	LOAD PUB	XXXXXXXX	#1	\$452e8000	\$84878000
	\$848bbb40				

Input the number of the data base>> 1 << ONLY ONE DATABASE IS OPEN >>

These values will be used for the dbu, dbg, and dbb:
 dbuadr = \$452e8000 dbgedr = \$84878000 dbbedr = \$848bbb40

Are these the correct values? (Y/N)>> y

Do you need help (Y/N)? >> n << A YES WILL DISPLAY LIST OF TI MACROS >>

\$173 (\$32) nmdebug > ti_u_intrtrace << MACRO TO LIST TRACE OF INTRINSICS >>

This macro will print a trace of the last #62 intrinsics.
 It starts from the most current and ends with the least current.

NOTE: The status array is displayed in HEX!!! The pcode and mode variables are printed in decimal.

Flag	Pcode	Intrinsic Name	Mode	S1 S6	S2 S7	S3 S8	S4 S9	S5 S10
====	=====	=====	====	=====	=====	=====	=====	=====
BEG	#405	DBGET	#5	0	a	f	17b3	0
				0	9	ffffe110	1c	ffffa4d9
END	#406	DBUPDATE	#1	0	4	9	33e9	0
				1	0	0	0	0
END	#405	DBGET	#7	0	4	9	33e9	0
				1	0	0	0	0
END	#405	DBGET	#5	0	a	9	2e59	0
				0	0	ffffa62d	1a	ffffce3c
END	#406	DBUPDATE	#1	0	4	0	ffffa62e	0
				1	0	0	0	0
END	#405	DBGET	#7	0	4	0	ffffa62e	0
				1	0	0	0	0
END	#405	DBGET	#5	0	a	0	ffffa62d	0
				0	0	ffff95bd	9	2e59
END	#406	DBUPDATE	#1	0	4	0	ffff95be	0
				1	0	0	0	0
BEG	#405	DBGET	#5	0	a	1d	749a	0
				0	1a	ffff9f08	20	2bc5
END	#406	DBUPDATE	#1	0	4	1a	ffffa827	0
				1	0	0	0	0

...
 control-Y encountered

Interpreting the trace:

DBGET, mode 5 is a chained read, mode 7 is a hashed read. This program seems to be reading a chain in a detail dataset, then using a field from it to hash into a master data set to update it. In a longer version of the trace we would see the DBFIND that started the chain read. The status words from it contained the number of entries in the chain. This data would be useful for making a decision on switching the program to use a serial read and a sort of the extracted file.

The above technique can be more powerful when used in combination with other tools and DEBUG macros. The DEBUG TRACE command will give a trace of the process that includes the code offset that originates the intrinsic call. The FS_ALL_FILES macro will display everything about the files that the process is accessing.

MPE/iX DUMP ANALYSIS:

After a system hang or abort a decision needs to be made on whether or not to perform a memory dump. The dump adds measurably to the down time and may or may not be of value once the system is restarted. The decision is usually based on the frequency, nature, and timing of the event. Most sites do not perform a dump until after the problem has reoccurred. The purpose of this section is to show how to make a first pass analysis of a dump with the goal being to determine what was executing at the time of the dump and to identify probable cause.

There are three situations that will lead to the point of dumping and restarting an MPE/iX system. The obvious one is a system abort of some kind. The other two are flavors of hangs, idle and busy. An idle hang usually occurs because of a deadlock for system semaphores. The system is idle because all the processes are impeded waiting for some resource to become available. A busy hang is caused by a high priority process that is in a loop and will not allow user level processes to get any CPU.

Some system aborts are easy to analyze and determine the cause, others require more intensive investigation. The CPU busy hangs usually are very straight forward because the offending process will be the current process in the dump. The idle hangs can be very frustrating because nothing is happening at the time of the dump, and the processes that caused the deadlock may be difficult to sniff out.

LOADING UP A DUMP:

This next sequence shows how to load a dump from tape to disc. Be careful to note the system disc space situation before and after loading up the dump. On a very large and busy system, dumps will take millions of sectors of disc space. I very frequently find old dump files sitting on a system, even if the system has plenty of free disc space, you don't want to be backing up a memory dump, or even worse reloading one during an install.

```
:hello mgr.telesup.dat
...
:dat
DAT/iX B.30.45 Copyright Hewlett-Packard Co. 1987. All rights reserved.

$10 ($0) rmdat > getdump sa0 << This command copies the dump from tape to disc >>
<< SAOMEM & SAOVAR will be the disc files it creates
>>
Please mount dump volume #1.

SYSTEM ABORT 0 SS 1
Tape created by SOFTDUMP 99999X B.03.00
MPE-XL B.30.45 dumped on MON, MAY 16, 1994, 11:37 AM
```

```
...
This dump will require approximately 59.1 Mbytes (#241976 sectors) of disc space.
...
$10 ($0) rmdat > macstart,"1"
Welcome to the DAT Macro facility.
Enter the dump file set name to process: sa0
...
```

DUMP ANALYSIS:

I caused this system abort by calling system_abort directly from DEBUG. This was accomplished by using the DC command to find the address of system_abort, and the MR command to set the program pointer to this address. This in effect caused DEBUG to branch to system_abort. I had two motives for doing this; first, I wanted to show that it is very easy to find the guilty party. So please take the aforementioned disclaimer to heart. Second, I wanted a simple system abort as an example.

```
$136 ($4a) rmdat > machine_state << Macro that displays the machine status >>

SYSTEM ABORT #0 FROM SUBSYSTEM #1 (#1)
External error - subsys: #98 info: #0

SECONDARY STATUS: INFO = #0, SUBSYSTEM = #1 (#1)
External error - subsys: #1 info: #0

MPE/XL VERSION: B.30.45          Current CPU Last Pin : $4a << This is the one >>

SYSTEM CONSOLE AT LDEV #20

Processor: 00 HPA: fff80000 IVA: 00125000 Config: TRUE

Current CPU: 0 Original CPU: 0 Monarch CPU: 0 MP array at: 7ba000

      CPU($0):   PROCESS_RUNNING   Last Pin : $4a

HP3000 SERIES 99999 With Processor Revision 0. << Someone has changed the model #
>>

CURRENT REGISTERS:

R0 =00000000 800206f0 0098481c 00000002 R4 =40332b28 403326e0 40332318 00000400
R8 =00000000 40331280 00000045 00000049 R12=00000000 00000000 00000000 00000000
R16=00000000 00000000 00000000 00000000 R20=00000001 00000001 800206f0 40332b28
R24=00000001 00000001 00000001 c0202008 R28=00000000 00000000 40332ff0 00000001

IPSW=0006000f=jthlnxbCVmrQPD1 PRIV=0 SAR=0011 PCQF=a.167894 a.167898

SR0=0000000a 00000000 00000000 00000000 SR4=0000000a 00000658 0000000b 0000000a
TR0=007fe200 0083e200 00139034 00000000 TR4=40333148 40332ff0 cefbc000 403324dc
PID1=05c0=02e0(w) PID2=00e2=0071(w) PID3=0000=0000(w) PID4=0000=0000(w)

RCTR=fffffff ISR=0000000a IOR=00000000 IIR=00020005 IVA=00125000 ITMR=d6241abc
EIEM=fffffff EIRR=00000000 CCR=0080
```

\$137 (\$4a) nmdat > pm_program << What was the cuuent pin's program? >>

PROGRAM FILE: CI.PUB.SYS

\$138 (\$4a) nmdat > pm_trace << Macro to show call sequence >>

PC=a.00167894 system_abort << system_abort called by hxdebug >>

NM* 0) SP=40332ff0 RP=a.0098481c hxdebug+\$144 << Debug called system_abort! >>

NM 1) SP=40332ff0 RP=a.0098d558 exec_cmd+\$7b4

NM 2) SP=40332b70 RP=a.0098cd70 ?exec_cmd+\$8

export stub: a.0098eff4 try_exec_cmd+\$c8

NM 3) SP=40332b20 RP=a.0098cbf8 command_interpret+\$354

NM 4) SP=403326c8 RP=a.0098c870 ?command_interpret+\$8

export stub: a.0098fdc8 xeqcommand+\$194

NM 5) SP=40332278 RP=a.0098fc20 ?xeqcommand+\$8

export stub: f4.000067d8

NM 6) SP=403321f8 RP=f4.0000745c

NM 7) SP=403321b0 RP=f4.00000000

(end of NM stack)

\$13b (\$4a) nmdat > ui_job << Who done it? >>

Information About Job/Session #S4166

JOBNUM STATE IPRI JIN JLIST INTRODUCED JOB NAME
#S4166 EXEC 8 9 9 136 11:11 MGR.TELESUP,PATCHXL

JSMAIN
PIN
\$26

PROCESS TREE

=====

JSMAIN PIN : \$26

Parent PIN	Program File	JSMAIN PIN	Program File
-----	-----	-----	-----
\$21	SESSION.PUB.SYS	\$26	JSMAIN.PUB.SYS

Family Tree for Process Number \$26

\$26 (JSMAIN.PUB.SYS)

\$4a (CI.PUB.SYS)

\$2f (VTSERVER.NET.SYS)

\$28 (VTSERVER.NET.SYS)

DEVICES

=====

LDEV(s) associated with #S4166 :

#9

CURRENT COMMAND

=====

Last command executed by PIN \$4a is:

debug << user ran debug from colon prompt >>

COMMAND HISTORY STACK

=====

CI History Stack for PIN # \$4a

1) LISTF,2

...

11) debug

COMMAND INTERPRETER VARIABLE(S)

=====

Job/Session Variables for Job #S4166

```
Variable Name      Value
-----
CIERROR           Integer      : 908
...

```

```
TEMPORARY FILES
=====
No temp files

```

```
CAPABILITY INFORMATION
=====
ALLOW MASK :
No operator commands ALLOWED for job/session #S4166

```

```
USER ATTRIBUTES :
SM,AM,AL,GL,DI,OP,UV,LG,CS,ND,SF,IA,BA,PM,MR,DS,PH

```

COMPATIBILITY MODE SUDDEN DEATH:

In the following example we have a system abort from a compatibility mode program. The dump shows the reason for the abort and the state of the offending process. This was a one time abort and it since has not repeated. The names of the programs have been changed to protect their identity.

```
$14a ($115) nmdat > machine_state
```

```
SYSTEM ABORT #2559 FROM SUBSYSTEM #99 (CM MPE)
A COMPATABILITY MODE sudden death has occurred. The error number displayed on
the console is the sudden death number.
```

```
SECONDARY STATUS: INFO = #16, SUBSYSTEM = #99 (CM MPE)
External error - subsys: #99 info: #16
```

```
MPE/XL VERSION: B.30.45          Current CPU Last Pin : $281
...

```

```
$14e ($281) nmdat > pm_program << The Variable PIN is set to $281 >>
```

```
PROGRAM FILE: FATHER.PUB.ACCOUNT
```

```
$14f ($281) nmdat > pm_trace
```

```
PC=a.00185894 system_abort
NM* 0) SP=403325b8 RP=a.00380e24 sudden_death+$54
NM 1) SP=403325b8 RP=a.00380d9c ?sudden_death+$8
      export stub: a.004b5944 arg_regs+$28
NM 2) SP=40332570 RP=a.004a7884 nm_switch_code+$984
NM 3) SP=40332440 RP=a.00496fd4 Compatibility_Mode
      (switch marker frame)
CM   SYS % 213.7323 SWITCH'TO'NM'+%4 (Mitroc CCG) SUSER1
CM * 0) SYS % 213.7323 SWITCH'TO'NM'+%4 (Mitroc CCG) SUSER1
CM 1) SYS % 153.4173 SUDDENDEATH+%15 (Mitroc CCG) CMSWITCH
CM 2) SYS % 152.107 DSTVIOLATION+%2 (Mitroc CCG) CMININ
CM 3) SYS % 241.27677 AS'ENV'GET'ENTR+%5 (Mitroc CCL) ASENVSEG
CM 4) SYS % 241.26661 AS'ENV'GET'WAIT+%30 (Mitroc CCG) ASENVSEG
CM 5) SYS % 241.26131 AS'ENV'UNLOCK'D'+%6 (Mitroc CCG) ASENVSEG
CM 6) SYS % 242.10313 DSEXPIRE+%47 (Mitroc CCE) DSUTIL
CM 7) SYS % 162.1247 FREE'SECONDARY'+%23 (Mitroc CCL) KNSEG1
CM 10) switch marker (Mitroc CCG)
```

```

NM 4) SP=40332120 RP=a.004a6530 switch_to_cm+$818
NM 5) SP=40331f30 RP=a.0037b348 terminate_process.free_cm_resources+$84
NM 6) SP=40331bd8 RP=a.0037b9a8 terminate_process+$4f4
NM 7) SP=40331b90 RP=a.0037b480 ?terminate_process+$8
    export stub: a.00474c28 TERMINATE+$20
NM 8) SP=403317a0 RP=a.00474bd4 ?TERMINATE+$8
    export stub: a.004b5944 arg_regs+$28
NM 9) SP=40331710 RP=a.004a7884 nm_switch_code+$984
NM a) SP=403315e0 RP=a.00496fd4 Compatability_Mode
    (switch marker frame)
    CM 11) SYS % 153.0 ?TERMINATE (Mitroc CCG) CMSWITCH
NM b) SP=403312c0 RP=a.00929294 outer_block+$148
NM c) SP=403310d0 RP=a.00000000
    (end of NM stack)

```

```

$150 ($281) nmdat > pm_pib_info << This macro formats the process information
block >>

```

```

===== FAMILY TREE =====
PIN STATE PARENT JSMAIN CHILD SIBLING LEN PORT MISC VALUES SWITCH SWITCH
-----
$281 DYING $24e $9c $115 $0 $f50 $0 $1 $2

```

```

===== DISPATCHER INFORMATION FOR A PROCESS =====
c PIN # State Wait Event Pri Class Blocked Reason
-----
$281 EXECUTING Not Waiting $33ff CS NOT_BLOCKED
Last priority : $33ff
Hold priority set :
[ DSP_SIR_HOLD]

```

```

$151 ($281) nmdat > pin $115 << Change the value of PIN to look at the child
process >>
$152 ($115) nmdat > pm_program

```

```

PROGRAM FILE: CHILD.PUB.ACCOUNT

```

```

$153 ($115) nmdat > pm_trace
PC=a.0017a6e8 enable_int+$2c
NM* 0) SP=4035fa88 RP=a.002f7918 notify_dispatcher.block_current_process+$480
NM 1) SP=4035fa88 RP=a.002fb340 notify_dispatcher+$254
NM 2) SP=4035f9f8 RP=a.0043f2fc wait_for_active_port+$ec
NM 3) SP=4035f918 RP=a.0043fd94 receive_from_port+$320
NM 4) SP=4035f8b8 RP=a.0037fb18 ipc_wait_process+$384
NM 5) SP=4035f758 RP=a.003acd08 getsir+$204
NM 6) SP=4035f548 RP=a.00a1eec8 getsir_stub+$24
NM 7) SP=4035f488 RP=a.00a1ee90 ?getsir_stub+$8
    export stub: a.00496fd4 Compatability_Mode
NM 8) SP=4035f410 RP=a.004a6530 switch_to_cm+$818
NM 9) SP=4035f220 RP=a.0037b348 terminate_process.free_cm_resources+$84
NM a) SP=4035eec8 RP=a.0037b9a8 terminate_process+$4f4
NM b) SP=4035ee80 RP=a.0037b480 ?terminate_process+$8
    export stub: a.0037e0ac pm_interrupt_handler+$5c4
NM c) SP=4035ea90 RP=a.004d70ac execute_interrupt+$158
NM d) SP=4035e8d8 RP=a.002c69d0 process_int+$34
NM e) SP=4035e7e0 RP=a.00136038 hpe_interrupt_marker_stub
--- Interrupt Marker
NM 1) SP=4035e7a8 RP=a.004b57d0 return_from_hidden_pi
--- End Interrupt Marker Frame ---
PC=a.004b57d0 return_from_hidden_pi
NM 0) SP=4035e6b0 RP=a.00380acc ipc_awake_process+$9c0

```

NM 1) SP=4035e6b0 RP=a.002895b4 ACTIVATE+\$4e4
 NM 2) SP=4035e430 RP=a.002890bc ?ACTIVATE+\$8
 \$154 (\$115) nmdat > pm_pib_info

```

===== FAMILY TREE =====
                                     ===== MISC VALUES =====
                                     CM      SWITCH  SWITCH
                                     TOS    TO      TO
PIN  STATE  PARENT  JSMAIN  CHILD  SIBLING  LEN  JSMAIN  PORT  CM  NM
-----
$115 DYING  $281   $9c   $0     $0     $e50   $0     $1     $0

```

...

```

===== DISPATCHER INFORMATION FOR A PROCESS =====
c PIN #  State      Wait Event  Pri  Class  Blocked Reason
-----
$115    LONG_WAIT   IPC                      $33ff CS  SIR_WAIT

```

...

\$158 (\$115) nmdat > rm_format_sirs << Child was waiting on a SIR >>

 SIR Table (Locked SIRs)

SIR \$136 : Job Sir for JPCNT Index #267
 Owner PIN: \$281
 # of Waiting Processes: \$1
 Waiting PINs: \$115

\$15e (\$115) nmdat > pin \$24e << Switch to parent process >>

...

\$163 (\$24e) nmdat > ui_cihistory

CI History Stack for PIN # \$24e

- 1) chgroup samdta
- 2) listf
- 3) RUNF
- 4) SHOWGROUP
- 5) LISTGROUP SAMDTA
- 6) NEWGROUP SAMDTA1;CAP=IA,BA
- 7) NEWGROUP SAMDTA2;CAP=IA,BA
- 8) CHGROUP SAMDTA1
- 9) RUNF

\$164 (\$24e) nmdat > ui_showvar

Job/Session Variables for Job #S7264

```

Variable Name      Value
-----
CIERROR           Integer      : 9113
HPCIERR           Integer      : 9113
...

```

BIOGRAPHICAL SKETCH:

Michael Hornsby, President of Beechglen Development, Inc and SIGIMAGE Member. Mike has given numerous INTEREX presentations and tutorials over the past few years. His company's main mission is to provide custom software and support services for HP3000 sites worldwide.

PAPER 8011
Life With a "Cluster" of HP-3000s
Steve Thomas
Graco Children's Products, Inc.
P.O. Box 100
Elverson, PA 19520
(610) 286-5951

It is amazing what will keep you up at night. Yet baby swings can do that. I can lose sleep over a cluster of HP-3000s performing overnight processing. That overnight processing is critical to GRACO Children's Product's, Inc. ability to produce baby swings and other popular juvenile products for retailers all over the world. It must run. If we can not accept Electronic Data Interchange (EDI) orders from our customers, plan for manufacturing, ship product, or handle calls from customers and consumers we lose business, and profits, FAST. Retailers are not noted for their patience. If we can not ship baby swings or other juvenile products to them, they will go elsewhere. They are constantly squeezing the time between order placement and shipment.

Efficient business systems are critical. They are becoming more critical every day. To say that our HP-3000 based computer systems are vital to the process of producing and delivering GRACO's products is a gross understatement. Our systems have to be available. All day every day. GRACO has designed our information systems to support this rapidly growing need for information by our staff, customers and consumers.

So, baby swings can keep me awake at night because part of my job is to see that we can ship swings and other products. The cluster of five HP-3000s that I manage have helped me to sleep better at night and have helped to achieve some of the GRACO corporate goals.

'Meet and exceed consumer's expectations' is one of our three corporate goals at GRACO. 'Achieve financial strength and stability' and 'Meet and exceed employee's expectations' are the other two. From the CEO down to the assembly worker everyone at GRACO works to meet those goals. That includes the MIS department. Information technology has become one of our means to meet our corporate goals and the cluster of HP-3000s in use at GRACO plays a large part in meeting those goals. We MUST be able to deliver the product the consumer wants, when the consumer wants it, at a price the consumer is willing to pay. The central information systems that we have built have harnessed the latest in Hewlett-Packard technology helps us meet those goals. Software from third parties augment HP machines to create a mainframe class environment around a cluster of super minicomputers. Every day they help us to meet the consumer's expectations by helping us produce more efficiently. With the cluster of systems we have been able to keep information systems costs down helping to achieve GRACO's financial goal. By providing fast response times on our current HP-3000 based applications we are helping to meet employees work expectations.

Corporate Goal

GRACO's Cluster Helps to Achieve Goal By . . .

- * Meet and Exceed Employee Expectations
 - 1. Exceptional on-line response times eliminate on-the-job frustration.
 - 2. Capacity available to add a resource-intensive Human Resources application that will improve company-employee relations.

- * Achieve Financial Strength and Stability
 - 1. Providing a lower cost computing environment for our existing applications.
 - 2. Allowing capacity to be added or removed with lower marginal cost.
 - 3. Providing licensing options for new software at lower pricing tiers.

- * Meet and Exceed Consumer's Expectations
 - 1. Lowering overhead as compared to larger corporate systems.
 - 2. Permitting implementation of an efficient 'consumer service' system.
 - 3. Providing a low cost link to retailers improving product availability.

EVOLUTION TO A CLUSTER OF SYSTEMS

In 1989 GRACO made use of one HP-3000 for its business computing needs. The Series 70 in use at the time was at full capacity and no additional applications could be added. When the requirement to add Electronic Data Interchange (EDI) software was recognized, there was no place for it on the 70. A micro3000 was brought in to provide a breathing space until a new Series 960 could be installed. The intent was to eliminate both the 70 and the micro3000 when the Series 960 was in place. While pricing the new 960 system the MIS management discovered that the RJE link hardware and software required to run the EDI system would cost more than the cost of keeping the micro3000. Software cost economics drove us to add the second machine. The economic reasons to make such a decision still exist today. New technical innovations also make multiple systems a easier choice to make. With the addition of the micro3000 GRACO's shift to a cluster of systems began. Over the next four years, as additional computing power was required we have added additional machines as opposed to upgrading a central system.

We look on our systems as being a cluster because we have them physically and electronically networked them close together. Each systems processing power is additive so that the capacity of the whole is greater than each individual system. Our systems may not have the batch performance of the Series 99X Corporate Business Systems but they can produce a very respectable on-line transaction throughput. While HP-3000s can not be clustered in quite the same way that VAX systems can, using HP and third-party networking products can provide a powerful system capable of supporting mid-size and larger businesses.

Since the introduction of DS/3000 in the late 1970s, networking HP-3000s together has become progressively easier and more efficient. While some expertise is required particularly when managing a large group of networked computers, it is no longer the proverbial 'rocket science' that it was in the past. Today's NS/3000 product is VERY MUCH easier to install and configure than its predecessors. The screen driven configuration program includes a simplified 'guided config' option that allows the easy input of standard parameters. Extensive 'help' is available from anywhere via the f7 function key. With thousands of HP-3000 systems networked together around the world, the HP Response Center has a large amount of experience with networking systems and can be of assistance when first beginning to 'cluster' HP-3000s.

Steps in Networking HP-3000s

- * Select a network transport such as ThinLAN Link, Token Ring, etc.
- * Select desired network services such as NS/3000, ARPA, NetBase, etc.
- * Install and configure on each node.
- * Start the services

With the exception of a small system in the Far East, all of GRACO's HP-3000s are housed in the corporate datacenter. This reduces our operating and staff costs. With the availability of high speed telecommunications it is possible to build a cluster of HP-3000s with some or all of the individual computers located in different places. Dispersing the systems can be part of a disaster recovery strategy. One company with an office in southeastern Pennsylvania uses just such a strategy to protect their business computing. The main office in New England has two HP-3000s networked together then a third is outside Philadelphia. This system is on a high-speed network link to the main office. Data from the main office is 'shadowed' to the remote office real-time.

GRACO Children's Products, Inc.
HP-3000 Computer Systems Configuration

Table 1

Configuration	HP-3000 Series				
	987	947	927	917	Micro
Slots	12	2	2	2	---
Memory (Mb)	256	160	120	40	4
Disc (Gb)	16.0	8.5	9.3	4.6	0.3
SCSI	4	2	2	1	---
HP-IB	1	0	0	0	1
HP-FL	1	0	0	0	---
PSI	0	0	1	1	1 (INP)
ThinLAN		1	1	1	1
User License	256	100	20	8	---
Avg Users	103	46	9	4	2
Performance *	32.0	10.0	10.0	10.0	1.3

* HP-3000 Series 44 = 1.0 Source: April, 1994 HP Advisor Volume 1, Number 3, Page 4

Table 1 shows the configuration of our systems that we have clustered in our Pennsylvania data center. The aggregate performance of our systems is 63.3 which compares favorably with an HP-3000 Series 992/200. We can not claim to get 992/200 performance in batch applications. When considering on-line transaction throughput our cluster can provide similar speed.

HP's OpenView DTC Manager manages the terminal connections to the HP-3000s in GRACO's datacenter. This provides us with the ability to give users access to any system on the network from their desktop. Networked PCs can also access any computer via the 3000 Connection from Walker, Richer, and Quinn. Our goal is to provide a single, consistent system view for all users so that they do not have to concern themselves with the details of logging on to a particular host. We want to fulfill the saying 'the computer is the network, the network is the computer' as they say in the Sun Computer Systems advertisements.

HP OpenView DTC Manager Benefits:

- * Access to computers and devices on the network
 - > MPE/IX, MPE/V, and HP-UX systems
 - > Non-HP systems such as SUN and DEC
 - > Other devices such as modems, FAX machines, etc.
- * Centralized management of local and remote DTCs
- * Simple, superior fault detection and diagnosis
- * Simplified maintenance over multiple systems
- * Easy configuration of ports
- * Powerfail monitoring

Another HP OpenView product, OperationsCenter, provides the ability to centrally manager systems including UNIX computers. With OperationsCenter computers located inches or miles apart can be supported locally reducing staff requirement. Should your application need computers located in different facilities this is a way to manage them and keep costs down. We do not use OperationsCenter at GRACO but may consider it in the future should we need to manage systems at our other plant facilities.

GRACO Children's Products, Inc.
HP-3000 Based Software Applications

Figure 1

Software Application : Automatic Data Collection (ADC)
Developed By : In-house Staff and Eagle Consulting
Host Computer : 987
Availability Requirement : Very High-24 Hours per Day, Mon.-Sat.
Backup System : 927
Average Users : 10

Software Application : Consumer Service (STAR)
Developed By : Strategic Marketing Solutions
Host Computer : 947
Availability Requirement : High - 3:00AM to midnight, Mon.-Sat.
Backup System : 987
Average Users : 32

Software Application : Electronic Data Interchange (EDI)
Developed By : EDI Solutions and In-house Staff
Host Computer : 917
Availability Requirement : High - Times Vary
Backup System : 927
Average Users : 2

Software Application : Financial - A/P and General Ledger
Developed By : Spectrum Associates
Host Computer : 947
Availability Requirement : Low-Peak usage-Friday AM and Month-end
Backup System : ----
Average Users : 4

Software Application : HPDesk Electronic Mail
Developed By : Hewlett-Packard
Host Computer : 987
Availability Requirement : Mod.-Peak usage-Mon.-Fri.8AM to 8PM
Backup System : ----
Average Users : 10

Software Application : MRP II
Developed By : Spectrum Associates
Host Computer : 987
Availability Requirement : Moderate - Monday thru Saturday
Backup System : ----
Average Users : 45

Software Application : Payroll/Personnel
Developed By : High Line
Host Computer : 927
Availability Requirement : Low Friday thru Tuesday - High
Wednesday and Thursday
Backup System : 987
Average Users : 10

Software Application : Sales and Order Entry
Developed By : In-house Staff and Spectrum Associates
Host Computer : 987
Availability Requirement : High
Backup System : ----
Average Users : 35

MAKING THE CLUSTER WORK

We have built our cluster and assigned applications to members of that cluster. Each assignment was based on the attributes of the application and its role in our computing environment. Applications with a small user community or a heavy batch orientation have been assigned to the smaller HP-3000s. Systems with a need for higher transaction throughput are installed on the larger systems.

GRACO uses eight major HP-3000 based applications. See figure 1 for a list. Most do not have on-line interfaces with other applications. For example, our Consumer Service application is used to handle telephone inquiries from consumers of our products. The systems is used to handle everything from questions on the use of the product to ordering spare parts to processing damage claims. This system must be able to handle a high volume of transactions but it does not need to interface directly with our manufacturing system.

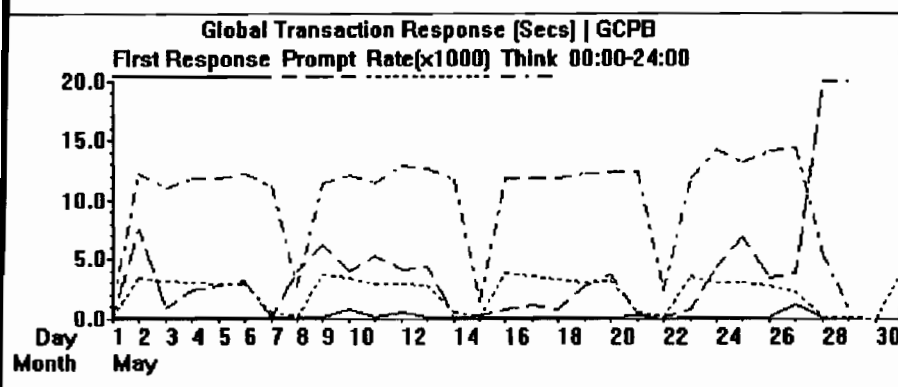
With the 9X7 and now the 9X8 Series of HP-3000, Hewlett-Packard has produced a range of computer systems that can be used to support specific applications. An HP-3000 Series 917LX provides over twice the computational power of a Series 70 at a fraction of the cost. This makes it possible to have individual systems serve a small number of applications cost effectively. For example, a 918LX system with a 20-user license, Allbase and IMAGE/SQL, 64Mb memory, 4.0Gb disc, and ThinLAN link software can be purchased for around \$30,000 based on HP's February, 1994 price list. Upgrading a 987SX to a 987/150SX to support about 20 more users would cost at least \$48,000 without adding any disc storage or memory.

We have used individual systems to handle some of our key applications. To process Electronic Data Interchange transactions from our retailer partners a Series 917 fits that niche in our cluster. As with many other consumer businesses, EDI has become a vital part of our business. Over 80% of our orders come to us via EDI. This process, while computationally intensive, involves few on-line users. Once the EDI transactions from a retailer are translated they can be made available to other computers in the cluster in real time. With a processor more than twice as powerful as a Series 70 by HP's measurements, the 917 can handle complex EDI translations without impacting other on-line users. Software for this tier 1 System Processing Unit (SPU) can be 10% to 15% of the cost of high-end licenses.

A very important part of our business is customer service. One of GRACO's corporate goals is to meet and exceed the consumer's expectations for our products and we have a toll-free number for consumers to use for questions, parts, and assistance with our products. Management here has determined that providing superior customer service gives us an advantage over our competitors in the juvenile furniture industry. To serve this requirement we have dedicated most of a Series 947LX system to our Customer

service application. This system handles hundreds of calls on GRACO's 800 number daily with one second or less response time. Data from HPLaserRX shows that we have been able to consistently meet that goal as shown on Graph 1. Response to prompt has been consistently under 0.5 seconds. In the past it would have been very expensive to dedicate a system like this.

Graph 1 Response Times on the HP3000 Series 947 for May, 1994.



Handling over 2300 calls per day, the 947 provides Customer Service with excellent response.

Manufacturing systems at GRACO combine a fairly traditional MRP system with automated data collection to provide greater manufacturing efficiency. Barcodes on work orders and templates give production workers an easy way to collect shop floor data very accurately. This system operates three shifts per day, six days per week. Even a few minutes of downtime can be an enormous problem. Our HP-3000 cluster supports the manufacturing application with a minimum of downtime. Shadow copies of critical databases are stored on the cluster to provide backup and recovery capabilities for the data collection system.

The local area network that ties the HP-3000s together is the glue that binds our systems together. Thanks to the network processes on one machine can cooperate with other computers and users can easily access multiple systems without much difficulty. Having multiple systems does add complexity to the computing environment. For example, you must plan for the situation when one of the members of the cluster is down. Some of the batch scheduling packages available such as JobTime from NSD permit you to hold a production run if a system is not available. You can do the same thing without a job scheduler by carefully coding your JCL.

BENEFITS AND CAUTIONS

Here are some of the benefits that we have found using multiple HP-3000s in a cluster:

1. Lower software costs. It is possible to license software on a smaller machine than would otherwise be necessary saving thousands in licensing fees.

2. Lower hardware costs. Depending on the size of the upgrade required, an additional computer could be more cost effective.
3. Multiple systems can provide fault tolerance. One system in the cluster could pick up the processing from another that is down. An example of this will be discussed later in this article.
4. Users can be insulated from peak processing demands placed on other machines in the cluster. Month-end processing on one member of the cluster will not directly impact performance on other members.
5. Software updates and patches can be tested on a small system before introducing them to larger systems. This is particularly useful for operating system updates and new utilities such as MPEX or DBGENERAL. The programming and operations staff have an opportunity to test new software.
6. Upgrades occur as several incremental changes over a period of time rather than large, disruptive events every few years.
7. The cost of planned and unplanned downtime is spread out among the users. One computer that is down does not disrupt operations for all.

There are some drawbacks to this approach that must be considered:

1. There is some more administrative work with multiple HP-3000s. Each system has its own software license and support agreements that must be managed.
2. Each system will require operating system upgrades. Instead of updating software on just one system, several will have to be done.
3. There is more work in managing several systems on a network versus one single system. The MIS staff must have the expertise to create and maintain the network though this is much easier than it was in the past.
4. Careful planning is vital to ensure that one system in the cluster does not end up carrying most of the processing load while others have excess capacity.
5. Multiple applications that are tightly integrated will not perform as well if split across different member of a cluster.

During a recent upgrade cycle when we replaced our Series 960 with a Series 987SX, we briefly considered a board-swap upgrade for the 960 to a 980/100. HP is aggressively discounting the cost of these upgrades which would have saved us some hardware costs. However, the 980/100 is in the top tier of HP's software pricing structure, tier 7. The 987 system is a tier 6 system. Since the 960 was also a tier 6 system GRACO saved thousands in software uplift charges by remaining at the tier 6 level. Should we decide to add software to the 987 system in the future this choice could continue to save us thousands of dollars in license fees. Hardware and software support costs would have increased about 15% had we chosen the 980/100 system. With the 987 software support costs remain flat while hardware support decreased substantially.

Our heavy month-end processing period is spread out over all of the 9X7 systems in our cluster. Each system handles a part of the process then passes the results to the 947 for consolidation in the General Ledger. Users can still see some impact of the extra usage but we believe that their overall system response is better with our cluster than if we had one centralized system.

As mentioned before, GRACO makes great use of HPDesk. Our cluster arrangement gives us a platform to install and test the new HPOpenDesk update on one of our smaller systems before we release it to the user community. Our 927 system has a license for HPDesk and we use that system to test and demonstrate not only HPDesk but other software packages as well. The payroll application that runs on that system is used 8am to 5pm Monday through Friday so it is easy to find time to add patches or updates after hours.

In the future, should we decide to add new HP-3000 based applications to our cluster, we can pick and choose the platform that makes sense from a user cost point of view. If the application is priced based on the user license or CPU size, we have choices as to where to install it. Software with few users could, perhaps, be added to our 927 system or the 947 system saving the cost of licensing for the large system.

As we look into new software solutions for our business UNIX, Windows NT, and other operating environments are always a possibility. With multiple HP-3000s we can move applications to new platforms in an orderly fashion. Over time we could allow the lease on one or more of the HP-3000 system expire if they are no longer needed. This will allow us to avoid the situation where we would be left with a large, expensive HP-3000 supporting a dwindling application base. Similarly, if we were forced to downsize our operation one of the leases could be allowed to expire and the smaller user base could be consolidated onto the remaining machines. Four of our five HP-3000s based in Pennsylvania are on leases. This fall the lease on the micro3000 will expire and it is unlikely that we will purchase the system or extend the lease. The 917 and 947 systems have leases that run until 1996 and the 987 is leased until 1999. The 927 was purchased for an overseas project that was canceled. To be honest we did not plan this way when we began leasing the systems. However, this arrangement will give us the opportunity to evaluate our systems needs over time, dropping or adding capacity as our computing environment changes.

Calculating the cost of downtime is a topic unto it self and I will not try to discuss how to determine that cost. It is sufficient to say that companies depend more than ever on reliable information systems to keep the business going. During the past two years at GRACO the tolerance of computer downtime has decreased dramatically with the introduction of automated information collection systems and the increase in customer demands for fast order turn around. Using a cluster of systems can spread the risk of a system failure shutting down the entire business for a period of time.

The extra administrative burden of multiple systems can not be overlooked. Separate hardware and software contracts are written for each system. HP is usually very helpful in assisting in managing the contracts by making renewal time correspond with the user's fiscal accounting periods. Often the number of tapes, documentation, support materials, and related items seems to increase exponentially as you add systems.

Keeping track of the software patch level of the systems in the cluster can be difficult as well. Here are some suggestions that can make this task less time consuming.

- * Select one system to be the 'test' computer. Install and test all patches on that system first. After the patches have been on the test computer for a short period of time install them on the rest of the systems in the cluster. At GRACO we use our 917 or our 927 systems to test software patches because they have few on-line users to inconvenience should we experience problems with a patch. The growing importance of EDI is pushing us toward using the 927 for this testing more often.

- * Unless you can not because of a particular software system requirement, apply all patches that you get for any system to all systems. It is very often the case that a patch for one computer will apply to all. It is of great benefit when you are experiencing trouble with one computer on the cluster. The HP Response Center will frequently ask about patch levels and having the systems consistent will simplify the search for the problem.

- * Do not allow much time to elapse between the 'test' computer patch installation and the installation on other systems. 'Much time' is relative to the type of patch and systems effected. How critical the systems are and how urgently the patch is needed will determine how quickly the software is rolled out to other systems.

- * Work with the HP Response Center to make sure you have the latest versions of patches before you apply them. HP is constantly updating the software for critical operating system components. Before installing any patch that has been around for some time check to ensure that it has not been superseded by a more current version. If, for example, you discover that you need a NS transport patch you may also wish to obtain other general release patches for NS at the same time.

NETBASE

NetBase from Quest Software has taken our cluster approach to a new level. This software was selected because it has the capability to make resources on all of the systems in our cluster much more readily available than before. It makes very good use of our network to efficiently share files and printers among the members of the cluster. In the past we made use of HP NS/3000 and remote file access to do some data transmission between systems. With NetBase this process is many times more efficient. We now have the capability to maintain multiple copies of a database or file on several systems with each as up-to-date as the master copy.

NetBase Features:

- * FAST Network File Access
- * Network Spooling
- * File and Database Shadowing
- * Detailed Transaction Statistics
- * Automatic Remote Process Management (AutoRPM)

NetBase operates by intercepting file system calls and redirects the I/O to the proper system using user-maintained tables. Utilizing low-level NetIPC calls, it very efficiently moves transactions from one computer to another. It can also very efficiently destroy your system if you are not careful!

A team was formed at GRACO to oversee the implementation of NetBase on our computers. We recognized the vast benefits and the great dangers of the product and were determined to have a smooth implementation. To begin we tested the system on a small scale with a few files in test accounts. This provided us with a good feel for how the software works. At about the same time we reviewed our applications to determine exactly how NetBase could improve them. From this discussion we listed our applications in order of importance to the company. Time constraints and ease of recovery were considered when evaluating our applications.

GRACO Children's Products, Inc.
Critical Applications

1. Manufacturing Data Collection (ADC)
2. Customer Service
3. Payroll
4. Electronic Data Interchange (EDI)
5. Manufacturing
6. Sales and Order Entry
7. HPDesk Electronic Mail

Three days of training on-site by a Quest Software support person proved invaluable in helping us understand how to use the product. We were also given installation assistance because NetBase must be added to the system libraries.

Our first production use of NetBase was with a KSAM file used by the EDI system and the Sales department. Previously this file was maintained on the Series 917LX with the Sales personnel logging on to that system to update it. This presented a problem since the 917 has but an eight user license. At times when there was development work going on the 917 would reach its user limit. Using NetBase we now update that file on the 987 system and have a shadow copy available on the 917 for the EDI system to read. This eliminated the need for the Sales department to log on to another system to maintain that file.

Figure 3

Sample NetBase Directory Entries

GCP_A:NBDIR.PUB.NETBASE

NetBase Directory Program [0.9.4] Update 9 (C) QUEST Software 1987

D> LISTF @.@.@

```

Node: GCPA      Directory: NBD.DATA.NETBASE      O D F S P
                                                    P S L Y R L
-----LOCAL REFERENCE-----  -----REMOTE EQUIVALENT-----  E E A N O O
File   Group   Account   File   Group   Account   Node   N T G C T G
FMGL1  DATA   FMCORP   ->      --Same--      GCPB   - - - - -
PRODCUST PUB   FMGCP   -> PRODCUST DATABASE EDITRAN4 <Local> - - L - - Y
      (Shadow)      GCPC
PRODUCTS PUB   FMGCP   ->      --Same--      <Local> - - L - - Y
      (Shadow)      GCPC
ADCDB   DATA   FMGMP   ->      --Same--      <Local> - - I - - Y
      (Shadow)      GCPD
ADCWK   DATA   FMGMP   ->      --Same--      <Local> - - I - - Y
      (Shadow)      GCPD
FMMF2   DATA   FMGMP   ->      --Same--      <Local> - - I - - Y
      (Shadow)      GCPD
FMPC    DATA   FMGMP   ->      --Same--      <Local> - - I - - Y
      (Shadow)      GCPD
ADCDB   DATA   FMSC    ->      --Same--      <Local> - - I - - Y
      (Shadow)      GCPD
ADCWK   DATA   FMSC    ->      --Same--      <Local> - - I - - Y
      (Shadow)      GCPD
FMMF2   DATA   FMSC    ->      --Same--      <Local> - - I - - Y
      (Shadow)      GCPD
FMPC    DATA   FMSC    ->      --Same--      <Local> - - I - - Y
      (Shadow)      GCPD
ADCDB   DATA   FMTRAIN ->      --Same--      <Local> - - - - - Y
      (Shadow)      GCPD
ADCWK   DATA   FMTRAIN ->      --Same--      <Local> - - - - - Y
      (Shadow)      GCPD
REQDB   PUB     STAFF   ->      --Same--      <Local> - - - - - Y
      (Shadow)      GCPB
D>

```

Our next, more ambitious use for NetBase was to improve the availability of our Manufacturing Automatic Data Collection System (ADC). Using Hewlett-Packard's TurboSTORE II product with on-line backup we had been able to reduce the downtime to around 20 minutes per day before the introduction of NetBase. Our goal is to have that part of the manufacturing system available 24 hours per day Monday through Saturday. Using NetBase we placed a copy of the ADC system databases on the 927 system. All updates to the databases on the host 987 will be shadowed to the databases on the 927. When backups are desired, posting to the shadow databases is stopped and the databases are copied to tape on the 927 system using TurboSTORE II with on-line backup. When TurboSTORE has finished preparing the system for the on-line backup, posting to the shadow copy can be resumed. There no longer needs to be any downtime for the ADC system. In practice though we do take the ADC and NetBase systems down to do a weekly full backup.

Figure 4 NetBase Shadowing Activity

```
GCP_D:run NBCTRL.PUB.NETBASE;info="SHOW POST 1"
```

File	Type	Updates	Time Stamp of Last Record Processed	S	Q	D	Y	U	I	N	E	S
ADDCB.DATA.FMDRYRUN	IMAGE	1122	05/18/94 14:57:43	-	-	-	-	-	-	-	-	-
ADDCB.DATA.FMGMP	IMAGE	1091	05/18/94 14:52:35	-	-	-	-	-	-	-	-	-
ADDCB.DATA.FMSC	IMAGE	964	05/18/94 14:56:01	-	-	-	-	-	-	-	-	-
ADCMK.DATA.FMGMP	IMAGE	1057	05/18/94 13:45:58	-	-	-	-	-	-	-	-	-
ADCMK.DATA.FMSC	IMAGE	1147	05/18/94 12:52:31	-	-	-	-	-	-	-	-	-
FMMF2.DATA.FMGMP	IMAGE	14831	05/18/94 14:58:12	-	-	-	-	-	-	-	-	-
FMMF2.DATA.FMSC	IMAGE	3058	05/18/94 13:04:19	-	-	-	-	-	-	-	-	-
FMPC.DATA.FMGMP	IMAGE	595	05/18/94 14:54:36	-	-	-	-	-	-	-	-	-
FMPC.DATA.FMSC	IMAGE	718	05/18/94 12:23:13	-	-	-	-	-	-	-	-	-

```
END OF PROGRAM
GCP_D:
```

Figure 4 presents the status of shadow file posting and indicates the number of updates performed on the shadow file since the posting process was started. We stop the posting process every morning to backup the shadow copy of the above databases so the number of updates represents total DBPUTs, DBDELETes, and DBUPDATES since about 5:30 AM that morning. The columns at right indicate the synchronization status and disposition of records should the database loose synchronization with the master copy. With a '-' in the SYN column there are no synchronization problems. There are other databases that have shadow copies on this system however they have not been updated thus far today so NetBase has not opened them.

An added benefit to having the ADC databases on the 927 system was that the Payroll system had easy access to employee time data stored there. Daily lookups and extracts of this information now occurs more efficiently. The jobs and on-line procedures that use the time data are simpler than before when they had to perform remote logons. End users and the programming staff noticed significant performance increases in this process since NetBase shadowing was introduced.

Note that with NetBase all updating must occur on the master system. The shadow copies of data can not, except in special circumstances with KSAM files, have updates passed back to the master system.

In the near future we will be using the copy of the ADC data on the 927 system to provide recovery system should the 987 system be down for any reason. A complete copy of the data collection software will be placed on the 927 along with procedures to quickly switch over to using that system. By running a recovery job the computer operator will be able to switch to the backup copy resulting in but a few minutes of downtime for the ADC system. When the 987 is back on-line we can choose a convenient time to take the ADC system down again to switch to using the 987 as the master system. This can be done hours or perhaps even days later depending on the processing demands. We hope to be able to use this method to permit software updates to critical systems like the 987 while keeping essential services running.

Many of our systems perform inquiries on the General Ledger database to validate transactions. This does not occur frequently and before NetBase there would have to be a remote logon to the 947 system each time. Now programs can access the General Ledger database as if it were local to the system saving time and effort when accessing it from another member of the cluster.

Printing to the system line printers has always been centralized to one of the computers in GRACO's cluster. Previously remote logons and remote file equates were used to access the printers. With NetBase and its companion product NBSpool we now automatically move spool files to the cluster's "print server." This is accomplished by a batch job that checks the spool queue for output and transfers it over the network to the server system. If the print server is unavailable, the output will wait on the originating computer's spooler queue until the server is on-line again. This has eliminated much complexity from our production batch jobs.

Along with "print servers" NetBase can support "batch processing servers" that allow batch work to be offloaded onto another system. This can be particularly useful when an application has many on-line users and frequent ad-hoc reports. By shadowing the necessary files to another system all reporting can be directed there freeing the CPU supporting the on-line programs from the burden of the ad-hoc reports. For example we could shift all batch reporting during the day to our 917LX improving the on-line response time on the 947 and or the 987. To date we have not created a "batch processing server" because we do not see significant problems with on-line response.

EDI will begin to make greater use of NetBase in the coming months at GRACO. Our retailer EDI partners are demanding more data from us and fast turn around for orders so there can be very little downtime. We have configured the 927 so that it can function as a backup for EDI by equipping it with the proper hardware and communications software.

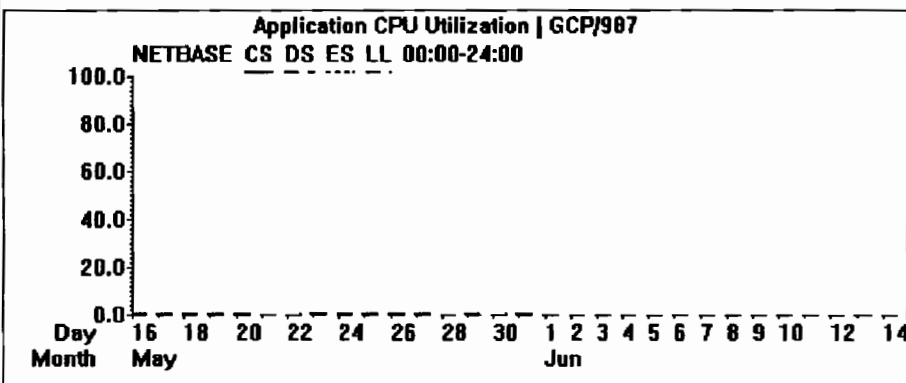
Our Customer Service system will be 'shadowed' to the 987 in the future to provide data protection and quick recovery for that system as well.

One point of concern while we were planning the shadowing and recovery of the systems on our cluster was the software licensing issue. Licenses for our tools and applications are for specific machines and we could be technically violating agreements by using the

software on another computer even in a disaster situation. We are pleased that all of the software vendors that this is an issue with have been very willing to permit us to use their software on other systems in an emergency situation or a temporary basis.

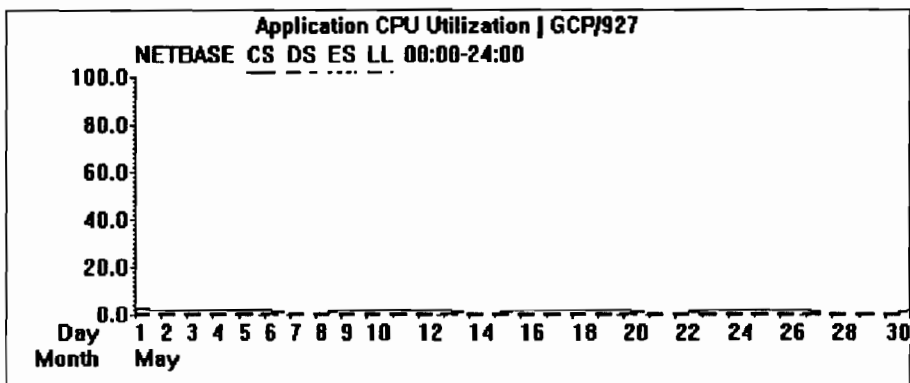
Figure 3 is a listing of the NetBase directory file entries on our 987 system. That and similar entries on the 927 system tell the software to make duplicate copies of all updates to those databases. We use HP GlancePlus and HP LaserRX to monitor performance on all of our HP-3000 9X7 systems and we have found that the overhead added is minimal. To date the NetBase product does not even consume 1% (Graph 2) of its resources on the 987. During peak transaction periods the 927 has about 2% of its resources used to accommodate the 40,000 database transactions generated on a typical day (Graph 3). We expect the transaction volume to more than double within the next three months without an impact on on-line performance on the 927 or the 987. Presently we are experiencing very acceptable on-line response from all of our systems.

Graph 2 May 16 through 31, 1994, on the HP3000 System 987.



NetBase consumes very little on the 987 system.

Graph 3 May, 1994, on the HP3000 Series 927.



NetBase does not overload the 927 even though over 50,000 database updates are passed to the system daily.

To the end-users of our HP-3000s there appears to be no difference in the system since the addition of NetBase. In fact they are in some cases less aware that the software is distributed over several systems. NetBase AutoRPM is used on the 947 and 927 systems to provide access to HPDesk on the 987. Previously, users would see and sometimes have to enter by hand, DSLINE and REMOTE HELLO commands to access HPDesk on those systems. Now it is a remote logon handled by NetBase and it appears to the user that their HPDesk resides on their system. Database shadowing and remote file access is completely transparent to the end user.

NetBase does cause you to buy more disk drives than you would just using a single system. In theory you could double your space requirement. In practice not every file or system on the HP-3000s make use of NetBase shadowing. With disk storage from HP running at about \$1.80 per megabyte or less for a 2 gigabyte expansion kit, adding disk space can be done affordably.

Figure 5 - Program and Logon Directory Entries

NetBase Directory Program [0.9.4] Update 9 (C) QUEST Software 1987

D> LISTL @.@:@

Node: GCPD Directory: NBD.DATA.NETBASE

-- LOCAL LOGON -- -- TO -- ----- REMOTE LOGON -----

User	Acct	Node	Logon String
@	PAYROLL	GCPA	NETBASE,NBGCPD.HP1,PUB
AMIE	PAYROLL	GCPA	AMIE,NBGCPD.HP1,PUB
DARLENE	PAYROLL	GCPA	DARLENE,NBGCPD.HP1,PUB
DOLLY	PAYROLL	GCPA	DOLLY,NBGCPD.HP1,PUB
JENNIFER	PAYROLL	GCPA	ENNIFER,NBGCPD.HP1,PUB
JEWEL	PAYROLL	GCPA	JEWEL,NBGCPD.HP1,PUB
JUDIE	PAYROLL	GCPA	JUDIE,NBGCPD.HP1,PUB
LAURA	PAYROLL	GCPA	LAURA,NBGCPD.HP1,PUB
MGR	PAYROLL	GCPA	NETBASE,NBGCPD.HP1,PUB
PAUL	PAYROLL	GCPA	PAUL,NBGCPD.HP1,PUB
REBECCA	PAYROLL	GCPA	REBECCA,NBGCPD.HP1,PUB
MANAGER	SYS	GCPA	NETBASE,NBGCPD.HP1,PUB

D>

The programming staff must still be aware of the location of files. Unintended updates to shadow copies of databases will cause problems when trying to use that technique to backup data. There was an instance here where a programmer could not find the current valid data dictionary for a system because it existed on another system and was accessed using NetBase Network File Access.

Figure 6 Shadow and Remote File Access

GCP_C:NBDIR.PUB.NETBASE

NetBase Directory Program [0.9.4] Update 9 (C) QUEST Software 1987

D> LISTF @.@.@

Node: GCPC Directory: NBD.DATA.NETBASE

-----LOCAL REFERENCE-----			-----REMOTE EQUIVALENT-----				O D F S P
File	Group	Account	File	Group	Account	Node	P S L Y R L
							E E A N O O
							N T G C T G
PRODCUST DATABASE	EDITRAN4	->	PRODCUST	PUB	FMGCP	GCPA	- - L - - -
				(Shadow)		<Local>	
FORMELIM DATA	FMGCP	->	--Same--			GCPA	- - - - -
COMBOQ	PUB	FMGCP	->	--Same--		GCPA	- - - - -
PRODUCTS	PUB	FMGCP	->	--Same--		GCPA	- - L - - -
				(Shadow)		<Local>	

D>

The directory entry for COMBOQ.PUB.FMGCP in Figure 6 directs all file I/O to be sent to the COMBOQ.PUB.FMGCP file on our 987SX system. This was done because we could then maintain one copy of the current dictionary on the system. Unfortunately, an old copy of the COMBOQ file remained on the 917 (GCPC) system. A LISTF command would show the old file but it would not show as being accessed even when our programmer 'knew' he was using it. Needless to say, this caused a bit of confusion. The above NetBase Directory entry was made but it was not communicated to all staff members. Communication and coordination between the programming and operations group is critical for successful implementation of NetBase. This situation could have resulted in real problems had the data dictionaries been different on the two systems.

FINAL THOUGHTS

Building a cluster of HP-3000 is not for everyone. Here are some guidelines to consider before moving away from a single system.

1. There are multiple applications to support. There must be a physical way to separate each application. If you run one or two tightly integrated applications on your HP-3000s there may not be a way to run pieces of the software on different computers efficiently.
2. At least some of the systems must be discrete with limited interfaces with other systems. For example, our Consumer Service system has limited links to the manufacturing system. Network file access, though vastly improved recently, is still not as efficient as local file access. Systems with frequent on-line data interchange should be located on the same computer if possible.
3. The Information Services staff must be able to handle multiple systems. There is extra administrative and technical support work such as operating systems updates that is required when supporting multiple computers. If the staff is small or lacks experience with networked systems, adding additional computers should be considered carefully.

Has GRACO really saved money using the 'cluster' approach? As an experiment we priced out a HP-3000 Series 992/200 system to replace our four 9X7 Series systems. Using list pricing as of May, 1994 we found that we are saving approximately \$15,000 annually in HP hardware and software maintenance. That figure takes into consideration the additional cost of maintenance for software required on each machine like NS/3000. On one of our third-party software products we are saving nearly \$10,000 per year in software support. A similar analysis of replacing our current 9X7 systems with a Series 987/200SX showed that the HP hardware and software costs would work out to be about even. Savings begin to accrue when the cost of software license fees is taken into account though the savings are not as significant. A great disadvantage of consolidating our cluster to a single 987/200SX is that we would be at the top end of the processor family with no real room for growth.

By intelligently licensing our software we have been able to save thousands of dollars. Compilers like our COBOL compiler are licensed on only one CPU. We have a full PowerHouse license on our 987 system only. The other 9X7 systems in our cluster have runtime with reporting only saving us tens of thousands of dollars. NSD's JobRescue is licensed on only three of our four systems. With NetBase we can move the \$STDLIST files we want included in that product to a system that has a license automatically. The

cluster member that does not have a NSD license has a lower volume of activity so the job listings can be managed from another computer without significant difficulty.

There is a point of diminishing returns when upgrading horizontally. At some point the cost of replicating software on multiple systems can become greater than the unit cost savings from the lower software tier. It may be possible to get the same throughput with six 947s as with one 992/200 but the cost of replicating software that many times could negate any unit cost savings. For example, NS3000/iX software license for six systems would cost between \$50,000 and \$114,000. Very much depends on how the applications software packages that will run on your computers is priced.

User-based software pricing can have a big impact on system purchasing decisions. HP has been using user-based pricing for some time with their system management and networking software products. Cognos has begun offering something similar. It is now possible to purchase a 5-user development pack for PowerHouse that will cost the same whether the software is loaded on a 917LX or a 995/800. If this pricing method becomes common place in the MPE/iX software market the advantage of having multiple small systems may become less justifiable from a cost point of view. However, it is very likely that hardware prices will continue to decrease so we can continue to look forward to having inexpensive yet powerful computers available.

Should your next HP-3000 system upgrade be accomplished by adding another system? As with many questions like this the answer is 'it depends.' With this article I have tried to show how clustering HP-3000s can be a viable option when it is necessary to add computing capacity. This approach has inherent strengths such as fault tolerance and the potential of cost savings. The primary weakness is more complex system management. When upgrading or replacing a system it is more important than ever to review the entire hardware and software environment before deciding on an upgrade plan. Depending on the application, adding an additional computer can be more cost effective than even a board-swap upgrade when all of the software and support costs are included.

PAPER NUMBER 8012
HELP! MY PC'S DOWN AND I CAN'T GET IT UP!

PEGGY A. BEALL
(606) 745-0704



Have you ever walked in on a Monday morning and heard those words? You haven't if you have established a help desk and staffed it with efficient personnel trained to handle initial calls and sometimes very simple problems.

After all the long decision making, hair-pulling hours spent designing, installing, and implementing your network, you want to keep it running smoothly. A well-designed help desk should be the liaison between end users and MIS/network personnel.

The help desk is the first line of contact with MIS when an end user is confronted with a problem they can't handle. It may be as simple as "Where do I plug it in?" or as complex as, "I was typing and everything went dead." If it is a simple question the help desk operator can handle the situation without interrupting network personnel. As the operator gains more experience and knowledge of the most common types of problems occurring, she/he becomes better equipped at handling calls without asking for assistance.

The duties and responsibilities of help desk personnel are varied and some basic duties will be discussed here. The main duty, or the first duty, is to answer the telephones and/or "greet" users as they make their initial contact with MIS personnel when confronted with a problem.

Keep in mind that when a user finally calls the help desk the majority of them are experiencing frustration which may be taken out on the first person they talk with. The operator must remain courteous, professional, sympathetic, and show understanding at all times. They must obtain basic information from each and every caller, which often causes more frustration.

Help! My PC's Down And I Can't Get It Up!
8012-1

Information needed must include the user's name, department, phone number, and exact details of the problem. The operator must determine what the problem is, when it started, does it occur continuously or intermittently, which piece of computer equipment is affected and how, and exactly what the user was doing when the problem occurred. Which software application was being used at that time, and what steps the user tried to fix the problem before calling the help desk must be found out.

In some instances, the problem may be a recurring one belonging to the same user or a variety of users that the operator has documented. If this is the case, the problem may be fixed in a matter of minutes as a result of the operator looking up the problem and resolution and "talking" the user through it or sending another operator to the user's pc. As you may have guessed documentation is an important part of the help desk and will be discussed later in this paper.

During the initial contact with the user, the operator should be recording all details of the call. Along with the basic information discussed earlier, this record should also include the date and time of call, a call number, and who it is being assigned to. The user should be given the call number before hanging up the phone. This will enable the operator to find the call faster if it is assigned to another section and the user calls back.

Most MIS departments are broken into two or more sections. These sections may be networking, programming, training, and operations (which maintains the help desk). When taking the initial call it is up to the operator to determine if the call is to be assigned as a network problem, an internal programming problem, a training problem, or if it will remain with the operations section.

Sometimes the problem may be one the operator has taken so many times she/he knows the solution to it and can talk the user through

Help! My PC's Down And I Can't Get It UP!

8012-2

it, or can simply assign another operator to fix it. If the user can be talked through it, and can actually fix the problem, she/he feels a sense of accomplishment and obtains a better understanding of how his computer works. This approach results in greater user satisfaction.

The help desk is also responsible for keeping track of all open and closed calls. As stated earlier, each call is assigned a number and given to the user. If the user feels his call is becoming a higher priority than originally thought, he can use this number to call back and find the status. The operator will call up the number and report the status to the user. If the call shows no activity has occurred, the operator should first talk with the person assigned to the call to determine the progress. If the operator cannot get a satisfactory answer she/he should report the problem to the operations supervisor who in turn talks with the network administrator.

Keeping track of the closed calls is also valuable in helping the operator solve similar problems, as stated before. Some companies charge each department for work done inside the company. Closed calls keep track of time spent and helps in determining charges.

The help desk personnel are ultimately responsible for ensuring user satisfaction. They make initial contact with users, keep users updated on the status of the call, and follows up after the call is completed, thus making sure the user received the service required.

The help desk/operations section of MIS should also be responsible for keeping track of computer inventory; service and maintenance contacts; and software licenses. A good software package can be a tremendous help in this area.

Normal maintenance and installation are part of the help desk/operations section. This includes cleaning printers, changing ribbons (when users plead ignorance), initial troubleshooting of printers/pc's when unknown

Help! My PC's Down And I Can't Get It UP!
8012-3

problems pop-up; and installing pc's and printers.

The ideal help desk/operations section should have at least three (3) operators working each shift. One operator to go out "in the field" to perform maintenance and troubleshoot calls; one operator to answer the phones; with the third operator being responsible for the software licenses, contracts, tracking computer equipment (company-wide), and serving as a back-up to the other two operators.

As you may have guessed, selecting personnel to fill these three (3) positions should be done with care. You not only have to consider their background experience/education, but also their attitude. Notice I said attitude, not personality. These three people must be willing and able to handle an extreme amount of stress. They have to juggle upset users and sometimes equally upset MIS personnel. The operators must show good judgement, patience, understanding, sympathy, and an aptitude to learn continuously. They must be very detail oriented! These operators must be team players and willing to go the extra mile to represent MIS.

Once chosen, the operators should be trained on the basics of network hardware and techniques. Even if the operator does not know how to solve the problem, she/he will at least know what to ask and understand what the user is trying to describe.

The operators should be given a very basic course on each software application used on the network. This enables the operator to determine if the users problem is caused by user error, hardware difficulty, or the software itself.

As new applications and equipment are added on to the network, it is necessary that operations/help desk be included in knowing what it is, how it works, and when it will be put into place.

Although user training is not a function of the help desk, it should not be overlooked.

Help! My PC's Down And I Can't Get It UP!

Training users in just the basics of pc's and printers should prevent unnecessary calls. Software application training classes or self-taught videos should be made available to users. You would be surprised at the number of users calling to find out how to use "bold" in WordPerfect, for example.

When the network is first implemented, it is suggested that classes be started at that time. If you start them too early, before they have pc's, they will forget what they've learned and will be calling for a refresher course. If they have the pc, they will go back and start playing around with them, so watch for help desk calls.

A valuable tool of the help desk is the type of software specifically used to track and record calls. It is possible to use a "paper log", but it would be inefficient to go that route when you have a great network backing you up.

An evaluation of your requirements should be done to determine which application best suits your needs. Most companies will be happy to send you a demo to help you decide. The best thing to do before buying one is to look at and try several. You may even decide one of your programmers can write an "in-house" application that will better suit your needs.

The ideal package will automatically assign the call number, date and time stamp it. Sufficient room is needed for a problem/resolution description. You may decide you would like to have a section that would call up all calls recorded by a single word, such as "printer" or "software". The ideal application will also be able to record and track inventory.

An application of this type may take a while to set up, especially if you chose to include user information, but the results are well worth it in the long run. If the inventory portion of the software is kept up-to-date, imagine the tremendous amount of time saved when it comes time for the annual inventory.

Of course, if your users move workstations around a lot, a rule should be strictly enforced that MIS must be notified before the move takes place. This enables the help desk to keep track of the equipment, and also determine if additional lines must be dropped or any special adjustments done before the move is made.

After choosing the right personnel and software, don't forget about the documentation. Careful and complete documentation of each type of trouble call and it's resolution is advised. If a few precious minutes are taken to log the problem and the steps taken to resolve it, the next time a problem of this nature arises it may be handled over the phone by help desk personnel or network personnel quickly instead of using valuable time researching the problem again.

Here again, a good software application can be used which contains the single word call up. Each step taken to determine the resolution should be noted. Chances are another user will have the same problem at some time or other. If the problem and solution are documented so anyone can resolve it, valuable time and resources will be saved.

Documentation of this sort also proves useful in determining specific training needed for different users or departments. If the right training is given to users, problems should be cut tremendously.

You may find that as each week ends it would be beneficial for a report to be run which selects the open calls on the Help Desk. This report should be evaluated by the operations supervisor to determine if the calls are receiving the required attention. If necessary, the supervisor should contact any personnel involved to try and resolve the "problem" calls. It may be necessary to set-up a system that would place on-going calls that are hard to resolve or that needs more attention on a list of tasks designated as projects. This would remove the call from the Help Desk and assure the user that it is now getting a higher priority of attention.

SCREEN EXAMPLE

CALL # _____
TIME _____

DATE _____
OPERATOR _____

USER _____ DEPT _____
PHONE _____

ASSIGNED TO _____ SECTION _____
STATUS _____

KEYWORD _____ DATE CLOSED _____
TIME SPENT _____

PROBLEM _____

RESOLUTION _____

USER NOTIFIED _____

DEPT. CHARGED _____

Help! My PC's Down And I Can't Get It UP!
8012-7

An example of one screen on a software application may look like the preceding page. This screen includes all necessary information required for most calls. You may want more information or less, depending on your company's requirements.

As the Help Desk Operators learn and perform their daily tasks, the other MIS staff will soon learn to appreciate them, especially if you decide to let some of the other section members man the Help Desk periodically to see what it's like. All in all, if you plan and establish your Help Desk to fit your company's requirements, it will soon prove to be one of the most valuable assets belonging to MIS.

In closing, be sure to let your operators know every now and then just how much you do appreciate them.

8015
Prepare Your MPE/iX System Before Updating to 5.0
Al Dulaney
Hewlett-Packard
(919) 460-2326

Each new release of the operating system requires additional disk space. The MPE/iX 5.0 (C.50.00) release which includes Hierarchical File System (HFS), POSIX interface, IMAGESQL and enhanced Operating System is no exception. As System Managers, we can identify and remove nonessential system files, utilities, and databases that are not part of the production environment prior to updating. Any downtime is critical to a production environment, and it must be held to a minimum. The successful update to the latest operating system and minimizing the downtime stands out as a critical operation. Careful planning and implementation is essential to performing this task.

As a System Support Engineer (SSE), I help assist customers with the update process. Each HP 3000 I update is different. Most, if not all, customers are in the process of consolidation of their systems. This action has dictated higher availability and shorter planned downtime during the update process. The update must be done in a timely and professional manner. I approach every update as if it were my system.

The window of availability to update your production system is becoming increasingly smaller. Planning for this event is the key to minimizing the disruption of the update process to your production environment. We must review in advance the update materials, available dates, people resources, HP software products, 3RD Party software products and hardware requirements prior to the update process.

The update material received with your tapes should be reviewed prior to the update. It outlines any exceptions in the "Read Me First" section. These are not the only sources that need to be reviewed. Read the update material carefully. Hewlett-Packard cannot guarantee that you will be successful if you try to update your system to Release 5.0 from an unsupported release. Significant changes have been made to the operating system which may not be compatible with unsupported releases.

Discuss with your production schedulers the dates that the system will be available for a full backup and update. Are there any other upgrade or consolidation activities planned around the update to the computing facility? These activities should be reviewed to see which ones can be done in parallel and which ones need the longest lead time. A typical update normally takes 4-5 hours, not counting the

Prepare Your MPE/iX System Before Updating to 5.0

backup time. I recommend to update a test system or development, first to minimize any possible tape problem, missing product or documentation error if possible.

Patch information can be obtained from the HP SupportLine, PowerPatch, and RC Engineer. If you want to review them yourself, a HP SupportLink can be used from your own terminal or PC with modem and a regular phone line. Other circumstances may dictate that you will have to update before the PowerPatch tape is available. A PowerPatch tape is usually produced every three months after manufacturing release. Response Center Advocates, (RCA), if your support level is Personalize System Support (PSS), familiar with your system environment should be notified prior to updating to discuss known installation problems and patches. Otherwise the on-line RC engineer can help you review current problems. RCA patch review and consolidation from your current release to the 5.0 release should be planned. You may want to wait until the first PowerPatch is available.

Third Party Vendors will certify their products on the latest release. You should notify them of your current product version and intention to update to the planned release. They will confirm the status of your version and it's supportability. If not, you should plan to obtain their latest version and plan accordingly the migration/update to that version.

Your Customer Engineer should be consulted regarding the update window and any hardware items that may be of concern. Adding additional disk or disk placement could coincide with the update, if time permits. We are trying to minimize the number of things that could go wrong.

This checklist is long and is intended to review all possible exceptions, as I see them. It does not imply that your installation will experience any or all of them. Knowledge and preparation for these possible events will minimize the overall planned downtime and contribute to a successful update from MPE/iX Release 4.0 or 4.5 to MPE/iX Release 5.0.

An assessment of your current operating environment and review is very important. Over the next hour we will review the following areas:

- o MPE/iX Operating System and configuration parameters
- o Software patches installed
- o HP Manufacturing application groups and accounts
- o HP Networking subsystems and products (Host/PC based OVDTCMGR)
- o HP subsystems, compilers and utilities
- o HP Office Automation and configuration parameters
- o Hardware configuration parameters
- o 3RD Party application software

Prepare Your MPE/iX System Before Updating to 5.0

- o Identify, and remove any duplicate utilities from the system.
- o Identify all TurboIMAGE databases and remove those that are no longer needed.
- o UDCs that are over-ridden or duplicates

Operating System and configuration parameters should be reviewed within SYSGEN. If possible, prior to the update turn off any excess system logging events. This will reduce the total number of system logging files and size created. After the update re-activate any logging events that you previously had turned on.

Within the MISC configuration review SESSION and JOB limits settings. Review with the customer the HPUSERLIMIT setting and its implications. If the HPUSERLIMIT is 64 and session/job current setting is default of 60, review with customer how to increase to maximum. Changed setting will take affect at next reboot.

Within the MISC configuration review RIN settings used by applications. Use the LRIN command within SYSGEN. In the early beta release of 5.0 these rins were erroneously removed during the update. This should not be a problem with the PULL release, but review settings just in case.

Review READY output spoolfiles as these files are candidates to be stored to tape and purged for temporary space requirements for the update. Appropriate care must be taken to ensure that the appropriate amount of contiguous space is obtainable on LDEV1. Any HP or user accounts, MPE logging files, LOG####.PUB.SYS and EDITOR/3000 temporary work files, K#####.@@, need to be reviewed as candidates to be stored and purged for temporary space requirements. Use CONTIGXL.MPEXL.TELESUP to locate appropriate space. Once 5.0 has been updated use VOLUTIL with the SHOWUSAGE command, as CONTIGXL is no longer correct. Disk drives with a capacity of less than 500 MB (HP7933, HP7935) can no longer be used as LDEV1 on release 5.0. LDEV1 must have a capacity of at least 500 MB (i.e. an HP7937 or larger). It is important to note that only the following disks are supported as LDEV1 due to the disk space requirements of MPE/iX: all newer SCSI disks, HP-FL versions of C22X drives, HP-IB C2202(3)A, and HP7937. Older 7933 and 7035 drives are not supported as LDEV1.

Software patches applied in the last few months should be reviewed with the RC engineer to verify if they have been fixed in 5.0 or will need a 5.0 version applied prior to the production system update. How many PowerPatch Tape have been applied? PowerPatch B.40.08 cover letter and addendum discuss a potential problem, that only affects users whose systems contain no memory controller/arrays larger than 8 Mbytes AND whose NL.PUB.SYS essentially

Prepare Your MPE/iX System Before Updating to 5.0

contains all products or has been heavily patched with Relinker patches. The systems that COULD have ONLY 8 MB memory controllers include: Series 922/922/LX/922RX, 925, 930, 935, 920, 948, 949 AND 958. This limitation is removed starting with Release 5.0. If you have the above configuration you need to discuss the workarounds with a Response Center Engineer prior to updating.

HP Manufacturing application and tools need to be checked to see if the version you are currently running is supported with Release 5.0. Updating to the supported versions may need to be scheduled and performed prior to the update. This would avoid the possibility of backdating.

HP Networking subsystems and products should be reviewed prior to updating. Remove any excess dump or trace files from PUB.SYS. TRM####.PUB.SYS are port dump files created by TERMDISM. S####@.PUB.SYS are box dump files created by TERMDISM. NMTRC####.PUB.SYS and NETDMPn.PUB.SYS are communication trace files created by network tracing. These files are not large in size but should be removed after the problem is corrected. These files may not contribute to a large amount of space taken, but every bit of disc space needs to be returned to the available disk space pool. This cleanup will also contribute to fewer files for your daily and weekly backups.

NMMGR can be run in batch, and the output listing can be useful in documentation of your NMCONFIG.PUB.SYS file, in the event you have to recreate it, and in update planning. Review output listing from NMMGR to verify type of DTC manager, Host-based or PC-based. If PC-based, steps must be taken to verify the current version of the PC product and its supportability with 5.0. In addition, if it is not at the latest supported level, it would be a best-practice to bring it up to the currently supported level in a planned manner prior to the update. This would ensure that if any problems arose, the Response Center can assist in a timely fashion. Review output from NMAINT command to verify data communication subsystems present and version ids.

HP subsystems, compilers and utilities should be reviewed prior to updating. TurboIMAGE upgrade to IMAGESQL is based on your current user limit. To view your current limit, type SHOWVAR HPUSERLIMIT. You are eligible for a free upgrade of the product and the support cost is based on the user limit which is added to you monthly support contract. If you are not aware of the free HP IMAGE/SQL upgrade offer, contact Support Direct Marketing at 1-800-437-9140 for further details. Review TurboIMAGE logging status and identifiers with the SHOWLOGSTATUS and LISTLOG commands. Review TurboIMAGE 3RD Party Indexing products using QUERY.PUB.SYS with the VER command. Review HP ALLBASE/SQL versions using the SQLVER command. If the customer is an active SQL user, then migration of existing DBE will need to take

Prepare Your MPE/iX System Before Updating to 5.0

place after the update prior to user access. Any site-specific patches would need to be verified by the Response Center to see if applicable to the SQL G.0 version. All 3RD Party vendor tools would need to be notified prior to update, requesting status of their products with 5.0.

Office Automation configuration parameters and status need to be reviewed if present. Review and discuss Open Desk Manager's new features. HP DeskManager has been renamed HP Open Desk. The upgrade product would work on 4.0 or 5.0. If you are a current HP DeskManager, you may want to review the features of the upgraded product and take advantage of them as soon as possible.

Hardware configuration parameters need to be verified for memory and disk configuration. SYSDIAG is used to verify memory configuration. SYSDIAG is used to verify I/O configuration. SYSDIAG is used to verify firmware version of DAT DDS tape drives. PSCONFIG.PRED.SYS is used to verify that predictive support is configured, current and running properly. With 5.0 the online diagnostics will introduce licensing of our diagnostics to MPE/iX systems, and they will be password protected. Passwords will be implemented only on hardware troubleshooting diagnostics. The typical user with HP support will find that their needs are met by support utilities which will not be password protected.

All 3RD Party application software needs to be reviewed. Identify all 3RD Party vendor products used by your installation. This should not be a problem, but System Managers do change and the current System Manager might not be aware of the total number and currently used 3RD Party products. Notify each vendor of your intention to update and the status of their product on 5.0. Verify the current version of their product. Updating or migration may need to be performed prior to updating to 5.0. Obtain latest version for testing, and convert prior to update to 5.0. Test production run on test account prior to move to production.

Identify, and remove any duplicate utilities from the system, utilities that are currently supported by the Response Center or those that have been superseded by MPE commands. Here is an opportunity to gain control of the use of out-of-date or unauthorized utilities prior to the introduction of new POSIX utilities.

Identify all HP TurboIMAGE databases and remove those that are no longer needed, as disk spaces is a premium. Test databases and demo databases should be identified and removed from your production systems. Review all production databases for current flag settings and any customized buffer settings. CKIMAGE command file was created to assist in the search for these databases. (see command file figure 2)

Prepare Your MPE/iX System Before Updating to 5.0

Review for duplicate or overridden UDCs. Some system managers will make an MPE command inoperative by placing a UDC to deactivate the command, an example is PASSWORD. Some 3RD Party vendors and System Managers have UDCs that need to be reviewed with MPE/iX 5.0 commands, such as MAKE, TOUCH.

These steps alone will not ensure success in updating to the next operating system, but will result in additional free space in preparation of the update by removing any excess files from the system. (see activity checklist figure 1)

The article "A Programmers Look at MPE/iX, Managing a POSIX HP 3000 System" in the Jan '94 issue of INTEREX discusses the evolution of POSIX on MPE/iX and the new demands placed on system managers. This article goes into great detail regarding the HFS, POSIX, and security concerns with 5.0. I do not wish to repeat them here, but strongly recommend you, the system manager, review this article.

More attention should be given to the security of your systems. Today the System Manager is busy managing PC LANs and UNIX servers, plus managing a production HP 3000. Time given to review basic security conventions usually isn't done until it is too late. MPE/iX 5.0 introduces a new develop environment, consisting of UNIX file security at the file level. HFS file naming conventions consisting of upper and lower-case file names. Periodically, you should review all accounts with SM (System Manager), OP (System Supervisor), and PM (Privileged Mode) capability to see if the requirement is still needed. If that capability is still needed, make sure they are passworded and the password changes on a regular basis. The new environment enables any user with SM capability to have complete ROOT or Super User capability. You had this same concern with MPE/iX, but let's review the current capabilities before adding new ones. In the UNIX environment there is only one SU (Super User) which is heavily guarded with passwords. My is this important, well, if you have SU capability and are unfamiliar with UNIX commands, you could purge files and directories that you didn't plan to. The commands assume you are a knowledgeable user and you have the permission to do this, so it executes the commands as requested. MPE prompts you to verify if you really want to purge PUB.SYS. It gives you a second chance, and until you need the power of Open Systems and the POSIX .2 Shell and utilities care must be taken to guard against "Good Intentions".

Backdating your system is not something you care to plan for, but in the event of some unforeseen problem, this may need to take place. The procedures are outlined in Appendix F "Backdating your System" in the "HP 3000 MPE/iX Installation, Update, and Add-On Manual MPE/iX Release 5.0 (Limited)". This appendix describes general backdating methods that will work in most cases. To

Prepare Your MPE/iX System Before Updating to 5.0

learn which method is optimal for your system and your situation, contact the Response Center before beginning to backdate your system.

Careful reading of the following material prior to the update is equally important:

- 1) Communicator 3000 MPE/iX Release 5.0 (Core Software Release X.50.20)
- 2) New Features and Functions of MPE/iX 5.0 (Software Release Planning class notes)
- 3) HP 3000 MPE/iX Installation, Update, and Add-On Manual MPE/iX Release 5.0
- 4) Related INTEREX articles (i.e. "A Programmers Look at MPE/iX, Managing a POSIX HP 3000 System", by Jeff Vance Jan. 94)

You should plan to attend the scheduled Software Release Planning session for MPE/iX 5.0 at your local Education Center prior to your planned update. At this session the instructor will discuss the MPE/iX 5.0 release strategy and new features included in 5.0 in greater detail.

Customers running on versions prior to 4.0 must update first to MPE/iX 4.0 and then to 5.0. All other update paths are unsupported and thus, discouraged. Based upon the review of your system and SRP training, updating your system to the currently supported versions of the following products may be necessary, prior to 5.0: Operating System, manufacturing applications, OpenView PC applications, 3RD Party software, and appropriate diagnostic patches.

This effort will further minimize the amount of downtime experienced during the update process. It will prepare a cleaner environment for the POSIX interface and HFS directory structure. We can now devote our limited resources to learning the new POSIX.2 shell commands and utilities. Cleaning up your current MPE/iX environment and file system before the development of any POSIX applications is important.

I am an advocate of taking a few hours monthly and reviewing the status of your system. Control at this level will help you better prepare for the next Operating System (OS) in a calm and peaceful manner. Instead of a panic, no pun intended, manner when you are trying to add disk, new CPU, and operating system in the same weekend.

Finally, careful planning and implementing of these steps will result in the smooth update to MPE/iX (C.50.00).

(CleanUp + Planning + More Planning + SRP) = Smooth Update

Prepare Your MPE/iX System Before Updating to 5.0

Activity Checklist Prior to Updating to 5.0

- MPE/iX Release 5.0 Software Release Planning (SRP) session (POSIX and HFS Overview similar to 4.5 Seminar)
 - Identify HP location near you
 - Who should attend?

The New Features and Functions of MPE/iX 5.0 is designed for experienced System Managers, System Operators, Programmers, and application developers who have working knowledge of an HP 3000 Series 900. This course is designed to give them the information they need to make a smooth transition to MPE/iX 5.0. A major portion of the course will focus on the new Hierarchical File Structure (HFS) and the MPE/iX Command enhancements.

Programmers will benefit from the information contained in this course, but should be aware that this is an overview seminar. Follow-on training is probably necessary.
 - Attend seminar ___/___/___
 - Read all materials

- Communicator 3000 MPE/iX 5.0 (Core Software Release X.50.20)
 - Read first to see all the new improvements
 - Read second for specific products
 - Read third for MPE/iX command enhancements

- HFS/POSIX/ACDs training:
 - Read Communicator Section 5, POSIX/Open Solutions
 - Complete POSIX Computer Based Training (CBT)

CBT comes with 5.0, but can be obtained for 4.0

 - System/Alternate Manager review
 - Database Administrators and programmers
 - Other sources:
 - "Getting Started with MPE/iX" P/N 32650-90351
 - INTEREX articles:
 - "A Programmers Look at MPE/iX, Managing a POSIX HP 3000" Jan. 94

- Review 5.0 Backdating procedures
 - Appendix F, "Backdating Your System" in HP 3000 MPE/iX Installation, Update, and Add-On Manual MPE/iX Release 5.0 (Limited)
 - Method 1, if you have a resent CSLT and complete backup

Prepare Your MPE/iX System Before Updating to 5.0

- Create new group in SYS, named BULDACCT
- CHGROUP BULDACCT (to new group)
- BULDACCT, will create two job streams containing the commands to rebuild your current accounting structure
 - BULDJOB1-job stream to recreate all accounts and user volume sets
 - BULDJOB2-job stream to setcatalog all your UDC files after you have restored all files

Note: Best-Practice to recreate files once a month

- CSLT, create and label Custom System Load Tape of current version
 - :SYSGEN
 - >TAPE
 - >EXIT
- Store and label STORE @.@.SYS prior to UPDATE
 - STORE @.@.SYS;*T;DIRECTORY;SHOW=OFFLINE
- Full backup with directory option
 - STORE @.@.SYS,@.@.@.@.SYS;*T;DIRECTORY;SHOW=OFFLINE
- Label and store tapes in computer facility
 - Keep close at hand during the update process
- Method 2, if you do not have CSLT and full backup
 - Use SLT, FOS, and SUBSYS tapes from the earlier release

Review IMAGESQL Upgrades

HP TurboIMAGE/SQL upgrade at no charge with a corresponding support price increase to their software support agreement.

- Identify and correct before free offer expires
- Upgrade 36391A TurboIMAGE with 36385B IMAGE/SQL Upgrade
 - Select option for user limits
 - Opt. UA3 for use by 1 to 8 users
 - Opt. 0AF for use by 1 to 20 users
 - Opt. UA7 for use by 1 to 32 users
 - Opt. UCY for use by 1 to 40 users
 - Opt. UA9 for use by 1 to 64 users
 - Opt. UBD for use by 1 to 100 users
 - Opt. UAB for use by 1 to 128 users
 - Opt. UCN for use by 1 to 160 users
 - Opt. UAD for use by 1 to 256 users
 - Opt. UDW for use by 1 to 384 users
 - Opt. UAT for use by unlimited users
- Verify product and proper user limit is on packing slip
- HP ALLBASE/SQL Training is available on your system:

Prepare Your MPE/iX System Before Updating to 5.0

- "Up and Running with ALLBASE/SQL" P/N 36389-90011
- LISTFILE @.SAMPLEDB.SYS,2 (if present)
- PRINT CREASQL.SAMPLEDB.SYS (follow instructions)

- Review HP products currently installed
 - Rel 4.0 and older
 - SWINVXL.SWINV.TELESUP
 - Rel 4.5 and later run
 - PSIRPT.PRED.SYS
 - Contact Support Contract, CE or SE, if not correct
- HP ships Pull/Push release to customer
 - Verify products with SOFTREP procedure

Prior to installation of a customized SUBSYS tape, you may want to determine which products and files are located on the tape. There is information available on CUSTOMIZED HP 3000 SUBSYS tapes which helps the field representative or customer to determine which products are physically located on a tape.

Each customized tape that SRDO produces contains a T File (EXCEPT for customized SUBSYS tapes created for the NO CHARGE or HOT order process). This T File contains a listing of products stored to the customer's tape, plus the customer's system-handle. By restoring this file from the tape to the system and viewing it with an editor, you can determine what products are actually on the tape.

Instructions for Returning the Customer File to the System

At the prompt type:

```
:FILE T;DEV=TAPE
:RESTORE *T;T#####.PROD.SOFTREP;CREATE;SHOW
```

Go into EDITOR, text the file and list on the screen.

If a printout of the file is desired, issue the following command:

```
:FCOPY FROM=T#####.PROD.SOFTREP;TO=*LP;NORECNUM;CHAR
```

NOTE: The file size is 256B wide and the screen is 80B wide, therefore the lines will wrap around on the screen.

This is a fast and easy way to determine if your tape has been created with the

Prepare Your MPE/iX System Before Updating to 5.0

correct products. Additionally, results of the tape status are quickly obtained.

Corrections needed

If you determine that a tape is actually defective (missing or incorrect products on the tape), please contact your support contracts coordinator.

IMAGESQL user limit support option

Media option

Missing products

Contact Support Contract, if products are missing

CHECKSLT.MPEXL.TELESUP

Verify space requirements with CHECKSLT, Option 7

CHECKSLT is a non-privileged program in the group MPEXL.TELESUP on all MPE/iX systems. It reads and checks System Load Tapes (SLTs). With option 7 it will check SLT's tapes for media errors and estimates required contiguous disk space requirements. Once you have identified the amount of space required, you can build a file "AXLDEV1.PUB.SYS" days before the actual update.

Review HP 3000 MPE/iX Installation, Update, and Add-On Manual MPE/iX Release 5.0 (Limited)

Select appropriate section

Read Me First

Updating MPE/iX

Adding On Purchased Products - SUBSYS Only

Installing MPE/iX

Manually Installed Products

Diagnostic Passworded on 5.0

Review impact at this location

What SYSDIAG utilities are the customer using

LOGTOOL, SYSMAP, TERMDSM, VERIFY, CONMSG, IOMAP, CLKUTIL, DISCUTIL are not passworded

Discuss options, if concerns remain

Diagnostic Password Protection Guidelines

Diagnostics License Agreement, should accompany update material

MPE/iX Operating System (OS) Configuration:

SYSGEN:

Review Logging settings

Review Job and Session LIMIT settings

Prepare Your MPE/iX System Before Updating to 5.0

- Review Global RINS (HP DeskManager)
- Review AutoRestart configuration if present

SPOOLING:

- Total ready spoolfiles value
- Review/store/purge to obtain space
- LISTSPF @,STATUS

Volume Management: (Review if present)

- VSUSERS
- DSTAT ALL

Current # JOBS/SESSIONS activity

- SHOWJOB JOB=@S
- SHOWPROC JOB=@S
- SHOWJOB JOB=@J
- SHOWPROC JOB=@J

MPE Logging:

- Review/store/purge to obtain space
- LISTFILE LOG####.PUB.SYS,2

EDITOR/3000 temporary files:

- Review/purge to obtain space
- LISTFILE K#####.@.@,2

CONMSG review:

- Print and review for error messages
- RUN CONMSG.MPEXL.TELESUP

This program prints out a disk file containing the last BOOTUP commands. It should be reviewed for errors. Take what action is needed to correct.

HP Accounts review:

See Communicator for list of HP Accounts

- Report and review if any accounts can be purged for space

MPE/iX Current Patches:

- PRINT HPSWINFO.PUB.SYS
- PowerPatch Tape previously installed?
- Review patches to see if additional needed
- Cleanup/purge files for disk space:

Prepare Your MPE/iX System Before Updating to 5.0

- LISTFILE PATCHAUD.PUB.SYS, and purge if present
 - LISTFILE HPINSTFL.PUB.SYS, and purge if present
 - LISTFILE AUTOHIST.PUB.SYS, and purge if present
 - CONTIGXL prior to update from 4.0 or 4.5 to 5.0
 - CONTIGXL.MPEXL.TELESUP "-d1 -c6000"
 - After install or update to 5.0 use VOLUTIL
 - VOLUTIL
 - SHOWUSAGE 1 60000 nonrestrict summary ; perm ;free
 - EXIT
 - Review permanent space currently on LDEVI
 - DISCFREE B,1
 - Review permanent and transient space allocation
 - DISCFREE C
- HP Manufacturing Products:
- Present, what version? else next step
 - Review supported versions
 - Obtain latest versions if needed
 - Application User exits?
 - Update to latest version prior to update of OS
- HP Networking Applications:
- Review current network configuration with NMMGR
 - !FILE NMMGRCMD=\$STDINX
 - !FILE FORMLIST=\$STDLIST
 - !RUN NMMGR.PUB.SYS
 - OPENCONF NMCONFIG.PUB.SYS
 - SUMMARYCONF ALL
 - EXIT
 - !RESET NMMGRCMD
- Cleanup/purge datacomm trace files in PUB.SYS for space:
- LISTFILE TRM####.PUB.SYS,2
 - LISTFILE S####.PUB.SYS,2
 - LISTFILE NMLG####.PUB.SYS,2
 - LISTFILE NMTC####.PUB.SYS,2
 - LISTFILE NETDMPnn.PUB.SYS,2
- OpenView DTC Manager:
- Present, what version? else next step
 - HP OpenView DTC Manager
 - ABOUT under OVRUN MAP
 - CHECKVER under MSDOS

Prepare Your MPE/iX System Before Updating to 5.0

- MS-DOS
 - Enter VER at prompt
- MS-Windows
 - Click on "ABOUT" in Windows Program Manager Help menu
- HP OpenView Windows
 - Click on "ABOUT" in OVRun's Map menu
- ARPA Services
 - what netsetup.exe (B.03.00)
- Obtain currently supported version
 - From whom?
 - Media type 3 1/2 and 5 1/4
- Update PC to supported configuration before update
- NMMGR DTC 5.0 enhancements
 - Automatic System Cross-validation with NMMGR
 - Support of 4649 Terminal I/O Devices
 - 5.0 now supports 1000 device classes
- Review NMMGR output for following products:
 - DTC 72MX (P/N J2070A)\
 - DTC 16MX (P/N J2062A) +-> FLASH EEPROM
 - DTC 16iX (P/N J2063A)
 - DTC 48 (P/N 2345A)
 - DTC 16 (2340A)
- Review data communication products present
 - NMMMAINT
- HP Sub-Systems:
 - TurboIMAGE:
 - SHOWLOGSTATUS command
 - LISTLOG command
 - RUN QUERY.PUB.SYS, enter VER
 - RUN DBDRIVER.PUB.SYS,VER
 - ALLBASE/SQL version and patch level
 - SQLVER
 - Follow SQLMIG migration procedures
 - VPLUS Intrinsic:
 - RUN HP32209S.PUB.SYS
 - VPLUS control file modified?
 - LISTFILE VENCNTL.PUB.SYS,2 if present
 - COBOL control file modified?
 - LISTFILE COBCNTL.PUB.SYS,2 if present
- HP Office Automation:

Prepare Your MPE/iX System Before Updating to 5.0

- HP Open DeskManager version check
 - run `dmver.deskmon.sys;lib=g`
 - RINs used?

- Hardware Configuration:
 - SYSDIAG
 - Memory configuration
 - 8Mbytes (small system configuration (Y/N))
 - LDEV1 Disk configuration:
 - Refer to the article "HP7933s and HP7935s as LDEV1" in the Communicator 3000 MPE/iX Release 5.0.
 - LDEV1 HP 7933 or HP 7935 (Y/N)
 - DDS dat version for firmware check:
 - Refer to the article "Autoinstall and DDS Firmware Problem" in the Communicator 3000 MPE/iX 5.0.
 - SYSDIAG
 - DUI> `scsidds;ldev-DDS-LDEV;section=50`
 - SCSIDDS> `rev`
 - the firmware revision is displayed
 - SCSIDDS> `exit`
 - Predictive configuration
 - Run PSCONFIG, review status

- 3RD Party Applications:
 - Identify 3RD party products
 - Notify 3RD party your intention to update 5.0
 - Verify w/3RD party products supported on 5.0
 - Verify w/3RD party any exceptions and patches needed
 - Obtain latest release product from 3RD party
 - Update/Convert to latest release if required
 - Test suite

- Check TurboIMAGE databases
 - Command file searches system for TurboIMAGE databases, and list their current flag settings for review. Opportunity to obtain additional disk space and remove any duplicate, test, or demo databases no longer required
 - CKIMAGE
 - Print and review CKIMAGE0
 - Purge any databases on longer required

- Check utilities that are supported by North America Response Center (NARC).
 - Opportunity to remove older and duplicate utilities, before adding POSIX .2

Prepare Your MPE/iX System Before Updating to 5.0

utilities

- Review all older TOOLS that are now supported by RC
 - LISTFILE CONTIGXL.@.@,2
 - LISTFILE SYSINFO.@.@,2
 - LISTFILE DTSINFO2.@.@,2
 - LISTFILE DIOGENES.@.@,2
 - LISTFILE TSDECOMP.@.@,2
 - LISTFILE ERR.@.@,2
 - LISTFILE TREELIST.@.@,2
 - LISTFILE VALIDATE.@.@,2
 - LISTFILE DTCVALID.@.@,2
 - LISTFILE SHOWCLKS.@.@,2
 - LISTFILE UHAUL.@.@,2
 - LISTFILE EPTFIND.@.@,2
 - LISTFILE LIST.@.@,2
 - LISTFILE PSCREEN.@.@,2
 - LISTFILE TSIF.@.@,2
 - LISTFILE SIF.@.@,2
- Remove all older TOOLS that are now supported by RC
- Check for additional utilities that are now supported by MPE
 - Opportunity to review and cleanup older utilities
 - Remove all older TOOLS that are now supported MPE
 - LISTFILE BULDACCT.@.@,2
 - LISTFILE CLKPROG@.@.@,2
 - Purge older and duplicate utilities
- Check COMMAND.PUB.SYS for User Defined Commands conflicts
 - Each new release offers enhancements to CI commands
 - Opportunity to review and cleanup User Defined Command files
 - Review for new commands that may be overwritten by UDCs
 - MAKE, TOUCH, PASSWORD can be used by other applications or 3RD Party software products
 - Review commands
 - Resolve conflicts, if any

CKIMAGE command file

```

PARM entry=main
if '!entry' = 'main' then
  comment =====
  comment | CKIMAGE          15Jun94          Al Dulaney
  comment | |                                     Hewlett Packard
  comment | The purpose of this command scripts is to search
  comment | for all TurboIMAGE databases and report the date
  comment | it was created using FINFO, pass the database
  comment | to DBUTIL and display flag information, and
  comment | finally pass database file name to LISTFILE,2.
  comment | Then review the output CKIMAGE0 to identify any
  comment | databases that can be deleted from the system.
  comment | =====
  setvar HPAUTOCONT TRUE
  setvar _savemsg HPMSGFENCE
  comment suppress "END OF PROGRAM" message and hide errors
  setvar HPMSGFENCE 2
  file testit;rec=-80,,f,ascii;temp;nocctl;disc=10000
  echo [CKIMAGE] Searching&I ...
  listf @01.@,6;*testit
  if HPCIEER = 724
    echo !HPCIERMSG
  endif
  setvar db_flen finfo("testit",19)
  PURGE CKIMAGE0
  BUILD CKIMAGE0;REC=-80,,F,ASCII;DISC=25000;NOCCTL
  FILE 11=CKIMAGE0,OLD;ACC=APPEND
  run ci.pub.sys;info="CKIMAGE sub1"; &
    stdin=testit;parm=3
  purge testit,temp
  comment reinstate previous HPMSGFENCE value
  setvar HPMSGFENCE _savemsg
  deletevar _savemsg,db_@
  echo [CKIMAGE] Completed!
  return
elseif '!entry' = 'sub1' then
  setvar _savemsg HPMSGFENCE
  comment suppress "END OF PROGRAM" message and hide errors
  setvar HPMSGFENCE 2
  setvar lcounter 0
  echo [CKIMAGE] Formatting names found&I ...
  while (lcounter < !db_flen) do
    setvar lcounter lcounter+1
    input theline
    setvar db_fflen len('!theline')
    setvar db_val pos('.', '!theline')
    setvar db_prt1 str('!theline',1,!db_val-3)
    setvar db_fflen !db_fflen+1
    setvar db_prt2 str('!theline',!db_val,!db_fflen-!db_val)
    setvar dbf1 db_prt1+db_prt2
    setvar dbf1 rtrim('!dbf1')
    run ci.pub.sys;info="CKIMAGE sub2"; &
      stdin=testit;parm=3
  endwhile
  setvar HPMSGFENCE _savemsg

```

Prepare Your MPE/iX System Before Updating to 5.0

```

deletevar theline, lcounter
elseif 'entry' = 'sub2' then
setvar _savemsg HPMSGFENCE
comment suppress "END OF PROGRAM" message and hide errors
setvar HPMSGFENCE 2
if not finfo('!dbf1',0) then
COMMENT Dbf1 does not exist.
if lft('!dbf1',1) <> '*' and lft('!dbf1',1) <> '$' then
COMMENT Qualify dbf1 before reporting non-existence.
if pos('.', '!dbf1') > 0 then
if &
pos('.',rht('!dbf1',len('!dbf1')-pos('.', '!dbf1')))&
> 0 then
comment echo ![ups('!dbf1')] does not exist.
return
else
comment echo ![ups('!dbf1')]!hpaccount does not exist.
return
endif
else
echo ![ups('!dbf1')]!hpgroup!hpaccount does not exist.
return
endif
else
echo !dbf1 does not exist.
endif
else
if finfo('!dbf1',9) <> 'PRIV' then
return
endif
echo [CKIMAGE] Processing !dbf1
COMMENT ** formal file designator **
echo (FINFO): Full file description for &
![[finfo('!dbf1',1)]] follows: > *11
COMMENT ** creator and create/modify dates **
echo Created by ![finfo('!dbf1',4)] on &
![finfo('!dbf1',6)]. > *11
echo Modified on ![finfo('!dbf1',8)] at &
![finfo('!dbf1',24)]. > *11
COMMENT ** file code **
if finfo('!dbf1',9) = '' then
echo Fcode: ![finfo('!dbf1',-9)]. > *11
else
echo Fcode: ![finfo('!dbf1',9)]&
(![finfo('!dbf1',-9)]). > *11
endif
COMMENT ** rec size, eof, flimit **
echo Recsize: ![finfo('!dbf1',14)], Eof:&
![finfo('!dbf1',19)], Flimit:![finfo('!dbf1',12)]. > *11
COMMENT ** foptions **
setvar _fopt finfo('!dbf1',-13)
echo Foptions: ![finfo('!dbf1',13)] (!#_fopt,&
![octal(_fopt)], ![hex(_fopt)]). > *11
echo > *11
deletevar _fopt
endif
setvar db_dbcmd 'show '

```

Prepare Your MPE/iX System Before Updating to 5.0

```

setvar db_dbcmd ('!db_dbcmd'+!'dbf1')
setvar db_dbcmd rtrim('!db_dbcmd')
setvar db_dbcmd ('!db_dbcmd'+ ' all')
build testit2;rec=-80,,f,ascii;nocctl;disc=50
file xx=testit2,old;acc=append
echo !db_dbcmd > *xx
setvar db_dbcmd 'EXIT '
echo !db_dbcmd > *xx
rename testit2,testit3
run dbutil.pub.sys;stdin=testit3;stdlist=*ll
purge testit3
echo > *ll
setvar db_dbcmd1 ('!db_prt1'+'@')
setvar db_dbcmd1 ('!db_dbcmd1'+!'db_prt2')
setvar db_dbcmd1 rtrim('!db_dbcmd1')
setvar db_dbcmd1 ('!db_dbcmd1'+',2')
comment LISTFILE dd@.GRP.ACCT,2
listfile !db_dbcmd1 > *ll
setvar HPMSGFENCE _savemsg
comment End-Of-Command file
endif

```

Prepare Your MPE/iX System Before Updating to 5.0



Paper Number 8016
Open Systems Backup: How to Maximize Performance

Andrew Ibbotson & Uwe Hinrichs
HICOMP Storage Technologies GmbH
Kapstadtring 2
Hamburg, GERMANY 22297
49.40.638.0917

Handouts will be provided at time of presentation



Standalone Clusters: A Model of Server-Workstation Clustering Using NFS

*Michael B. Williams
Medical Products Group
Hewlett-Packard Company
3000 Minuteman Road
Andover, MA 01810
(508) 659-3734*

1. Introduction

Workstation clustering combines the advantages of desktop workstations and individual multi-user systems, providing benefits such as the selective sharing of file and peripheral resources; tight workgroup coupling; flexibility in configuring local file systems and attaching local devices; high performance and functionality by maintaining a one user-one CPU ratio; clock synchronization; and a single point of system administration.

1.1 Common models of client-server computing

Many workgroup organizations are based on a traditional client-server model. This model places the bulk of resources and computing power on a machine designated as the server, and distributes lesser computing power to a group of machines designated as the clients of that server. The server's resources are dedicated to serving the clients, and the clients' resources are dedicated to serving the users of the workgroup. For that reason, the clients are also referred to as workstations.

1.1.1 The Network File System

One way of exploiting this client-server model is to use the NFS (Network File System) networking protocol. This protocol allows the clients to use resources (such as disks) from other networked machines, including the designated server. The clients use some of the server's resources, but the clients are still standalone machines with their own disk resources. Clients can normally function on their own.

Because each client in an NFS cluster must be able to function independently, each client has separate resources, and these resources may be configured differently. For example, each client has its own boot disk with startup files unique to each machine. Each client also has its own printer configuration and a different opinion of the correct time.

1.1.2 HP Diskless

One example of workstation clustering is an HP-proprietary implementation of the client-server model called HP Diskless clusters. (It is "proprietary" in that it is supplied, supported, and marketed solely by HP.) This protocol more strongly supports

workgroup computing, because it provides a tighter coupling between the server and its clients. The use of the term "cluster" reflects this strong coupling. The clients in an HP Diskless cluster are dependent on the server for resources other than just disk resources. Unlike an NFS client, an HP Diskless cluster client cannot boot without communicating with its server, and it continuously relies on its server during its operation.

In contrast with an NFS server, an HP Diskless cluster server shares many of its resources with its clients. Resources such as startup disks, printers, and even the system clock are shared (and therefore consistent) throughout an HP Diskless cluster. This strong dependency between the clients and their server makes administering an HP Diskless cluster much simpler and easier than a comparable NFS cluster.

1.2 A new clustering model

This paper describes a model of server-client workstation clustering that provides the functionality of the HP Diskless cluster model using standard NFS networking protocols. The server and its clients form a cluster of standalones, known simply as a standalone cluster. This consists of a standalone cluster server and one or more standalone cluster clients. The Standalone Cluster model represents an enhanced alternative to the traditional NFS client-server model, in which a client workstation boots from a local disk and mounts resources from a network server. Through a combination of NFS access and selective file replication, the model addresses the synchronization and administration challenges presented by the NFS model.

In addition, this paper presents a case study of the application of the Standalone Cluster model to a software development environment involving five HP-UX 9000/750-class servers and fifty HP-UX 9000/700-class workstations running HP-UX 9.01. The paper also describes the effects of the implementation on system administration at a local and site level, engineer productivity and acceptance, and the installation and use of software tools within a standalone cluster environment.

1.3 Motivation for a new clustering model

As a proprietary protocol, HP continues to support HP Diskless with its own products and services, but as with many proprietary technologies, compatibility problems may arise during integration with third-party solutions.

Such an incompatibility exists with Atria Software's *ClearCase*, the software configuration management tool that has been selected for use in the local software development effort.

This incompatibility would be resolved if Atria were to develop a version of *ClearCase* that is compatible with HP Diskless. However, this requires that HP provide the necessary operating system services for third-party products and that Atria implements, tests, and markets this product.

The alternative is to abandon HP Diskless networking in favor of standard NFS networking. This is undesirable, because the HP Diskless cluster model is familiar to most local users, and it simplifies the administration of a large number of systems.

These two unlikely choices provided the motivation to develop a third alternative: a clustering method that is based on NFS but that provides many of the services and benefits of HP Diskless. Such an approach provides benefits in two key areas:

- *Consistent user model.* Users can continue to work with a workgroup model almost identical to the one that HP Diskless provides. Unlike migration to a wholly NFS-based configuration, activities (such as logging in) and resources (such as printer settings) work exactly as they did before.
- *Minimal additional system administration.* The work of system administrators is kept to a minimum, because much of the synchronization and consistency that HP Diskless provides is managed automatically by the implementation of the model.

2. The Standalone Cluster Model

The Standalone Cluster model works by intelligently sharing and replicating resources. Fundamentally, the model shares as much of the server's filesystem with the clients as is feasible. The resources that cannot be shared in this manner are replicated on client machines. Together, the share-and-replicate method provides high availability for all the significant dynamically changing data on the server.

2.1 Sharing Resources

Most of the significant static and dynamic filesystem resources from the server can safely be shared simultaneously by the clients, without concern for debilitating conflicts. This includes users' home directories as well as most HP and third-party applications.

2.1.1 Sharing resources using NFS

In general, sharing filesystem data and applications over NFS is straightforward. It is possible to use NFS to mount an application's directory hierarchy that exists on one system (referred to as the server) to a location on another system (the client). The most straightforward approach is to create a separate and distinct mount point for each application, as depicted in the diagram below.

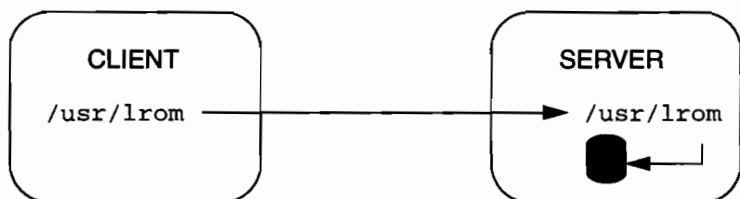


Figure 1. An NFS-mounted application directory references the same directory on the server.

Another approach is to mount the server's filesystem as a whole. If several such applications are to be shared, the server's filesystem can be mounted in a convenient place on the client and symbolic links can be created for each application.

For example, if the server `hpsrvr` is wholly mounted on the client `hpc1nt` under the directory `/nfs`, then an application whose directory listing looks like this on the server:

```
hpsrvr> ll -d /usr/lrom
dr-xr-xr-x 10 bin  bin 1024 Jan 11 09:41 /usr/lrom
```

and appears like this on the client:

```
hpc1nt> ll -d /usr/lrom
lrwxrwxrwx 1 root sys 21 Jun 8 1992 /usr/lrom ->
/nfs/hpsrvr/usr/lrom
```

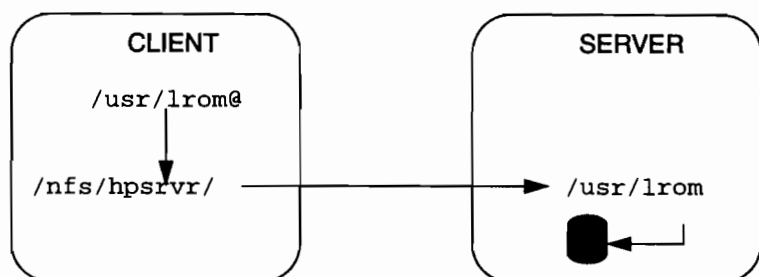


Figure 2. A symbolic link points to an application directory on an NFS-mounted disk.

2.1.2 Disadvantages of sharing information using symbolic links

Sharing applications by using symbolic link references has several disadvantages:

- *Routine and repetitive maintenance.* The process of creating a symbolic link on the client to the application directory on the server is not automatic. The system administrator must repeat this step on each client of the cluster, and it must be performed for each application that exists or is later installed on the server.
- *Lack of scalability.* This approach does not lend itself well to applications whose installation spans more than one or two directories. In such cases, additional symbolic links must be created. This multiplies the work for the system administrator and complicates the management of the associations.

As an example, consider a complete installation of HP C++ for HP-UX 9.01. The installation process places files or directories in the following locations:

/etc/newconfig/90RelNotes/	/usr/include/CC/
/system/HPCXX/	/usr/include/SC/
/usr/bin/	/usr/lib/nls/C/
/usr/contrib/CC/Examples/apollo-examples/	/usr/lib/
/usr/contrib/CC/Examples/att-examples/	/usr/lib/CC/eh/
/usr/contrib/CC/Examples/hp-examples/bank_ex/	/usr/lib/
/usr/contrib/CC/Examples/hp-examples/	/usr/man/man1.Z/
/usr/contrib/CC/Examples/hp-examples/xd_b_sessions/	/usr/man/man3.Z/
/usr/contrib/CC/Examples/hp-examples/library_ex/	/usr/CC/man/SC/man1/
/usr/contrib/CC/Examples/hp-examples/xd_b_commands/	/usr/CC/man/SC/
/usr/contrib/CC/Examples/hp-examples/calling_c/	/usr/CC/man/SC/man3/
/usr/contrib/CC/Examples/hp-examples/calling_CC/	/usr/CC/man/SC/man4/
/usr/contrib/CC/Examples/hp-examples/exception/	/system/HPCXX/
/usr/contrib/CC/Examples/hp-examples/template/	

At first glance, it seems ambitious to try to share this application using symbolic links to another NFS-mounted filesystem. It is possible (if tedious) to set up symbolic links from each client machine to each subdirectory on the server. However, this step alone would not solve the problem. The installation process places many files within *existing* subdirectories—most of which must exist in order to boot up the machine.

At a minimum, sharing HP C++ 9.01 requires the system administrator to create 47 symbolic links on each client machine, as shown in the table below.

Table 1: Distribution of Symbolic Links Required for HP C++ 9.01

Location	Number of Links Required
/etc/newconfig/90RelNotes/	0
/system/HPCXX/	0
/usr/bin/*	11
/usr/contrib/CC/	1
/usr/include/CC/	1
/usr/include/SC/	1
/usr/lib/nls/C/	2
/usr/lib/*	28
/usr/lib/CC/	1
/usr/man/	1
/usr/CC/	1
/system/HPCXX/	0
Total	47

2.1.3 The Standalone Cluster approach

The Standalone Cluster model takes a different approach to resolve the problems of the lack of scalability and the need for routine, repetitive maintenance. Instead of sharing application directories individually, the server's entire `/usr` hierarchy—where most application directories reside—is wholly shared by the clients.

One key to this approach is that on the client, `/usr` is not a symbolic link to the server's `/usr` hierarchy. Instead, the server's `/usr` hierarchy mounts over and obscures the client's `/usr` hierarchy. This approach also ensures that any files that are necessary for booting the client machine can remain anywhere within the client's filesystem, even under the `/usr` hierarchy.

However, even this approach is not as straightforward as it may first appear. Not all the directories within the `/usr` hierarchy may be safely shared by clients. Consider the numerous files and directories that contain logs and configuration files that are specific to a single machine.

For example, the SAM utility frequently generates scripts in the directory `/usr/sam/WORKSPACE` and logs in the directory `/usr/sam/log`. Many different system processes, from the `inetd` (the Internet services daemon) to `shutdown` (the HP-UX system shutdown utility), create logs in the directory `/usr/adm`.

Even seemingly unaffected directories such as `/usr/tmp` can create filename conflicts. Programs frequently consider it safe to use their process ID to create temporary files in `/usr/tmp`. It is possible that the same program running on two different machines will have the same PID and will use this number to generate the same temporary file name.

2.1.4 Completing the Standalone Cluster approach

The impact of multiple processes overwriting the same file ranges from innocuous to problematic. The Standalone Cluster model avoids sharing certain files and directories in the `/usr` hierarchy by redirecting requests for access to these locations back to the client machine's disk, as shown in the second diagram below.

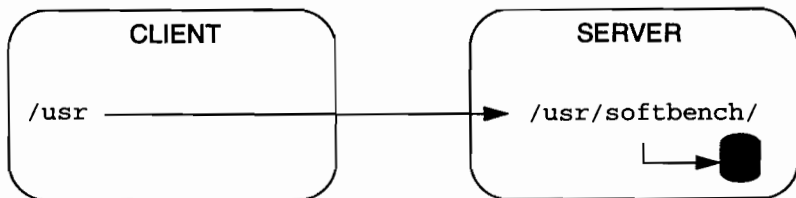


Figure 3. A direct request for the application directory `/usr/softbench`.

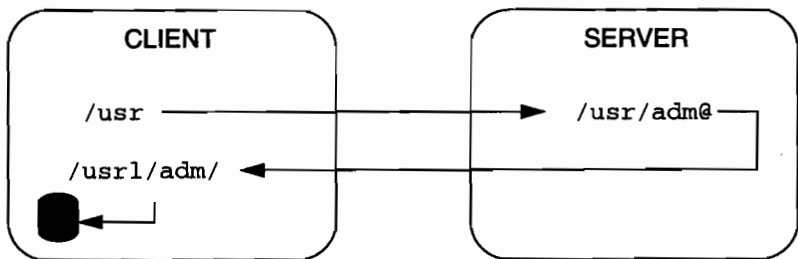


Figure 4. A redirected request for the system directory `/usr/adm`. `/usr/adm` is actually a symbolic link to a local directory on the client.

The diagram in the first figure above depicts a request to a directory that may be shared. The diagram in second figure depicts a request to a directory that cannot be safely shared. Instead, the request is diverted by a symbolic link to a location that exists on the client's disk.

2.2 Replicating Information

Resources that cannot be directly shared rely on replication. An example of a resource that is replicated is `/etc/password`, the password file. This file satisfies the two criteria for the selection of files that need to be replicated: consistency and location.

Consistency of the password file across a cluster of machines is important; a user that is added to one machine in the cluster should be able to log into any workstation within the cluster. This mimics the HP Diskless cluster model, and maintaining this paradigm in the Standalone Cluster model obscures the differences in the underlying implementations. The *location* of the file necessitates replication, because the file is located outside the `/usr` hierarchy, so it is not automatically shared.

A more complex resource that meets the criteria for replication is the printer spooler configuration. The selection of printer names, destination classes, and the organization of the print destination in classes should be consistent across a cluster. This makes it possible to predict the behavior and capabilities of one printer or print class. This will be difficult if a printer that is used on one machine is missing or configured with a different name on another machine.

Unfortunately, it is not possible to share the `/usr/spool` directory, where the printer configuration is stored. It does reside within the `/usr` hierarchy, but it is machine specific. The directory must be localized, and the printer configuration must be replicated.

2.2.1 Client Services

In the above cases, replication is an ongoing process. The Client Services distribution package was developed to manage this process. Client Services consists of an

`ninstall(1)` package whose purpose is to maintain substantial consistency throughout a standalone cluster in ways that cannot be achieved by the simple sharing of data.

Client Services is performed by the `cron` utility at predetermined intervals and is responsible for tasks such as these:

- Replicating system files such as `/etc/password`
- Synchronizing the clients' clock with the clock on the server
- Distributing `/etc/checklist` and mounting the volumes specified
- Updating the `cron` configuration file
- Replicating the printer configuration
- Executing any needed patches

In addition, Client Services can convert a standalone workstation into a client that can operate within a standalone cluster.

3. System Definition of the Standalone Cluster Model

When applied to a cluster of machines, the Standalone Cluster model, is composed of three fundamental parts:

1. *Localization*, the static structure of the filesystem
2. *Client Services*, the dynamic operations that support the model
3. *Support software*, the administrative operations for applying the model

3.1 Localization

The static filesystem structure is defined by the localization process that is performed on each node of a standalone cluster. This process occurs only once for any given system. A `localize` script performs the operation for the following files and directories:

Table 2: Localized Files and Directories

File or directory	Purpose
<code>/usr/adm</code>	Stores system logs created by the <code>syslog (1M)</code> facility, shutdown, super-user, and network file distribution logs; functions as a repository for system-specific data used by applications
<code>/usr/lib/cron</code>	Stores logs created by the cron utility
<code>/usr/local/etc</code>	Site-specific directory containing administration commands; updated under the control of the site administrators
<code>/usr/remwatch</code>	Contains data collected about the current system by the HP RemoteWatch utility
<code>/usr/sam</code>	Contains logs and scripts generated when the SAM utility is run
<code>/usr/spool</code>	Contains directories that relate to the <code>cron</code> utility, the printer configuration, and other machine-specific information
<code>/usr/tmp</code>	Contains temporary files with names often based on machine-generated process identification numbers
<code>/usr/lib/sendmail.fc</code>	Stores the "frozen" sendmail configuration; regenerated by site administrators each machine

3.2 Client Services

The Client Services `ninstall` package manages the dynamic operations of the standalone cluster. `ninstall` is a "pull" method of file distribution; a "push" method (such as `rdist`) could also be used as a means to perform dynamic operations. However, dynamic operations are only a subset of the entire standalone cluster system. The `ninstall` utility is flexible enough to provide operations that assist both in the areas of localization and administration.

The design of Client Services makes an important distinction is made between *replication* and *distribution*. In providing a common view of system services such as the password file, Client Services can simply replicate the server's resource onto the clients.

However, providing a common view of the currently mounted filesystems is not so simple. The server's `/etc/checklist` file typically contains references to local disks that do not exist on the clients. In these cases, the resource is defined specifically for, and distributed to, the client machines.

3.2.1 Replication

The following resources are replicated:

Table 3: Resources Replicated by Client Services

Resource	Purpose
/etc/scluster	Standalone Cluster utility directory
/etc/sclusterconf	Equivalent to HP Diskless /etc/sclusterconf
/.rhost	Remote host access permissions file
/etc/profile	System-wide Korn/Bourne/POSIX shell login file
/etc/csh.login	cshell login file
/.profile	Root's shell login file
/.kshrc	Root's shell rc file
/etc/rc	System initialization file
/etc/rc.local	Cluster-specific system initialization file
/etc/motd	Message of the day
/etc/ttytype	TTY information file
/etc/passwd	Password file
/etc/group	Group file
/etc/inetd.conf	Inetd daemon configuration file
/etc/netgroup	Network group file
/etc/X0.hosts	X Window System permission file
/etc/services	Services file
/etc/services.local	Cluster-specific service name data base

3.2.2 Distribution

Some distributed resources are automatically generated by Client Services. Others are manually defined by the system administrator. The following resources are distributed:

The checklist file. The file /etc/checklist contains the filesystems that are automatically mounted when the system is booted.

Server's Checklist File	Client's Checklist File
<pre># hpsrvr /etc/checklist file /dev/dsk/c201d6s0 / hfs defaults 0 1 # Root device entry /dev/dsk/c201d5s0 /users hfs defaults 0 2 # /users disk /dev/dsk/c201d5s0 /users swap end rw 0 2 # swap</pre>	<pre># standalone cluster client /etc/checklist file /dev/dsk/c201d6s0 / hfs defaults 0 1 hpsrvr:/usr /usr nfs hard,timeo=14,rw 0 0 hpsrvr:/users /users nfs soft,timeo=14,rw 0 0</pre>
<pre># FrameMaker mount /frame1:/nfs/frame1/root nfs soft,ro 0 0</pre>	<pre># FrameMaker mount /frame1:/nfs/frame1/root nfs soft,ro 0 0</pre>

Figure 5. Sample /etc/checklist files for a standalone cluster server and its client

The cron file. The cron file specifies a list of commands along with their scheduled execution time. A server's cron file typically contains many more commands than a client's cron file. A standalone client's cron file contains commands to invoke Client Services at predetermined intervals.

Server's Cron File	Client's Cron File
<pre># log kernel diagnostic messages every 10 minutes 05,15,25,35,45,55 *** */etc/dmesg - >>/usr/adm/messages # execute sync command every 15 minutes 5,20,35,50 *** */bin/sync # Clean up old core files 0 20 *** find / \(-path /view -prune \) -o \(-fsonly hfs -name core -exec rm {} \; \) # clean out /tmp and /usr/tmp->/usr/tmp directories 30 4 *** find /tmp /usr/tmp \(-path /tmp/.A -prune - o -fsonly hfs \) -atime +4 -exec rm {} \; # Check for files bigger than 1Mbyte 10 0 * * 6 find /users -fsonly hfs -size +2000 -print -exec ll {} \; mailx -s "Big Files" root # DAT backup 0 23 *** (/etc/backup -archive 2>&1) mailx -s "Backup Report" root</pre>	<pre># log kernel diagnostic messages every 10 minutes 05,15,25,35,45,55 *** */etc/dmesg - >>/usr/adm/messages # execute sync command every 15 minutes 5,20,35,50 *** */bin/sync # Clean up old core files 0 20 *** find / \(-path /view -prune \) -o \(-fsonly hfs -name core -exec rm {} \; \) # clean out /tmp and /usr/tmp directories 30 4 *** find /tmp /usr/tmp -fsonly hfs -atime +4 - exec rm {} \; # run Client Services 20 5 *** /usr/local/bin/ninstall -h' /usr/local/bin/scn- odes -r' client</pre>

Figure 6. Sample cron files for a standalone cluster server and client

The printer configuration. Unlike the checklist and cron files, the client machines' printer configuration is automatically generated by Client Services before being distributed to the clients. The configuration for client machines consists of an identical hierarchy of classes and printers. All printers are configured as remote printers hosted by the server.

3.2.3 Custom file distribution

The list of files replicated need not be limited to the predefined list. Client Services provides a method to distribute other files to the client machines. This is accomplished by replicating all files within a specific hierarchy marked for distribution.

Technically, this is a distribution, not a replication, because the files that are replicated are not (necessarily) shared by the server. The need for replication is addressed by treating symbolic links differently within the distribution hierarchy.

Instead of replicating the link itself, Client Services replicates the file that the link points to.

3.2.4 Patches

Client Services provides the ability to perform groups of operations on all workstations within a cluster. These operations are organized into shell scripts referred to as *patches*.

Patches have several characteristics that make these ideal for managing of a cluster of machines:

- Patches that complete successfully are guaranteed to be performed on a workstation only once.
- Patches that fail alert the administrator and are repeated until they are successful.
- Patch operations are always performed in a specific order. This insures the success of later patches that are dependent on earlier patches.
- If a machine is inoperable and cannot run Client Services, all previous patches are applied the next time that Client Services is run.

The above factors make patches a unique mechanism for easing the administration of a cluster of standalone machines.

Table 4: Sample Patches for Client Services

Patch	Purpose
0010FixUsrSpoolUucpPerm	Fixes the permission on the /usr/spool/uucp directory so that kermit can write a lock file for access to the device files
0020ClearCaseFix	Applies an HP-UX update patch for the ClearCase configuration management application
0030MakeEtcIssue	Customizes the /etc/issue file for each workstation, ensuring that the file specifies the proper system name and identifying characteristics
0040FixFloppyDevPerm	Fixes the permission on the floppy disk device files so that a user can read and write to the floppy using the doscp(1) family of commands
0050MakeFloppySoftLink	Creates symbolic links for easier access to the floppy disk device files

3.2.5 Clock synchronization

One important service that Client Services provides is the synchronization of the clocks on the client machines with the clock on the server. In many workgroup environments, a clock skew of a few minutes is not significant. In a networked software

development environment, it is crucial that all machines in the cluster have the same system time.

HP Diskless also maintains a consistent system clock throughout a diskless cluster. It is also important to provide similar functionality in order to maintain the end users' expectations of consistency.

3.3 Support software

In addition to replication, Client Services provides several administrative operations for maintenance of the standalone cluster configuration:

- *Localization.* The localization process which creates the symbolic links
- *Conversion.* Conversion of system files to match those on the standalone cluster server
- *Migration.* Detaching a standalone cluster client from a specific standalone cluster server. (The client's filesystem remains localized.)
- *Validation.* Verification that a system's configuration is valid for operation as a standalone cluster client. This is supported by the localization and conversion routines. These operations are idempotent, so they can be used as often as necessary whenever the client's configuration is in doubt.

and several utilities for convenient support operations:

- *scnodes.* The standalone equivalent of the HP-UX `cnodes(1)` utility
- *pull.* A utility for standalone cluster clients that manually invokes Client Services for the client
- *push.* A utility for standalone cluster servers that manually invokes Client Services on machines in the cluster.

4. Case Study: The Medical Products Group Software Development Effort

The development effort of the local division of the Hewlett-Packard Company Medical Products Group (MPG) consists of approximately 50 software development engineers developing software using HP 9000/700-series workstations running HP-UX 9.01.

4.1 Roles and responsibilities

The research, development, and support environment is composed of four overlapping classes of participants with varying roles and responsibilities:

4.1.1 Lab Users

Lab Users are the principal software developers. They do not generally have any responsibility for system administration. Lab Users are organized into project teams. The teams are supported directly by their Project Team Contact and indirectly by their standalone cluster's Cluster Coordinator.

This hierarchy is used whenever support concerns arise. Lab Users bring any questions or problems to their Project Team Contact. If the Project Team Contact needs

assistance in resolving the matter, the Project Team Contact escalates the issue to the Cluster Coordinator.

This escalation hierarchy serves two important purposes:

- It distributes knowledge about system administration processes and solutions throughout the organization.
- It ensures that, as the organization progresses, only significant problems are brought to the attention of the Cluster Coordinator, providing more time for system administration activities.

4.1.2 Project Team Contacts

Project Team Contacts, who are also software developers, provide support for Lab Users within their project teams. They are called upon for tasks such as the following:

- Answering questions from their team about the software development environment
- Finding solutions to problems that members of their teams encounter
- Managing equipment purchases by placing and tracking orders through the system

The Project Team Contact's most important responsibility is to ensure that their group's system administration needs are being met by the Cluster Coordinator.

Project Team Contacts do not bear responsibility for the day-to-day operation and maintenance of the standalone cluster. However, they may need to understand certain concepts of the administration of a standalone cluster in order to provide front-line support for Lab Users.

4.1.3 Cluster Coordinators

The roles and responsibilities of the designated Cluster Coordinator are similar to those for the administration of an HP Diskless cluster. Among other things, Cluster Coordinators are responsible for these tasks:

- Ensuring day-to-day maintenance of a standalone cluster
- Determining how to install software properly in a distributed standalone cluster environment
- Assisting Project Team Contacts in configuring new equipment properly

4.1.4 Site Administrators

Site Administrators are the support personnel for the site. They have responsibility for the system administration needs of many different groups at the site. They do not have any software development responsibility, except that which pertains to site-wide activities.

4.1.5 Organization of Roles

Cluster Coordinators were initially selected from the pool of Project Team Contacts, as shown in the first figure.

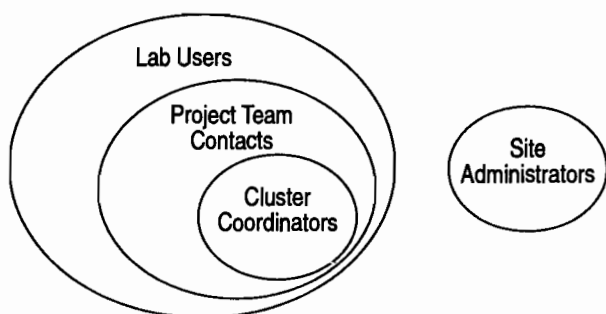


Figure 7. Roles in a Standalone Cluster (previous organization)

Later, a contract engineer was designated to serve as the Cluster Coordinator for all projects, as shown below.

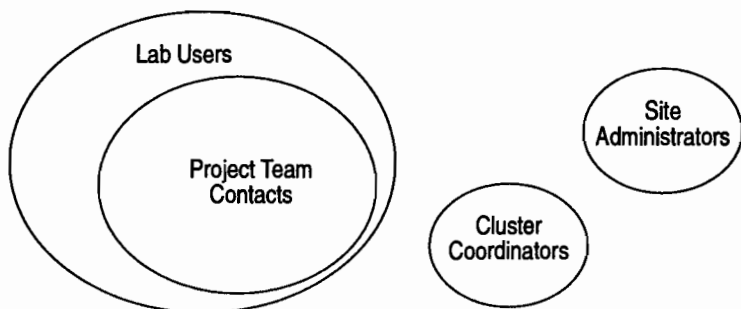


Figure 8. Roles in a Standalone Cluster (current organization)

4.2 Impact of a Standalone Cluster Environment

A standalone cluster environment affects the participants differently depending on their role.

4.2.1 Lab Users and Project Team Contacts

One of the goals of the Standalone Cluster model is to limit the visible impact of change on Lab Users, who are the end users of the environment. A successfully designed and implemented model will not significantly change the way that most lab users work. The Standalone Cluster model substantially satisfies this requirement.

The predominant workstation environment once consisted of HP Diskless clusters of eight to ten HP 9000 300- and 400-series machines. The majority of the Lab

Users are both familiar with and comfortable with this environment. For these users, the transition from an HP Diskless cluster to a standalone cluster is simple—only the name of their workstation changes.

There are two changes to entrenched policies that affect many Lab Users:

- *Proper shutdown techniques.* The most significant difference is that all systems are effectively standalone systems instead of HP Diskless cluster nodes. Standalone systems differ from diskless systems by having an internal disk from which to boot the operating system. It is relatively safe to shut down an HP Diskless cluster node by simply turning off the power to the workstation, although it is not recommended. This practice is not allowed on standalone systems, because the contents of the disk can become corrupted if users do not follow the proper shutdown procedure.
- *Limitation on super-user privileges.* The cluster coordinator has the ability to grant super-user privileges to selected users at his discretion. Most lab users do not need super-user privileges to perform their work. If super-user privileges are granted, then the user inherits the responsibility for the proper maintenance and operation of the standalone cluster.

Of these, only the limitation on super-user privileges is frequently an issue with Lab Users. The Cluster Coordinator works with the user to provide alternatives whenever possible.

Project Team Contacts do not require super-user privileges or system administration responsibilities to be effective in their role. They are only required to understand the basic concepts of the standalone cluster configuration and how this configuration affects the support that they provide to Lab Users.

4.2.2 Cluster Coordinators

A primary beneficiary of the standalone cluster environment is the Cluster Coordinator. Administering a standalone cluster is more complex in many areas than an equivalent HP Diskless cluster, but Cluster Coordinators benefit in reduced administration tasks compared with separate, standalone systems, as depicted in Figure 9.

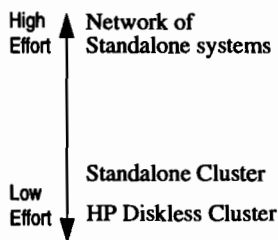


Figure 9. Relative system administration effort for three clustering models

There are a few aspects of system administration that require the Cluster Coordinator's special attention:

Installing systems. Normally, systems that are intended for use within the lab are configured by the Site Administrators. Because the procedure to install a standalone cluster client differs from the established site administration procedure, this task is instead performed by the Cluster Coordinator.

The additional effort expended in this regard is offset by a decreased workload for Site Administrators and a faster turn-around time for Lab Users. On the surface this may appear to be merely a shift of the workload from one person to another with no net benefit. However, consider the motives of the participants depicted in the roles and responsibilities diagram above: Cluster Coordinators have an obligation and an interest as both a Project Team Contact (who supports their particular project) and as a Lab User (who benefits from this support). There is no such motivation for Site Administrators to be as timely or responsive. The impact remains advantageous with the current organization, because the contract engineer has obligations to the development effort as a whole.

The installation process is simplified by the use of a Standalone Cluster Installation Disk as the model for the workstation's filesystem. This is a magneto-optical (MO) disk that contains the site-approved operating system with the standalone cluster organization. Several customized application installations (two of which are described below) are already applied to the disk. This filesystem is copied onto the workstation's disk, creating a working clone that is customized by the Client Services setup process.

Installing software. The design of the Standalone Cluster model accommodates the installation of most software without special treatment. Software that installs itself entirely in a single directory hierarchy, or completely within the `/usr` hierarchy, is automatically shared. It is not likely to require additional consideration.

Software that installs a critical portion of itself completely or partially outside the `/usr` hierarchy is an exception. In most cases, it is possible to complete the proper installation of such software by sharing files through data replication or NFS. The Standalone Cluster model provides facilities to manage both of these approaches.

The local MPG effort successfully uses a mix of HP and third-party software applications within the standalone cluster environment:

- *HP-UX C/Ansi C Developer's Bundle*, Hewlett-Packard Company
- *C++ SoftBench*, Hewlett-Packard Company
- *ClearCase*, Atria Software
- *Software through Pictures/Object Modeling Technique (StP/OMT)*, Interactive Development Environments (IDE)
- *Synchronize*, CrossWind Technologies
- *FrameMaker*, Frame Technology Corporation
- Various GNU utilities and applications such as *gcc*

Only two of these applications have required special treatment in order to function properly in a standalone cluster environment.

- *HP-UX C/Ansi C Developer's Bundle* replaces the standard distribution compiler with a full ANSI-compatible C compiler. Because it replaces executables and libraries on the root disk, it is necessary to install this package on each machine. No other precautions are required, because the

HP-UX `/etc/update` utility updates only the files on the local machine. Any files mounted from the server remain untouched.

- *ClearCase*, the motivation to move a way from HP Diskless clusters, is not a user-level application so much as a kernel-level application. It requires low-level modifications to the HP-UX operating system kernel. These modifications must be performed on the booted version of the operating system.

These special cases were identified early in the development of the Standalone Cluster model. All systems receive these customized packages from the Standalone Cluster Installation Disk.

Updating system files. System files that are machine independent are replicated automatically by the Client Services distribution package. These include files such as `/etc/password` and `/etc/group`.

Not all system files can be distributed in this manner. Files such as the checklist and cron files will differ based on whether the machine is a server or a client. In these cases, the server's file cannot be shared with the client. Instead, these files are stored in a separate location. The Cluster Coordinator prepares and updates these files specifically for the clients.

Managing the printer configuration. The Client Services distribution package automatically replicates the standalone cluster server's printer configuration on its client machines. As with HP Diskless clusters, printer configuration takes place on the cluster server.

4.2.3 Site Administrators

The local site administration department was consulted during the initial development of the Standalone Cluster model. However, it is not a joint development effort.

The standalone cluster environment is designed to coexist with established site administration policies and to minimize any impact on the methods and processes of these policies. This is an important goal for three reasons:

- These policies are generally out of our direct control. Site Administrators sometimes solicit feedback on procedures that affect the users they support, but Site Administrators may lack the resources to act on this feedback.
- Site administration policies can and do change. Lessening the effect that a standalone cluster environment has on site policies insulates the environment from changes in these policies. Changes that arise from outside the development group can have a negative impact on the group, as related below.
- Group policies can and do change. Lessening the effect that group policies have on site policies increases the freedom to change the group policies. This was especially important during the formative stages of the Standalone Cluster model.

Nevertheless, the support of the Site Administrators is crucial to the success of the standalone cluster environment. Problems and conflicts do arise. The willingness of the Site Administrators to accommodate customized environments is key to maintaining the stability of the environment, as shown in the following example.

Site Administrators had added a new section to the nightly netconfig distribution, which is similar to a site-wide Client Services package. This addition standardized the HP VUE configuration files. Suddenly, HP VUE refused to restart on any local workstations. The changes that the Site Administrators made overrode other changes that affected the operation of HP VUE on a standalone cluster. The Site Administrators reacted quickly, adding a feature that allowed systems to selectively deactivate this portion of the netconfig distribution.

The standalone cluster environment has benefitted the Site Administrators in many ways:

- *Recentralization.* There is a potential decentralization of support when HP Diskless clusters are separated into individual workstations. The migration to standalone clusters allows Site Administrators to focus on a few servers, instead of several dozen workstations.
- *Responsiveness.* Previously, new systems were sent directly to Site Administrators for installation and configuration. Because the site configuration differs from the standalone cluster configuration, this step is now performed by Cluster Coordinators. This eliminates a time-consuming step for the Site Administrators and reduces the turn-around time for the delivery of new machines.

5. Conclusions

Workstation clustering combines the advantages of desktop workstations and individual multi-user systems. The Standalone Cluster model grew from the need for tight workgroup coupling and easier system administration. It is an effective model of server-workstation clustering that uses NFS to provide much of the functionality of HP Diskless clusters. The Standalone Cluster model minimizes the impact on users of the underlying change in architecture and ensures consistency of operation with minimal effort by the system administrator.

The Standalone Cluster model retains these advantages of HP Diskless:

- Synchronization of system clocks within the cluster
- Transparent file sharing, so that most changes occur only in one place
- Flexibility, enabling workstations to have local filesystems and peripherals
- A common printer spooler view
- High performance and functionality by supporting a one-user, one-CPU ratio
- Tight workgroup coupling
- A single point of system administration for multiple systems

The Standalone Cluster model provides the following features while avoiding significant pitfalls of the HP Diskless cluster implementation:

- Reliance on an HP-proprietary protocol with limited third-party support
- The potential for obsolescence
- A restricted LAN domain, because HP Diskless cluster node operation is not robust across multiple networks

The administrative costs of the Standalone Cluster model are not significantly greater than those of an HP Diskless cluster, but they occur in different areas. The Standalone Cluster model shares some of the costs of HP Diskless clusters:

- The architectural complexity of the underlying implementation, which affects both ease of system administration and ease of use
- The concept of context (the current working system), and its relevance to activities such as spooler administration and peripheral configuration
- Variable application response time, because virtually all resources are shared over the network, which is susceptible to LAN traffic and interruptions

and incurs additional costs as well:

- Requirement of a local file system of sufficient size
- Lack of a remote swap server
- No provision for attaching spooled printers to the client nodes
- Lack of mixed clusters (clusters containing machines of different architectures)

We have successfully applied the Standalone Cluster model to our current development effort within the local division of HP's Medical Products Group. It has eased developers' transition from earlier HP Diskless systems to an HP 9000/700-based development environment. The Standalone Cluster model has enabled a single engineer to administer five HP-UX servers and fifty HP-UX workstations in an effective and consistent manner.

#8020: A Comparison of Hardware and Software Data Compression

Speaker: Husni Sayed
Written By: Deborah Littlefield
Researched By: Khalid Aziz
Ed Overacker

IEM, Inc.
1629 Blue Spruce Drive
Fort Collins, CO 80524
(303) 221-3005

Introduction

Data compression was originally conceived and designed to meet the needs of telecommunications companies that were transferring large amounts of analog-based data. By the 1970s, the need to compress digital data was becoming apparent. IBM mainframes, connected via telephone lines, were spending hours transferring data from one site to another.

Today, one of the most widespread applications for data compression is in the backup and archival of vital data. As this is a fairly typical application in today's computing environment, this paper will focus on data compression as it relates to the backup process.

Everyone implementing data compression is searching for the perfect balance of speed and size: to store information in as little space as possible, as fast as possible. The bottom-line considerations are how much storage space is used, and how long the data storage takes. However, there are some "hidden" costs to data compression that should be considered when deciding whether to compress in software or hardware (or whether to compress at all!).

After a brief survey of some different compression techniques and how they are implemented, we will investigate how compression affects system performance during the backup process, comparing hardware and software compression using a "real world" data set. The test results presented, gathered in both the HP 3000 and HP 9000 environments, will illustrate how the different methods affect CPU usage, data transfer rates, and compression ratios.

Why Compress?

As today's systems continue to expand, the need to compress the data that is being backed up grows. Data compression offers the benefits of:

1. **Reduced storage size.** Since compressed data can be stored in less space than uncompressed data, fewer tapes are required to hold the data.
2. **Increased effective transfer rate.** With a 2:1 compression ratio, 4 GBytes of uncompressed data can be represented in only 2 GBytes of compressed data. Therefore, the equivalent of 4 GBytes of information can be stored in the time it takes to write only 2 GBytes. While the actual physical transfer rates remain the same, the "effective" transfer rate is doubled.
3. **Reduced error rates.** Since fewer characters are actually stored, there are fewer chances for write or transmission errors to occur.

Compression Theory

Experts have analyzed billions of gigabytes of data, and found that most data sets consist largely of blanks, strings of spaces, and numeric/alphabetic repetitions. Even binary data consists of many ASCII strings, spaces, and "empty" areas. Data compression, as the name implies, compresses these repeated characters and strings into a more compact form, reducing the amount of space that a given amount of information occupies.

Compression Techniques

There are two major "types" of data compression techniques: character-based, and statistical.

1. *Character-based algorithms*, also known as substitutional algorithms, operate by substituting frequently occurring character patterns, or groups of repeated characters, with a single symbol.
2. *Statistical algorithms* make use of the fact that certain characters or groups of characters are more frequently used than others. With statistical compression, frequently used letters, such as "e" will occupy less space than less commonly used letters, such as "x."

Some common compression techniques are presented below. Although these techniques can be used by themselves, it is generally more effective to combine a number of these techniques into a single algorithm.

Null Suppression The Null Suppression technique scans a data stream for repeated blanks or null characters. When a number of sequential blanks are encountered, they are replaced by a special ordered pair of characters: the first is a special character (represented in this example below by the S_c character), and the second is the count, or number of blanks being replaced. For example, this string:

XYZ_{LLLLLLLLLLLL}abc

would be replaced with:

XYZ S_c 9abc

effectively reducing the string length from 15 characters to 8.

Note that if your character set has an undefined character that you can use as the S_c , the encoding will occupy two characters (the S_c and the count)—so you will benefit from substituting strings of 3 or more blanks. If you must use an extended character set to define your S_c character, the encoding will occupy four characters (shift in, S_c , shift out, and the count), so you would need a sequence of at least 5 blanks to see any space savings.

Diatomic Encoding This technique replaces commonly-occurring pairs of characters with a single special character. Theoretically, a 2:1 compression ratio can be achieved with this method, as you are substituting 2 characters with 1. However, the standard keyboard has over 100 characters—yielding over 10,000 two-character combinations. Even if it were possible to come up with 10,000 special characters to use as replacements, the overhead involved in assigning these substitutions and implementing the algorithm would be counter-productive.

With this technique, then, only the most frequently occurring two-character pairs are encoded. This is largely data-dependent: the most frequently occurring character pairs in an English language text are not the same as the most frequently occurring character pairs in a BASIC program. In fact, not one of the three most popular BASIC pairs—“_{LL}P,” “NT,” and “RI” (all of which appear in the BASIC PRINT statement)—appear in the “top 25” chart of an English language text.

The trick to this method, then, lies in knowing which data pairs to encode. Therefore, this technique has widely varying compression ratios on different types of data.

Pattern Substitution Pattern substitution, an offshoot of diatomic encoding, replaces longer pre-defined patterns with a special character. In a file containing English-language text, for example, the character strings “and,” “the,” “that,” and “this” would be excellent candidates for pattern substitution. In a BASIC language program, strings such as “PRINT” and “INPUT” would be good choices.

Bit Mapping Bit mapping can be employed when a data set contains a high proportion of one specific character (designated S_c in this example).

With this technique, a "bit map" character is appended to the beginning of every compressed string. This character consists of eight bits, each of which "maps" onto one of the next 8 characters: if the bit is cleared (0), the corresponding character is the specific character (S_c); if the bit is set (1), the corresponding character is something else ("not- S_c "). After the bit map character, only the "not- S_c " characters are stored.

This is illustrated below, using the blank character as the specific character (S_c):

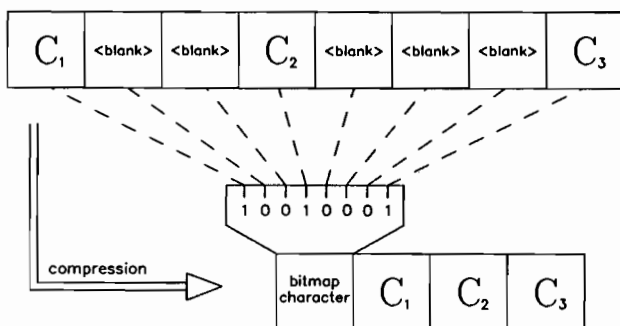


Figure 1: Bit Mapping Example

Run Length Run length encoding, a superset of null suppression, reduces any string of repeating characters by storing a special character (S_c) to indicate that the compression has taken place, followed by the character that is repeated, and then the repeat count. For example:

Original String	Compressed String
\$*****55.72	S_c *655.72
xxxxxxxxxx	S_c x9
This.....That	This S_c .8That

Figure 2: Run Length Encoding Samples

Huffman Coding Huffman coding makes use of the fact that certain characters are used more frequently than others. Typically, uncompressed data is stored so that each character occupies 8 bits. With Huffman coding, each character is mapped onto a code. The code for frequently used letters are shorter than those for less commonly used letters. This reduces the overall average code length.

The secret to Huffman Coding is that no "short" code can be identical to the beginning portion of a longer code. This ensures that the code is uniquely decipherable. Huffman coding uses a binary decision tree to make sure that no short code is identical to the start of a longer code. The final codes are formed by working backward through the decision tree. For example:

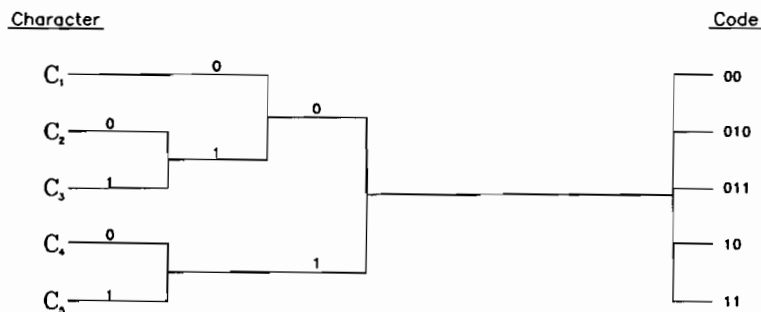


Figure 3: Assigning Codes in Huffman Coding

Compression Algorithms

Compression algorithms are generally comprised of one or more different techniques, to maximize the compression ratio achieved. Some common algorithms include:

- **Lempel-Ziv**

The Lempel-Ziv algorithm is implemented by a number of software packages, including *BackPack* (by Orbit), *Roadrunner* (by Tymlabs), and the *OmniBack* and *Compress* utilities (by HP).

- **DCLZ**

HP's 4mm DAT drives with compression use the DCLZ algorithm, a form of the Lempel-Ziv compression algorithm.

- **IDRC**

Exabyte's 8mm compression drives use the IDRC (Improved Data Recording Capability) algorithm, licensed from IBM.

Compression Ratios

Reviewing these compression techniques brings home an important point: the compression ratio that is achieved using a given algorithm is almost completely data-dependent—no single algorithm will be optimally effective on every data set. It is also important to keep in mind that all data compression algorithms have some degree of overhead. For instance, some sort of decoder or “dictionary” must be stored with the data so that it can be decompressed.

Therefore, there is no way to accurately predict the compression ratio that any given data set will achieve. The best that can be done is to offer a range of expected compression ratios, based upon the general “type” of data that is being stored. Binary data, for example, has an average compression ratio of about 1.8:1, while CAD files have an average compression ratio of more than 4:1. Some data, for example a table of random numbers, will not benefit from compression at all. In fact, with such data, the overhead involved in the compression process may be greater than the reduction achieved—resulting in the data actually expanding.

Table 1 below illustrates some typical compression ratios for different types of data.

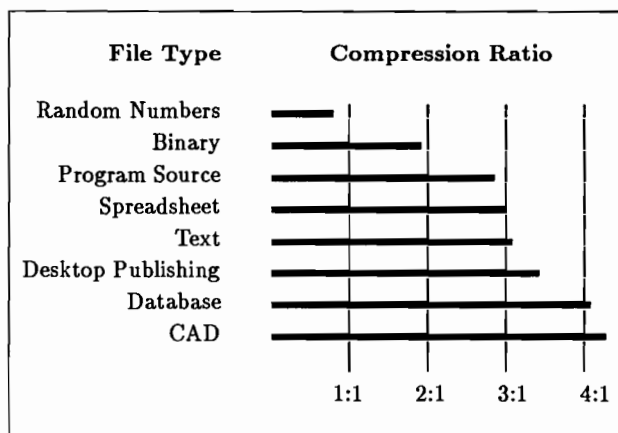


Table 1: Typical Data Compression Ratios

Hardware versus Software Compression

Until fairly recently, data compression was always implemented in software. Today, many storage products are available with compression implemented in the hardware. With this new choice comes the need to make an informed decision: does it make sense to compress in software or in hardware? Or, perhaps, in both? The answer is a resounding, "it depends." There are three major items to consider:

1. CPU time versus bus traffic

- Software data compression is highly computational, and therefore requires a great deal of CPU time. Hardware data compression requires no additional CPU time.
- With software data compression, data is compressed before it is sent out over the bus, reducing the amount of information that is being transferred. With hardware compression, the compression takes place after the data is transferred. Therefore, if the data set is highly compressible, software compression can result in significantly less bus traffic.

However, most buses have enough band width to accommodate a great deal of information: unless the bus is congested enough to become the bottleneck in your system, the reduced traffic does not buy you anything.

In most situations, the CPU drain of software compression negates any advantage of decreased bus traffic.

2. Compressed data size

If the size of the compressed data set is critical, you may need to choose the algorithm that provides the maximum compression. In any case, you are generally stuck with the algorithm that the manufacturer or software developer chose to implement. Experimenting with "typical" data is the easiest way to determine which method yields the highest compression ratio.

Compression in both software and hardware is generally not much more efficient than one or the other. In fact, re-compression data may actually expand the data—especially if the first algorithm was a good one.

Since helical scan technology offers such high single-tape capacities, most sites can easily store a full backup on a single tape. Therefore, maximizing the compression ratio is typically not the most critical consideration.

3. System Performance

While backups generally take place during off-peak hours, there is still a limited backup "window" during which the backup must complete so that the system is fully operational when normal business resumes. Therefore, for most sites, maximizing system performance is the critical factor.

Performance, and how it is affected by compression, is discussed in more detail in the following section.

Compression and System Performance

Virtually all computer-dependent organizations back up their data on a regular basis—and those that don't, should. The costs involved in recreating lost data far outweigh those involved in devising and implementing a sound backup strategy.

To see how compression impacts the performance of a system, it is important to have a basic understanding of what is happening. The following diagram illustrates a very simplified backup process. Data is read from one or more disk drives, transferred to the host computer where it is assembled and processed, and finally transferred to the tape drive or other storage device which formats the data and writes it to the tape.

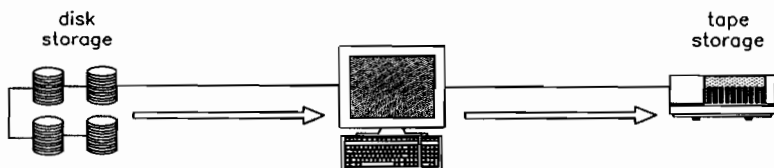


Figure 4: The Backup Process

The flow of information through the system cannot proceed any faster than the slowest step. Identifying the slowest step in the process identifies the "bottleneck." Potential bottlenecks include:

1. **Disk Drive** — how fast can the disk drive read data from the disk medium and send it to the host computer?
2. **Interface to the Host Computer** — how fast can information travel along the interface from the disk drives to the host computer?
3. **File System** — how fast can the operating system retrieve and assemble the records that make up each file?
4. **CPU Speed** — how fast can the CPU package and send data to the storage device? And, if software compression is being used, how quickly can the CPU perform this compression?
5. **Interface to the Tape Drive** — how fast can information travel along the interface from the host computer to the tape drive?

6. **Tape Controller** — how fast can the tape drive receive information? If hardware compression is enabled, how quickly can the tape drive perform the compression?
7. **Tape Media** — how fast can information be written onto the tape? This is affected by the speed at which a tape or a rotating head can be moved, and the amount of read-after-write verification that is done.

Generally speaking, disk drives and interfaces are faster than tape devices, and typically not the source of system bottlenecks. An exception to this occurs when an interface has multiple tape devices attached to it, with simultaneous bus activity taking place. For example, an interface with a throughput of 1 MB/second can easily handle a device that runs at 500 KB/sec. However, attach three such devices that are operating simultaneously, and the interface will need a throughput of 1.5 MB/second to keep up. For this reason, many installations find it is wise to have backup devices attached on a dedicated bus.

Surprisingly, the file system is often responsible for slowing down the backup process. The file system is responsible for placing the records of a file on disk, and retrieving and assembling those records when the file is read. If the file system scatters these records across the disk when storing the files, there will be a lot of overhead in seek time to retrieve every one of those records. A fast CPU combined with fast tape drives and disks can easily be bogged down by an inefficient file system.

When compression is implemented, the areas that need to be most closely scrutinized when searching for performance bottlenecks are the CPU, the tape controller, and the tape media.

Performance and the CPU

The CPU has a lot of work to do during the backup process. The CPU needs to manage the interface drivers that allow the information to be transferred. Once a file is retrieved, the CPU adds auxiliary information that helps it manage the backup process and media. Then the CPU must break each file into records of the appropriate length, and send them on their way.

Software compression is a highly computational task, and therefore demanding on CPU time. Since the CPU is already responsible for a large portion of what is going on, software compression may result in only minor improvements in performance. In fact, making the CPU responsible for compression could well cause a large decrease in performance.

Performance and the Tape Device

Once the CPU is finished with the information, it is sent along the interface to the tape drive. The tape drive controller then compresses the data (if compression is enabled), formats it, and writes it out to the tape medium.

If the data has been compressed in software, the amount of information received and processed by the tape controller is reduced. This would seem to indicate that compressing in software speeds up the work that is done by the tape device.

This is actually not the case, for two reasons. First, the delays that are incurred during the process of hardware compression are on the order of nanoseconds, and therefore negligible. Second, no matter how quickly the controller finishes its work, it is invariably the process of writing to the tape that slows everything down. The faster the controller finishes its work, the more time it spends "waiting" while the information is written to tape.

The rate at which information can be written to the tape, or the "laydown rate," is a function of the recording method used, the speed at which a tape or rotating head can be moved, and the number of media error retries and corrections that are made.

Device	Laydown Rate
8mm tape drive	500 KBytes/second
4mm DDS-1 tape drive	183 KBytes/second
4mm DDS-2 tape drive	512 KBytes/second

Table 2: Laydown rates for tape drives

With no compression performed in hardware, then, the laydown rate becomes the overall data transfer rate for the tape device. Because of this, it really doesn't cost any time to compress in hardware: the time the controller normally spends waiting can be spent compressing instead.

When hardware compression is turned on, the effective transfer rate increases linearly with the compression ratio, until the maximum transfer rate of the tape controller is reached:

Device	Maximum Transfer Rate
8mm tape drive	2.5 MBytes/second
4mm DDS-1 tape drive	732 KBytes/second
4mm DDS-2 tape drive	1.5 MBytes/second

Table 3: Maximum transfer rates for tape drives

Improving Overall Performance

The speed of all the steps of a backup process vary widely from system to system, and depend upon a number of factors: the speed of each component in the system, the utilities being used, and the other jobs running on the system at the time, to name a few.

The only way to improve performance of the overall system is to improve the performance of the slowest step. Adding a faster tape drive, or turning on compression to improve the effective transfer rate, will not help if the computer cannot send information along the interface fast enough. In fact, replacing a tape drive that is just slower than the computer with a tape drive the computer can't keep up with (which sometimes happens when you enable compression), could actually decrease system performance.

Tape drives have basically two modes: streaming mode, in which information is constantly being written to the tape, and stop/start mode, where the tape drive is waiting for information. On an 8mm tape drive, a single stop/start cycle takes about 3 seconds. As shown before, the effective transfer rate for an 8mm tape drive with no compression is 500 KBytes/second. With a 2:1 transfer rate, this becomes 1 MByte/second.

Suppose your drive is running, without compression, at 500 KBytes per second, even though it is receiving information at a slightly faster rate. In an effort to speed things up, you enable compression. Now your tape drive runs faster—so fast, in fact, that it runs for 1 second before it has to stop and wait for your data stream to catch up. It's now spending 3 of every 4 seconds in a stop/start cycle—changing the effective transfer rate from 1 MByte/second to 250 KBytes/second. Conclusion: if you can't keep the tape streaming with compression enabled, you're better off without it.

Test Results

The following test results illustrate how CPU usage, transfer rates, and compression ratios are affected by software and hardware compression on a "real world" data set.

Test 1: HP 3000

The first test was run with the following criteria:

- The test machine was an HP 3000/922 running MPE/iX 4.0.
- The system disk being backed up was attached via an HP-IB interface.
- The data set consisted of 81.72 MBytes of system files.
- The backup device was a 4mm DDS-1 tape drive attached via a single-ended SCSI interface.
- The backup was the only job running on the system.

Compression Method		CPU Usage	Transfer Rate (KB/sec)	Time (sec)	Tape Space (MBytes)	Compression Ratio		
SW	HW					SW	HW	Total
OFF	OFF	30%	179.2	467	81.72	—	—	1.00:1
OFF	ON	30%	344.4	243	38.91	—	2.0:1	2.00:1
ON(1)	ON	100%	334.7	250	39.29	1.3:1	1.6:1	2.08:1
ON(2)	OFF	100%	96.8	864	43.01	1.9:1	—	1.90:1
ON(2)	ON	100%	96.5	867	47.79	1.9:1	0.9:1	1.71:1
OFF(3)	OFF	85%	255.1	328	58.37	1.4:1	—	1.40:1
OFF(3)	ON	85%	355.6	235	26.53	1.4:1	2.2:1	3.08:1
ON(3)	OFF	90%	314.7	266	43.01	1.9:1	—	1.90:1
ON(3)	ON	90%	347.2	241	25.30	1.9:1	1.7:1	3.23:1

- (1) Backpack/XL by Tymlabs, compression level 3
- (2) Backpack/XL by Tymlabs, compression level 6
- (3) HiBack/XL by HiComp. Note that this software always compresses image datasets, whether software compression is ON or OFF.

Table 4: Test Results: 4mm DDS-1 Tape Drive on HP 3000

Test 2: HP 3000

The second test was run with the same parameters as the first, but using an 8mm tape drive as the backup device:

- The test machine was an HP 3000/922 running MPE/iX 4.0.
- The system disk being backed up was attached via an HP-IB interface.
- The data set consisted of 81.72 MBytes of system files.
- The backup device was an 8mm tape drive attached via a single-ended SCSI interface.
- The backup was the only job running on the system.

Compression Method		CPU Usage	Transfer Rate (KB/sec)	Time (sec)	Tape Space (MBytes)	Compression Ratio		
SW	HW					SW	HW	Total
OFF	OFF	30%	387.4	216	81.72	—	—	1.00:1
OFF	ON	30%	387.4	216	48.07	—	1.7:1	1.70:1
ON(1)	ON	100%	362.3	231	52.38	1.3:1	1.2:1	1.56:1
ON(2)	OFF	100%	96.4	868	43.01	1.9:1	—	1.90:1
ON(2)	ON	100%	96.4	868	43.01	1.9:1	1.0:1	1.90:1
OFF(3)	OFF	85%	382.0	219	58.37	1.4:1	—	1.40:1
OFF(3)	ON	85%	397.6	210	32.42	1.4:1	1.8:1	2.52:1
ON(3)	OFF	90%	337.8	248	43.01	1.9:1	—	1.90:1
ON(3)	ON	90%	343.1	244	33.08	1.9:1	1.3:1	2.47:1

- (1) Backpack/XL by Tymlabs, compression level 3
- (2) Backpack/XL by Tymlabs, compression level 6
- (3) HiBack/XL by HiComp. Note that this software always compresses image datasets, whether software compression is ON or OFF.

Table 5: Test Results: 8mm Tape Drive on HP 3000

Discussion: Tests 1 and 2

Tests 1 and 2 were identical, with the exception of the backup hardware that was used. Therefore, the results are roughly parallel, the main difference being that the overall transfer rates for 8mm are faster than for 4mm DDS-1. Some interesting notes about these test results:

1. Part of the test was conducted using *Backpack/XL* by Tymlabs, which offers various compression "levels". The results using the 8mm tape drive are shown below. The 4mm results are similar, with slightly different transfer rates.

Compression Method		CPU Usage	Transfer Rate (KB/sec)	Compression Ratio		
SW	HW			SW	HW	Total
OFF	OFF	30%	387.4	—	—	1.00:1
ON(1)	ON	100%	362.3	1.3:1	1.2:1	1.56:1
ON(2)	OFF	100%	96.4	1.9:1	—	1.90:1
ON(2)	ON	100%	96.4	1.9:1	1.0:1	1.90:1

(1) compression level 3

(2) compression level 6

Table 6: Comparing compression levels with Backpack/XL

Compression level 3 is less comprehensive, achieving a software compression ratio of 1.3:1. Even though it is using 100% of the CPU time, the transfer rate is affected minimally.

Compression level 6 is a more comprehensive algorithm, yielding a compression ratio of 1.9:1. The cost, however, is in the transfer rate—the added computations bog down the CPU, and the transfer rate plummets to roughly 25% of what it was without the software compression.

Generally speaking, the more thorough the software compression algorithm is, the more of a drain it will be on the CPU.

2. Another portion of the test showed the effects of compression in both software and hardware. The following table shows the results that were achieved using a 4mm DDS-1 drive as the backup device. Again, the results achieved with the 8mm tape drive were similar.

Compression Method		CPU Usage	Transfer Rate (KB/sec)	Compression Ratio		
SW	HW			SW	HW	Total
ON(1)	ON	100%	334.7	1.3:1	1.6:1	2.08:1
ON(2)	ON	100%	96.5	1.9:1	0.9:1	1.71:1
ON(3)	ON	90%	347.2	1.9:1	1.7:1	3.23:1

- (1) Backpack/XL by Tynlabs, compression level 3
- (2) Backpack/XL by Tynlabs, compression level 6
- (3) HiBack/XL by HiComp.

Table 7: Combining software and hardware compression

The compression ratio achieved by the hardware varies from 0.9:1 (in which case the data set is expanding slightly), to 1.7:1. Note that expansion of the data set occurred when the more extensive (level 6) software compression algorithm was implemented first. Before you make the decision to compress in both software and hardware, make sure you are actually getting something for your efforts. Trial and error is generally the only way to determine this.

3. Choice of a backup device also affects the system. This is a function both of the transfer rate of the device, and the compression algorithm implemented. Note, from these excerpts from the test results, that a single data set will achieve different results, based upon the backup device:

Backup Device	Compression Method		Transfer Rate (KB/sec)	Compression Ratio		
	SW	HW		SW	HW	Total
4mm DDS-1	OFF	ON	344.4	—	2.0:1	2.00:1
8mm	OFF	ON	387.4	—	1.7:1	1.70:1
4mm DDS-1	ON(1)	ON	334.7	1.3:1	1.6:1	2.08:1
8mm	ON(1)	ON	362.3	1.3:1	1.2:1	1.56:1
4mm DDS-1	OFF(3)	ON	355.6	1.4:1	2.2:1	3.08:1
8mm	OFF(3)	ON	397.6	1.4:1	1.8:1	2.52:1
4mm DDS-1	ON(3)	ON	347.2	1.9:1	1.7:1	3.23:1
8mm	ON(3)	ON	343.1	1.9:1	1.3:1	2.47:1

Table 8: Comparison of 4mm (DDS-1) and 8mm compression performance

4. Finally, in all cases, the transfer rates were the highest when hardware data compression was turned ON.

Test 3: HP 9000

The final test was run on an HP 9000 system. Because the HP 9000 utilities do not have the same capabilities as the HP 3000 utilities, information on the compression ratios and CPU usage were not obtained. This test merely compared the total system performance:

- The test machine was an HP 9000 712/60 machine running HP-UX 9.03.
- The system disk being backed up was attached via a single-ended SCSI interface, and the backup device was a single-ended SCSI 8mm tape drive. Both the system disk and the backup device were attached on the same SCSI bus. No other jobs were running while the backup was taking place.
- The data set being backed up consisted of 1612.53 MBytes of files containing ASCII data.

Compression Method		Transfer Rate (KB/sec)	Backup Time (sec)
SW	HW		
OFF	OFF	235.2	7020
ON	OFF	245.7	6720
OFF	ON	417.0	3960
ON	ON	218.4	7560

Table 9: Test Results: 8mm Tape Drive on HP 9000

Discussion: Test 3

Test 3, run on an HP 9000 machine, was primarily a comparison of transfer rates, since the ability to obtain information on CPU usage and compression ratios is not built into the utilities that were used.

In this test:

1. The transfer rate achieved with hardware compression ON and software compression OFF is significantly faster than any of the other test conditions.

2. Note that when software compression is ON, the use of hardware compression slows things down rather than speeds them up.

Theoretically, this doesn't seem to make sense—until you remember about tape stop/start times. In this example, the software compression is taking up CPU time, slowing the transfer of information to the tape controller. With hardware compression turned OFF, this is still fast enough to keep the tape streaming. When compression is turned ON, however, the tape drive operates faster than the computer can send it information. This incurs stop/start cycles, and slows the entire process down.

With a software algorithm that is less CPU-intensive, this might not be the case.

Conclusion

All sites have different needs, different data sets, and different equipment with which to implement their backup plan. As the pieces of the puzzle are so diverse, what works for one site will be much different from what works for another. There are, however, some general conclusions that can be drawn from the data gathered;

In general:

1. The use of hardware compression provides higher overall effective transfer rates.
2. With a fast interface bus, such as SCSI, between the host computer and the storage device, the interface will rarely (if ever) be slower than the maximum transfer rate of the storage device. Therefore, the decreased bus traffic offered with software compression will not help the flow of information through the backup system.
3. With some slower interfaces, such as HP-IB, the interface may in fact become a bottleneck. This is especially true if there are multiple devices attached to the interface, operating simultaneously. In this case, the decreased bus traffic offered by software compression may in fact help overall system performance.
4. Overall compression ratios vary widely, depending largely upon the data set to the compressed, and the algorithm (or combination of algorithms) used. Trial and error is the best way to determine which combination works best for you.
5. Last, but not least—a critical thing to remember is to keep your tape drive streaming. Tape stop/start cycles not only have a significant impact on system performance, they wear the mechanisms and reduce the MTBF of the drive as well.

Paper Number 8022
Operational Management and Control in Distributed Environments

Timothy Early
President
Open Systems Integration, Inc.
2100 Clearwater Drive, Suite 208
Oak Brook, IL 60521

Handouts will be provided at time of presentation

Security in a Distributed Environment

by M. E. Kabay, Ph.D.

Director of Education, National Computer Security Association (Carlisle, PA)
President, JINBU Corporation (Montreal, Canada)

Copyright © 1994 by Plesman Publications*

Just because you've moved from your mainframe to a client-server network architecture doesn't mean you can forget about security. Client-server systems have specific security requirements which must be met to protect corporate information.

The client-server model is evolving to ever more specialization. Servers for specific functions are increasingly used in leading-edge networks. Even security fits this pattern: there are networks where user identification is carried out by interaction between each workstation and a security server.

Information systems security for client-server architectures involves people, hardware, software, communications and operations.

People

A fundamental problem for client-server systems is that they often grow out of PCs and small LANs. Management perceptions, abetted by the prejudices of traditional information systems staff, have downplayed the importance of such "small" systems. Client-server networks are therefore too often perceived as the mainframe's little siblings and therefore not worth much attention or effort to manage and secure.

Lack of training is the biggest problem facing users of client-server systems. We are watching an entire generation of systems and network managers take control of billions of dollars of equipment--and repeating errors solved decades ago in traditional mainframe environments.

Organizations have taken bright, talented people who attained excellence as office managers and administrative assistants and put them in charge of the new PC, the LAN and the new network.

Information security in such environments is terrible. Administrators tell their users to pick easy-to-remember words as their passwords--including their own name, their spouse's name, job function, or department. Backup tapes and cartridges are stored in cabinets right next to the servers.

With the proliferation of decentralized systems, lack of coordination is a constant problem. Access controls differ among LANs; users become frustrated by complicated, varying logon procedures for multiple systems. Resistance to all aspects of security increases.

Hardware

Vulnerabilities of client-server systems include configuration, access points and the boxes themselves. Data communications (datacomm) between elements of a network requires minutely-detailed configurations on both sides. Datacomm parameter files must be protected against inadvertent modification by untrained users. In addition, client-server systems naturally increase the number of possible access points for sensitive and critical data. Where once there was a single point of control for access—the mainframe at the hub—we now have multiple workstations. Too many networks today protect servers but leave workstation security entirely to the user. Finally, client-server systems have valuable workstations, each of which can be stolen or cannibalized for expensive parts.

Software

Security concerns include compatibility, data integrity, viruses and theft. Where mainframe data usually require proprietary, expensive software to make sense of the files, client-server systems use off-the-shelf products with standard file formats. Much of the work gets done locally on workstations where spreadsheets, word processors and graphics packages can easily exchange data. In addition, some naive users think that placing files on a server automatically makes it reasonable to provide shared access, not knowing that locking is required to assure data integrity for concurrent modifications.

Although there are few network operating system viruses, there are lots of workstation viruses (for PCs and Macintosh computers). Infected programs stored on a server can infect and reinfect unprotected workstations on the network and wreak havoc. Finally, software theft is a rare event for mainframe programs but is all too common among uninformed users.

Communications

Most people are still using non-encrypting LANs and modems. Their vulnerability to eavesdropping is enormous. Anyone running an off-the-shelf 'sniffer' package on their workstation can trap packets circulating on Ethernet or Token Ring networks and decode the contents of dialogues between servers and any workstation on the net. Non-encrypting modems are an invitation to eavesdrop on phone lines, and too easy to use.

The likelihood of criminal hacker attack is about one or two percent per year per modem. With 30 modems, the chances of being attacked are 36% in a year.

Operations

Client-server systems can cause headaches to managers who are aware of their responsibilities. How do you handle version control on eight LANs with 23 servers and 400 workstations, only some of which run any of the 13 different software packages and proprietary in-house software? Version control includes the need to distribute new versions of your programs and utilities and to implement them without disruption of operations.

Too many organizations still have little control over third-party licensed software; users have a hodge-podge of different versions; they pass older versions illegally to new users when they install updates; and they copy restricted software as if they were copying memos.

Solutions

Answers to these problems do not come easily. Solutions require careful evaluation, analysis of alternatives, and planned implementation--just like in any other system architecture. They include

- hardware inventories, locks and network management;
- network antivirus software;
- software license and version management;
- encrypting data channels;
- workstation and audit trails;
- centralized backup tools;
- password tokens; and
- single logons.

Manual inventories can go far in tracking hardware. Everyone should have a simple database which can instantly answer questions like the following:

- What equipment has been assigned to Nancy Drew?
- Who has a Xylo-3456 board that can blow up under brownouts?
- How many 80 Mb disk drives do we have in all?
- Which pieces of equipment will be going off warranty by the end of this month?

However, such a database will be of limited value if people can "borrow" equipment from each other during midnight requisitions. To keep your inventory accurate, you should arrange to lock systems to prevent components from being removed without authorization.

Tools

A variety of tools is available to improve security. You can seriously inconvenience thieves by physically attaching all computers, including laptops and notebooks, to their work area. In several federal government departments in Ottawa, all employees are required to thread sturdy cables through attachment points on their workstations or portable computers when the units are at the office.

You should also use network management software to monitor what's on your network and who's been using it. Anti-virus tools allow you to check all executable files on your servers at modest cost.

Use encrypting data channels. New versions of LAN operating systems such as Novell NetWare 4.0 offer dynamic, user-transparent encryption of data flows.

Users of asynchronous dialup lines can turn to encrypting modems. With one on each end of the telephone line, it will be virtually impossible for anyone to eavesdrop successfully on your data transfers into your client-server system.

You should generate workstation audit trails. Look for configurability and reporting capabilities. Ideally, you should have encrypted audit trails to prevent tampering with the very tools that permit you to detect tampering.

Use hand-held access-control tokens. Instead of forcing users to memorize secret character strings, you can give them inexpensive (\$30-\$70 depending on style and volume) calculator-like password generators. These little devices contain cryptographically-sound algorithms for generating random-looking sequences of numbers or numbers and letters. Generated sequences are virtually unique, so access-control software running on your workstations or on the servers can safely calculate the serial number of the device that generated any given sequence. Users enter their normal logon ID, then copy the current sequence of characters from their access-control device into the password field. The workstation or server can then authenticate the user of the user ID.

Implement a single logon system. Avoid a perplexing mish-mash of different logons for users trying to access various servers and networks in your environment. Once users have successfully been identified and authenticated, they should be able to access anything they're entitled to see or work with without further hindrance.

Conclusions

Client-server networks are not inherently secure and they have peculiarities that deserve attention. You should actively protect your investment in the hardware, software and data that reside there.

-
- * This article originally appeared in *Computing Canada* as "Distributed security can be a paradox but tools and common sense will help" and was published in the *Report on Client-Server* in March 1994 (p. 9).

Reprinted with permission of the publisher.

