z/OS

zSeries Platform Test Report for z/OS and Linux Virtual Servers

Version 1 Release 7

SA22-7997-03

z/OS

zSeries Platform Test Report for z/OS and Linux Virtual Servers

Version 1 Release 7

SA22-7997-03

Note!
Before using this information and the products it supports, be sure to read the general information under “Notices” on page 349.

Fourth Edition, June 2006

This is a major revision of SA22-7997-02.

This edition applies to Parallel Sysplex environment function that includes data sharing and parallelism. Parallel Sysplex uses the OS/390 (5647-A01), z/OS (5694-A01), or z/OS.e (5655-G52) operating system.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

IBM welcomes your comments. A form for readers' comments may be provided at the back of this publication, or you may address your comments to the following address:

IBM Corporation
Department B6ZH, Mail Station P350
2455 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States & Canada): 1+845+432-9414
FAX (Other Countries): Your International Access Code +1+845+432-9414

IBMLink (United States customers only): IBMUSM(DILORENZ)
Internet e-mail: dilorenz@us.ibm.com
World Wide Web: www.ibm.com/servers/eserver/zseries/zos/integtst/

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Opening remarks

A message from our team

We changed our title from the *z/OS® Parallel Sysplex Test Report* but it's still us! Same team, same testing, but we've gradually expanded our focus from Parallel Sysplex to a platform wide view of z/OS's and Linux on zSeries' place in the enterprise. To reflect that focus, we changed our title to be the ***zSeries® Platform Test Report*** .

As you read this document, keep in mind that ***we need your feedback.*** We want to hear anything you want to tell us, whether it's positive or less than positive. ***We especially want to know what you'd like to see in future editions.*** That helps us prioritize what we do in our next test phase. We will also make additional information available upon request if you see something that sparks your interest. To find out how to communicate with us, please see "How to send your comments" on page xx.

We are a team whose combined computing experience is hundreds of years, but we have a great deal to learn from you, our customers. We will try to put your input to the best possible use. Thank you.

Al Alexsa
Maria Sueli Almeida
Loraine Arnold
Ozan Baran
Ryan Bartoe
Duane Beyer
Jeff Bixler
Muriel Bixler
Dave Buehl
Jon Burke
Alex Caraballo
Phil Chan
John Corry
Don Costello
Vince Crose
Luis Cruz
Tony DiLorenzo
Jeff Dutton
Michael Everett
Bob Fantom
Nancy Finn
Bobby Gardinor
Kieron Hinds
Gerry Hirons
Lisa Case-Hook
Joan Kelley
Fred Lates
Al Lease
Frank LeFevre
Scott Loveland
Douglas Macintosh
Sue Marcotte
Tammy McAllister
James Mitchell
Bob Muenkel
Elaine Murphy
Fred Orosco
Gary Picher
Jim Rossi
Tom Sirc
Karen Smolar
Jeff Stokes
Jim Stutzman
Lissette Toledo
Ashwin Venkatraman
Jin Xiong
Jessie Yu

Important—Currency of the softcopy edition

Each release of the *z/OS Collection* (SK3T-4269 or SK3T-4270) and *z/OS DVD Collection* (SK3T-4271) contains a back-level edition of this test report.

Because we produce our test reports twice a year, June and December, we cannot meet the production deadline for the softcopy collections that coincide with the product's GA release and the softcopy collection refresh date six months later. Therefore, there is normally a one-edition lag between the release of our latest test report edition and the softcopy collection in which it is included. That is, the test report that appears in any given softcopy collection is normally one edition behind the most current edition available on the Web.

If you obtained this document from a softcopy collection on CD-ROM or DVD, you can get the most current edition from the zSeries Platform Test Report Web site at:

www.ibm.com/servers/eserver/zseries/zos/integtst/

Contents

Figures

Tables

About this document

This document is a test report written from the perspective of a system programmer. The IBM zSeries Integration Test team—a team of IBM testers and system programmers simulating a customer production Parallel Sysplex environment—wants to continuously communicate directly with you, the zSeries customer system programmer. We provide this test report to keep you abreast of our efforts and experiences in performing the final verification of each system release before it becomes generally available to customers.

An overview of Integration Test

We have been producing this test report since March, 1995. At that time, our sole focus of our testing was the S/390® MVS™ Parallel Sysplex. With the introduction of OS/390® in 1996, we expanded our scope to encompass various other elements and features, many of which are not necessarily sysplex-oriented. In 2001, OS/390 evolved into z/OS, yet our mission remains the same to this day. In 2005, we expanded to add a Linux Virtual Server arm to our overall environment, which will be used to emulate leading-edge customer environments, workloads, and activities.

Our mission and objectives

IBM's testing of its products is and always has been extensive. ***The test process described in this document is not a replacement for other test efforts.*** Rather, it is an additional test effort with a shift in emphasis, focusing more on the customer experience, cross-product dependencies, and high availability. We simulate the workload volume and variety, transaction rates, and lock contention rates that exist in a typical customer shop, stressing many of the same areas of the system that customers stress. When we encounter a problem, our goal is to keep systems up and running so that end users can still process work.

Even though our focus has expanded over the years, our objectives in writing this test report remain as they were:

- Run a Parallel Sysplex in a production shop in the same manner that customers do. We believe that only by being customers ourselves can we understand what our own customers actually experience when they use our products.
- Describe the cross-product and integrated testing that we do to verify that certain functions in specific releases of IBM mainframe server products work together.
- Share our experiences. In short, if any of our experiences turn out to be painful, we tell you how to avoid that pain.
- Provide you with specific recommendations that are tested and verified.

We continue to acknowledge the challenges that information technology professionals face in running multiple hardware and software products and making them work together. We're taking more of that challenge upon ourselves, ultimately to attempt to shield you from as much complexity as possible. The results of our testing should ultimately provide the following benefits:

- A more stable system for you at known, tested, and recreatable service levels
- A reduction in the time and cost of your migration to new product releases and functions.

Our test environment

The Parallel Sysplex that forms the core of our test environment has grown and changed over the years. Today, our test environment has evolved to a highly interconnected, multi-platform on demand enterprise—just like yours.

To see what our environment looks like, see the following:

- "Our Parallel Sysplex hardware configuration" on page 5
- "Our Parallel Sysplex software configuration" on page 15
- "Our networking and application enablement configuration" on page 125
- "Our workloads" on page 18

Who should read this document

System programmers should use this book to learn more about the integration testing that IBM performs on z/OS and certain related products, including selected test scenarios and their results. We assume that the reader has knowledge of MVS and Parallel Sysplex concepts and terminology and at least a basic level of experience with installing and managing the z/OS operating system, subsystems, network products, and other related software. See "Where to find more information" on page xix.

How to use this document

Use this document as a companion to—***never*** *a replacement for*—your reading of other z/OS element-, feature-, or product-specific documentation. Our configuration information and test scenarios should provide you with concrete, real-life examples that help you understand the "big picture" of the Parallel Sysplex environment. You might also find helpful tips or recommendations that you can apply or adapt to your own situation. Reading about our test experiences should help you to confidently move forward and exploit the key functions you need to get the most from your technology investment.

However, you also need to understand that, while the procedures we describe for testing various tasks (such as installation, configuration, operation, and so on) are based on the procedures that are published in the official IBM product documentation, they also reflect our own specific operational and environmental factors and are intended for illustrative purposes only. Therefore, ***do not*** use this document as your sole guide to performing any task on your system. Instead, follow the appropriate IBM product documentation that applies to your particular task.

How to find the zSeries Platform Test Report for z/OS and Linux Virtual Servers

We make all editions of our test reports available on our z/OS Integration Test Web site at:

www.ibm.com/servers/eserver/zseries/zos/integtst/

If you cannot get to our Web site for some reason, see Appendix D, "Availability of our test reports," on page 343 for other ways to access our test reports.

We have traditionally published our test report on a quarterly basis where each quarterly edition was cumulative for the current year. At the end of each year, we freeze the content in our last edition; we then begin with a new test report the

following year. The most recent quarterly edition as well as all of the previous year-end editions are available on our Web site.

In 2003, our publication schedule changed from our traditional quarterly cycle as a result of the change in the development cycle for annual z/OS releases. We now publish our report twice a year, every June and December. In any event, the contents of our test reports remain cumulative for any given year.

We also have a companion publication, *OS/390 Parallel Sysplex Recovery*, GA22-7286-00, which documents the Parallel Sysplex recovery scenarios we've executed in our test environment, including operating system, subsystem, and coupling facility recovery. We describe how to be prepared for potential problems in a Parallel Sysplex, what the indicators are to let you know that a problem exists, and what actions to take to recover.

Note: The recovery book was written in the OS/390 V2R4 time frame; however, many of the recovery concepts that we discuss still apply to later releases of OS/390 and z/OS.

Where to find more information

If you are unfamiliar with Parallel Sysplex terminology and concepts, you should start by reviewing the following publications:

Table 1. Parallel Sysplex planning library publications

Publication title	Order number
z/OS Parallel Sysplex Overview	SA22-7661
z/OS MVS Setting Up a Sysplex	SA22-7625
z/OS Parallel Sysplex Application Migration	SA22-7662
z/OS and z/OS.e Planning for Installation	GA22-7504

In addition, you can find lots of valuable information on the World Wide Web.

- See the Parallel Sysplex for OS/390 and z/OS Web site at: www.ibm.com/servers/eserver/zseries/pso/
- See the Parallel Sysplex Customization Wizard at: www.ibm.com/servers/eserver/zseries/pso/tools.html
- See the z/OS Managed System Infrastructure (msys) for Operations Web site at: www.ibm.com/servers/eserver/zseries/msys/msysops/
- See the IBM Education Assistant which integrates narrated presentations, Show Me Demonstrations, tutorials, and resource links to help you successfully use the IBM software products at:publib.boulder.ibm.com/infocenter/ieduasst/stgv1r0/index.jsp

Using LookAt to look up message explanations

LookAt is an online facility that lets you look up explanations for most of the IBM® messages you encounter, as well as for some system abends and codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can use LookAt from these locations to find IBM message explanations for z/OS elements and features, z/VM®, VSE/ESA™, and Clusters for AIX® and Linux™:

- The Internet. You can access IBM message explanations directly from the LookAt Web site at http://www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/.
- Your z/OS TSO/E host system. You can install code on your z/OS or z/OS.e systems to access IBM message explanations using LookAt from a TSO/E command line (for example: TSO/E prompt, ISPF, or z/OS UNIX® System Services).
- Your Microsoft® Windows® workstation. You can install LookAt directly from the *z/OS Collection* (SK3T-4269) or the *z/OS and Software Products DVD Collection* (SK3T-4271) and use it from the resulting Windows graphical user interface (GUI). The command prompt (also known as the DOS > command line) version can still be used from the directory in which you install the Windows version of LookAt.
- Your wireless handheld device. You can use the LookAt Mobile Edition from http://www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/lookatm.html with a handheld device that has wireless access and an Internet browser (for example: Internet Explorer for Pocket PCs, Blazer or Eudora for Palm OS, or Opera for Linux handheld devices).

You can obtain code to install LookAt on your host system or Microsoft Windows workstation from:

- A CD-ROM in the *z/OS Collection* (SK3T-4269).
- The *z/OS and Software Products DVD Collection* (SK3T-4271).
- The LookAt Web site (click **Download** and then select the platform, release, collection, and location that suit your needs). More information is available in the LOOKAT.ME files available during the download process.

Using IBM Health Checker for z/OS

IBM Health Checker for z/OS is a z/OS component that installations can use to gather information about their system environment and system parameters to help identify potential configuration problems before they impact availability or cause outages. Individual products, z/OS components, or ISV software can provide checks that take advantage of the IBM Health Checker for z/OS framework. This book refers to checks or messages associated with this component.

For additional information about checks and about IBM Health Checker for z/OS, see *IBM Health Checker for z/OS: User's Guide*. z/OS V1R4, V1R5, and V1R6 users can obtain the IBM Health Checker for z/OS from the z/OS Downloads page at http://www.ibm.com/servers/eserver/zseries/zos/downloads/.

SDSF also provides functions to simplify the management of checks. See *z/OS SDSF Operation and Customization* for additional information.

How to send your comments

Your feedback is important to us. If you have any comments about this document or any other aspect of Integration Test, you can send your comments by e-mail to:

- dilorenz@us.ibm.com for z/OS questions
- meveret@us.ibm.com for WebSphere Integration Test (WIT) questions
- jinxiong@us.ibm.com for Linux on zSeries questions

or use the contact form on our Web site at:

www.ibm.com/servers/eserver/zseries/zos/integtst/

You can also submit the Readers' Comments form located at the end of this document.

Be sure to include the document number and, if applicable, the specific location of the information you are commenting on (for example, a specific heading or page number).

Summary of changes

We periodically update our test report with new information and experiences. If the edition you are currently reading is more than a few months old, you may want to check whether a newer edition is available (see “How to find the zSeries Platform Test Report for z/OS and Linux Virtual Servers” on page xviii).

This information below summarizes the changes that we have made to this document.

Summary of changes
for SA22-7997-03
June 2006

This document contains information previously presented in SA22-7997-02.

New information

- “Creating a split plex for production and test” on page 26
- “Testing the IBM Encryption Facility for z/OS” on page 32
- “Setting up CICS Java in our CICS TS 3.1 environment” on page 60
- “A walkthrough sample on how to use the BPXWH2Z migration tool” on page 167
- “Using the zSeries File System (zFS) version root file system” on page 173
- “Displaying z/OS UNIX and zFS diagnostic information through message automation” on page 175
- “Managing your z/OS queue managers using WebSphere MQ V6 Explorer” on page 182
- “Migrating from WebSphere MQ V5.3.1 to V6” on page 186
- “Setting up WebSphere for eWLM monitoring of DB2 applications” on page 201
- “Defining JMS and JDBC Resources for Trade6” on page 204
- “Providing authentication, course-grained security, and single sign-on for Web/EJB based applications running on WebSphere Application Server for z/OS” on page 207
- Part 3, “WebSphere Integration Test (WIT),” on page 219 including:
  - Chapter 16, “Introducing our WebSphere Integration Test (WIT) environment,” on page 221
  - Chapter 17, “About our z/OS WebSphere Integration Test (WIT) environment,” on page 225
- Part 4, “Linux virtual servers,” on page 233 including:
  - Chapter 18, “About our environment,” on page 235
  - Chapter 19, “Implementing High Availability Architectures,” on page 237
  - Chapter 20, “Migrating middleware,” on page 311
  - Chapter 21, “Installing and configuring WebSphere Portal Server Cluster,” on page 323
  - Chapter 22, “Linux and z/VM system programmer tips,” on page 325

Changed information

- Our sysplex hardware configuration

Removed information

- Defining greater than 16 CPs per z/OS image
- CFCC Dispatcher Rewrite testing
- z/OS UNIX enhancements in z/OS V1R5
- z/OS UNIX enhancements in z/OS V1R6
- Issuing the su command and changing TSO identity
- Parallel Sysplex Automation
- Improving availability with our MQCICS workload
  - One WebSphere MQ-CICS bridge monitor running on one system handling the requests
  - Three systems with WebSphere MQ-CICS bridge monitor task handling the requests
- Testing WMQI V2.1 on DB2 V8
- Setting the _BPXK_MDUMP environment variable to write broker core dumps to MVS data sets
- Resolving a EC6-FF01 abend in the broker
- Using the IBM HTTP Server
- Setting up the LDAP server for RACF change logging
- Using the z/OS LDAP client with the Windows 2000 Active Directory service
- Using LDAP with Kerberos authentication
- Setting up SSL client and server authentication between z/OS LDAP V1R6 server/client and Sun ONE Directory Server 5.2 server/client
- Setting up SSL client and server authentication between z/OS LDAP V1R6 server/client and IBM Tivoli Directory Server 5.2 server/client
- LDAP Server enhancements in z/OS V1R6
- Using Kerberos (Network Authentication Service)
- Migrating WebSphere MQ Integrator V2.1 to WebSphere Business Integration Message Broker V5.0
- Applying WBIMB V5.0 Fix Pack 02 and Fix Pack 03
- Updating the Retail_IMS workload for workload sharing and high availability
- Testing shared channel recovery
- Migrating WebSphere Application Server for z/OS Version 5.1 to Version 6
- Migrating WebSphere Application Server for z/OS JDBC from DB2 V7 to DB2 V8
- Using DB2 UDB JCC Connectors
- Failover Testing for JDBC using the Sysplex Distributor
- Utilizing memory-to-memory replication
- Migrating to CICS Transaction Gateway Connector V6.0
- Migrating to IMS Connector for Java V9.1.0.1
- Using the LDAP User Registry for WebSphere Application Server for z/OS administration console authentication
- Enabling Global Security and SSL on WebSphere Application Server for z/OS
- Using the WebSphere Application Server for z/OS 5.x plug-in for HTTP Server and Sysplex Distributor with our WebSphere Application Server for z/OS J2EE Servers
- Using EIM authentication

The above removed information can be found in our December 2005 edition which is available on our z/OS Integration Test Web site at:

www.ibm.com/servers/eserver/zseries/zos/integtst/

This document contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

References to OpenEdition have been replaced with z/OS UNIX System Service or z/OS UNIX.

Summary of changes
for SA22-7997-02
December 2005

This document contains information previously presented in SA22-7997-01.

New information

- "RACF Security Server mixed case password support" on page 31
- "Tivoli Workload Scheduler (TWS) EXIT 51 tip:" on page 20
- "Migrating to z/OS V1R7" on page 35
- "Migrating JES2 large spool datasets" on page 37
- "Migrating to z/OS.e V1R7" on page 38
- "Migrating z/OS Images and a Coupling Facility to the z9" on page 47
- Chapter 5, "Migrating to CICS TS Version 3 Release 1," on page 55
- Chapter 10, "Using IBM Health Checker for z/OS," on page 115
- "z/OS UNIX System Services: 64 MB Maximum for OMVS ctrace Buffer" on page 139
- "z/OS UNIX System Services: Dynamic Service Activation" on page 140
- "z/OS UNIX System Services: Display Local AF_UNIX Sockets" on page 144
- "z/OS UNIX System Services: /dev/zero, /dev/random, dev/urandom" on page 146
- "z/OS UNIX System Services: Display Information About Move or Mount Failures" on page 147
- "z/OS UNIX System Services: SETOMVS Enhancements" on page 149
- "z/OS UNIX System Services: Display Mount Latch Contention Information" on page 150
- "z/OS UNIX System Services: Enhancements to display file systems" on page 153
- "z/OS UNIX System Services: ISHELL Enhancements" on page 153
- "zFS: Migrating the Sysplex Root File System from HFS to zFS" on page 164
- "zFS: Improved Mount Performance (Fast-Mount)" on page 166
- "zFS: Migrating from HFS to zFS in z/OS V1R7" on page 167
- "zFS: Unquiesce Console Modify Command" on page 174
- "Updating the Retail_IMS workload for workload sharing and high availability" on page 193
- Appendix B, "Some of our RMF reports," on page 333

Changed information

- Our sysplex hardware configuration

This document contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

References to OpenEdition have been replaced with z/OS UNIX System Service or z/OS UNIX.

Summary of changes
SA22-7997-01
June 2005

This document contains information previously presented in SA22-7997-00.

New information

- "Exploiting 64k cylinder logical volumes" on page 12
- "Our Integrated Cryptographic Service Facility (ICSF) configuration" on page 31
- "On/Off Capacity On Demand Testing" on page 14
- "z800 Concurrent upgrade testing" on page 14
- Chapter 6, "Migrating to DB2 Version 8," on page 69
- Chapter 7, "Migrating to IMS Version 9," on page 97
- Chapter 9, "Testing SPE Console Restructure (APAR OA09229)," on page 113
- "Removing additional diagnostic data collection from OMVS CTRACE LOCK processing" on page 177
- "About our z/OS V1R7 test environment running WebSphere Application Server" on page 197
- Chapter 23, "Future Linux on zSeries projects," on page 329
- Appendix B, "Some of our RMF reports," on page 333

Changed information

- Our sysplex hardware configuration
- "WebSphere MQ for z/OS workloads" on page 22
- "WebSphere Message Broker" on page 23

This document contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Summary of changes
for SA22-7997-00
September 2004

This document contains information previously presented in SA22-7663-11.

New information

- Enabling NFS recovery for system outages
- Automount enhancement for HFS to zSeries file system (zFS) migration
- Using multipliers with BPXPRMxx parameters
- Using the superkill command
- Added an ipV6 environment equivalent to our ipV4 environment. V1R6 now supports OSPF V3 for ipV6 and ipV6 support for DVIPA and Sysplex Distributor.
- Using wildcard characters in the automove system list (SYSLIST)

- Using the clear and uptime shell commands
- Enhanced latch contention detection
- Using distributed BRLM
- Using ISHELL enhancements
- zFS modify console command
- Using gskkyman support for storing a PKCS #7 file with a chain of certificates
- LDAP migration to z/OS V1R6
- Setting up a peer-to-peer replication network between an IBM Tivoli® Directory Server 5.2 and a z/OS LDAP Server
- Using LDAP DB2® restart/recovery function
- Using LDAP alias support
- Using the enhanced LDAP configuration utility (LDAPCNF)
- Using LDAP change logging with TDBM
- NAS accessing SYS1.SIEALNKE
- EIM enhancements in z/OS V1R6
- Updates to our z/OS V1R5 test environment running WebSphere® Application Server
- Migrating to WebSphere for z/OS V5.*X* on z/OS V1R6

Changed information

- Our sysplex hardware configuration

This document contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Summary of changes
SA22-7663-11
June 2004

This document contains information previously presented in SA22-7663-10.

New information

- XCF REALLOCATE processing
- Using zSeries Application Assist Processors (zAAPs)
- IBM Health Checker for z/OS and Sysplex Version 3
- Migrating to WebSphere Business Integration Message Broker Version 5.0
- Implementing shared channels in a distributed-queuing management (DQM) environment
- Setting up a Kerberos peer trust relationship between z/OS and Windows 2000
- Using the z/OS LDAP client with the Windows 2000 Active Directory service
- Using LDAP with Kerberos authentication

Changed information

- Our sysplex hardware configuration

This document contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Summary of changes
for SA22-7663-10
March 2004

This document contains information previously presented in SA22-7663-09.

New information

- Migrating to z/OS V1R5 and z/OS.e V1R5
- Using DFSMS enhanced data integrity for sequential data sets
- Implementing the common service layer (CSL) and single point of control (SPOC) in IMS™ V8
- Using System Automation OS/390 (SA OS/390) V2R2
- Enhancements in z/OS UNIX in z/OS V1R5, including:
  - Remounting a shared HFS
  - Mounting file systems using symbolic links
  - Creating directories during z/OS UNIX initialization
  - Temporary file system (TFS) enhancements
- Setting up the LDAP server for RACF® change logging
- Support for additional bind types for EIM authentication
- Using WebSphere MQ shared queues and coupling facility structures
- Migrating to WebSphere for z/OS V5.0

Changed information

- Overview of our LDAP Server configuration

Deleted information

- System-managed coupling facility structure duplexing
- Verifying use of the cryptographic hardware
- Migrating to IMS Version 8 Release 1
- Using z/OS DFSMStvs
- Setting up the CICSPlex SM Web User Interface
- Using IBM HTTP Server
- LDAP Server enhancements in z/OS V1R4
- Using IMS Connect for z/OS Version 1.2
- Implementing the System SSL started task
- Using PKI Services
- Using WebSphere Studio Workload Simulator in z/OS Integration Test

Although the above information has been deleted from this edition, it continues to be available in our December 2003 edition.

Part 1. Parallel Sysplex

The above chapters describe the Parallel Sysplex® aspects of our computing environment.

Chapter 1. About our Parallel Sysplex environment

In this chapter we describe our Parallel Sysplex computing environment, including information about our hardware and software configurations and descriptions of the workloads we run.

Note: Throughout this document, when you see the term ***sysplex***, understand it to mean a sysplex with a coupling facility, which is a ***Parallel Sysplex.***

Overview of our Parallel Sysplex environment

We currently run a 13-member Parallel Sysplex that consists of the following:

- Four central processor complexes (CPCs) running z/OS in 13 logical partitions (LPs).

  The CPCs consist of the following machine types:
  - One IBM @server System z9
  - One IBM @server zSeries 990 (z990) processor
  - One IBM @server zSeries 900 (z900) processor
  - One IBM @server zSeries 890 (z890) processor

  The z/OS images consist of the following:
  - Eight production z/OS systems
  - One production z/OS.e system
  - Three test z/OS systems
  - One z/OS system to run TPNS (Our December 1998 edition explains why we run TPNS on a non-production system.)
- Three coupling facilities:
  - One failure-independent coupling facility that runs in a LP on a standalone CPC
  - Two non-failure-independent coupling facilities that run in LPs on two of the CPCs that host other z/OS images in the sysplex
- Two Sysplex Timer® external time references (ETRs)
- Other I/O devices, including ESCON- and FICON-attached DASD and tape drives.

The remainder of this chapter describes all of the above in more detail.

Outside of the Parallel Sysplex itself, we also have ten LPs in which we run the following:
- Two native Linux images
- Eight z/VM images that host multiple Linux guest images running in virtual machines

Our Parallel Sysplex hardware configuration

This section provides an overview of our Parallel Sysplex hardware configuration as well as other details about the hardware components in our operating environment.

Overview of our hardware configuration

Figure 1 on page 6 is a high-level, conceptual view of our Parallel Sysplex hardware configuration. In the figure, broad arrows indicate general connectivity

between processors, coupling facilities, Sysplex Timers, and other I/O devices; they do not depict actual point-to-point connections.

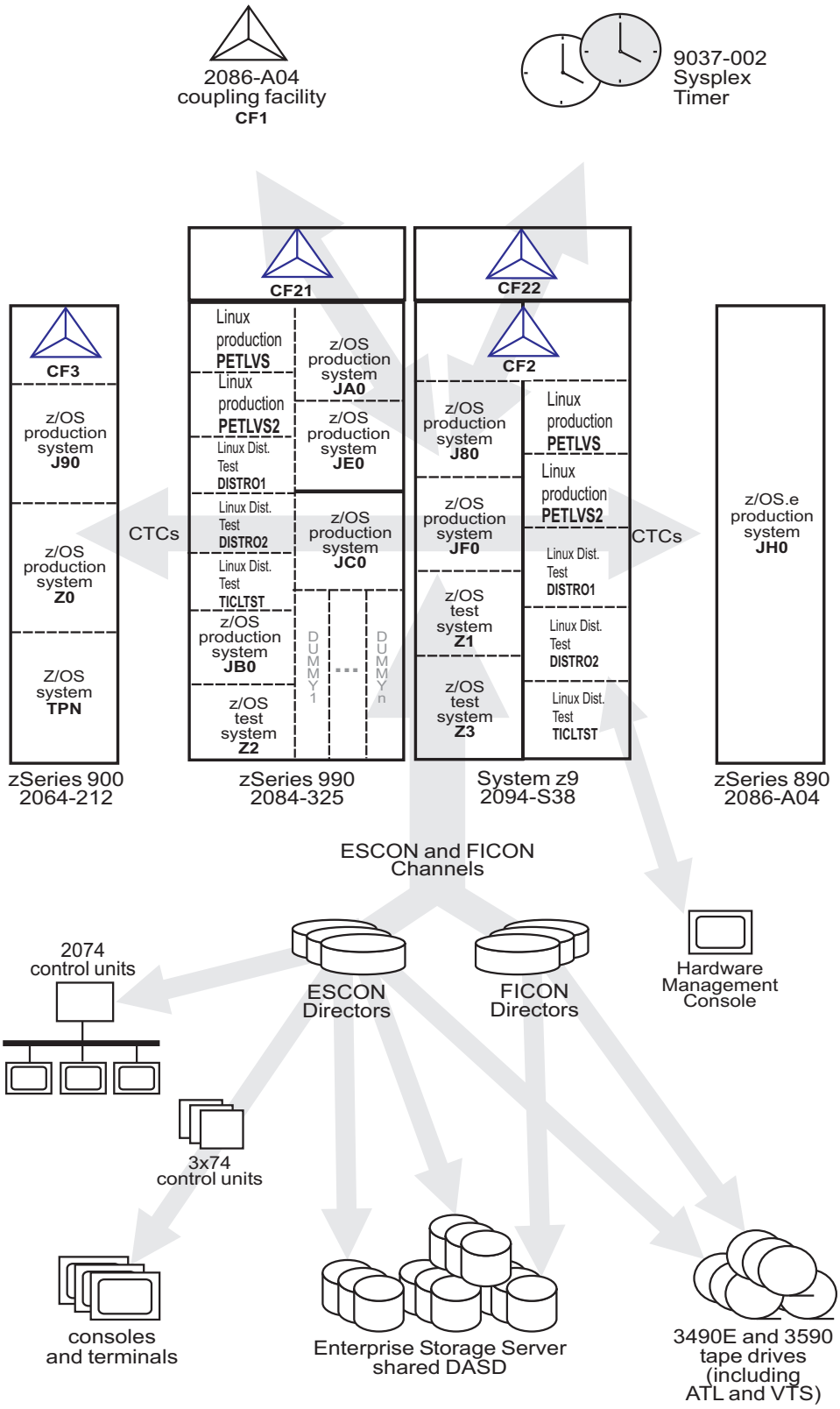


Figure 1. Our sysplex hardware configuration

Hardware configuration details

The figures and tables in this section provide additional details about the mainframe servers, coupling facilities, and other sysplex hardware shown in Figure 1 on page 6.

Mainframe server details

Table 2 provides information about the mainframe servers in our sysplex:

Table 2. Our mainframe servers

Server model (Machine type-model)	CPCs CPs	Mode LPs	HSA	Storage: Central Expanded	LCSS	System name, usage Virtual CPs (static, managed) Initial LPAR weight
IBM System @server z9 Model 104 (2094-S38)	1 CPC 38 CPs	LPAR 9 LPs	2112M	65536M	0-	**J80**, z/OS production system Plex 1 30 shared CPs, 2 shared zAAPs
				32768M	0	**JF0**, z/OS production system Plex 1 16 shared CPs, 2 SHARED zAAPS weight of 285
				16384M	0	**Z1**, z/OS test system Plex 2 10 shared CPs, 2 shared zAAPs weight of 145
				12288M	0	**Z3**, z/OS test system Plex 2 8 shared CPs, 2 shared zAAPs
				18432M	1	**PETLVS**, Linux production system shared CPs weight of 10
				4096M	1	**PETLVS2**, Linux production system 4 shared CPs weight of 10
				3072M	1	**DISTR01**, Linux distribution test system 2 Shared IFLs (Integrated Facility for Linux) weight of 10
				2048M	1	**DISTR02**, Linux distribution test system 2 Shared IFLs weight of 10
				1024M	1	**TICLTST**, Linux distribution test 1 shared IFL weight of 10
IBM @server zSeries 890 Model A04 (2086-A04) (see note 2 below)	1 CPC 4 CPs	LPAR mode 1 LP	1344M	14336M		**JH0**, z/OS.e production system Plex 1 3 shared CPs, 1 shared zAAP weight of 400

Table 2. Our mainframe servers (continued)

Server model (Machine type-model)	CPCs CPs	Mode LPs	HSA	Storage: Central / Expanded	LCSS	System name, usage Virtual CPs (static, managed) Initial LPAR weight
IBM @server zSeries 900 Model 212 (2064-212) (see note 1 below)	1 CPC 16 CPs (4 ICF)	LPAR mode 4 LPs (1 LP is a coupling facility)	256M	10240M		**J90**, z/OS production system Plex 1 8 shared CPs weight of 285
				9216M		**Z0**, z/OS production system Plex 1 8 shared CPs weight of 285
				6144M		**TPN**, z/OS system for TPNS Plex 1 12 shared CPs
IBM @server zSeries 990 Model 325 (2084-325)	1 CPC 32 CPs 2 IFL, 2 zAAP)	LPAR mode 20 LPs[3]	See note 4.	31G	2	**JA0**, z/OS production system Plex 1 16 shared CPs, 2 shared zAAPs
				31G	0	**JB0**, z/OS production system Plex 1 16 shared CPs, 2 shared zAAPs
				31G	0	**JC0**, z/OS production system Plex 1 16 shared CPs, 2 shared zAAPs
				31G	2	**JE0**, z/OS production system Plex 1 16 shared CPs, 2 shared zAAPs
				4096M	0	**Z2**, z/OS test system Plex 2 16 shared CPs, 2 shared zAAPs
				18432M	1	**PETLVS**, Linux production system 4 shared CPs weight of 10
				4096M	1	**PETLVS2**, Linux production system 4 shared CPs weight of 10
				3072M	1	**DISTR01**, Linux distribution test system 2 shared IFLs weight of 10
				2048M	1	**DISTR02**, Linux distribution test system 2 shared IFLs weight of 10
				1024M	1	**TICLTST**, Linux distribution test 1 shared IFL weight of 10

Notes:

1. For our z900 server, we applied the IYP version of IOCP 1.1.0, which is available with the fix for APAR OW46633 (PTF UW90695). We also applied the fix for HCD APAR OW43131 (PTFs UW99341, UW99342, UW99343, UW99344, UW99345) and the fix for HCM APAR IR43534 (PTFs UR90329 and UR90330).
2. Since z/OS.e is engine licensed, customers must define the MSU capacity of a z/OS.e LP to be on an engine boundary. To do this, IBM recommends using the

Defined capacity field in the activation profile on the z800 hardware management console (HMC). You must also send to IBM the Transmit System Availability Data (TSAD) for your z800 server, either by using the IBM Remote Support Facility (RSF) on the z800 or by mailing a diskette or DVD cartridge to IBM. For details, see *z/OS and z/OS.e Planning for Installation*, GA22-7504, and *z800 Software Pricing Configuration Technical Paper, GM13-0121*, available from the zSeries Library at www.ibm.com/servers/eserver/zseries/library/literature/.

3. We added several “dummy” logical partitions on our z990 server—LPs that are defined but not activated—in order to force the number of LPs to be greater than 15. Currently, you can define up to 30 LPs on the z990.
4. On the z990 and z890 support elements (SE), you no longer specify an HSA expansion percentage in the activation profile. Instead, the HSA size is now calculated from IOCP MAXDEV value.

Coupling facility details

Table 3 provides information about the coupling facilities in our sysplex. Figure 1 on page 6 further illustrates the coupling facility channel distribution as described in Table 3.

Table 3. Our coupling facilities

Coupling facility name	Model CPCs and CPs CFLEVEL (CFCC level) Controlled by	Storage: Central Expanded
CF1 (Plex 1)	zSeries 890 Model A04 (2086-A04) stand-alone coupling facility 1 CPC with 4 CPs CFLEVEL=14(CFCC Release 14.00, Service Level 00.17) Controlled by the HMC	6G
CF2 (Plex 1)	Coupling facility LP on a System z9 (2094-S38) 3 dedicated ICF CPs CFLEVEL=14(CFCC Release 14.00, Service Level 00.17) Controlled by the HMC	6G
CF3 (Plex 1)	Coupling facility LP on a zSeries 900 Model 212 (2064-212) 4 dedicated ICF CPs CFLEVEL=13 (CFCC Release 13.00, Service Level 04.08) Controlled by the HMC	6G
CF21 (Plex 2)	Coupling facility LP on a zSeries 900 Model 325 (2084-325) 1 dedicated ICF CP CFLEVEL=14 (CFCC Release 14.00, Service Level 00.25) Controlled by the HMC	6G
CF22 (Plex 2)	Coupling facility LP on a System z9 (2094-S38) 1 dedicated ICF CP CFLEVEL=14 (CFCC Release 14.00, Service Level 04.03) Controlled by the HMC	6G

Table 4 illustrates our coupling facility channel configuration on Plex 1.

Table 4. Our coupling facility channel configuration on Plex 1.

Coupling Facility Channel Connections on Plex 1				
		Coupling Facility (CF) Images		
		2086-A04	2094-S38	2064-212
		CF1	CF2	CF3
z/OS and CF Images				
2084-325 JA0, JE0, JC0, JB0, Z2		1 CBP 3 CFP	1 CBP 3 CFP	4 CBP 4 CFP
2086-A04 JH0		1 CBP 4 CFP	1 CBP 2 CFP	1 CBP 2 CFP
2064-212 Z0, J90, TPN, CF3		6 CFP	2 CBP *	4 ICP
2094-S38 J80, JF0, Z1, Z3, CF2		4 CFP	4 ICP	2 CBP *
			* = Same Links	

Table 5 illustrates our coupling facility channel configuration on Plex 2.

Table 5. Our coupling facility channel configuration on Plex 2.

Coupling Facility Channel Connections on Plex 2			
		Coupling Facility (CF) Images	
		2094-S38	2084-325
		CF22	CF21
z/OS and CF Images			
2084-325 Z2 CF21		4 ISC * 2 CBP*	2 ICP
2094-S38 Z1,Z3 CF22		2 ICP	4 ISC * 2 CBP*
		* = Same Links	

Other sysplex hardware details

Table 6 highlights information about the other hardware components in our sysplex:

Table 6. Other sysplex hardware configuration details

Hardware element	Model or type	Additional information
External Time Reference (ETR)	Sysplex Timer (9037-002 with feature code 4048)	We use the Sysplex Timer with the Expanded Availability feature, which provides two 9037 control units connected with fiber optic links. We don't have any Sysplex Timer logical offsets defined for any of the LPs in our sysplex.

Table 6. Other sysplex hardware configuration details (continued)

Hardware element	Model or type	Additional information
Channel subsystem	CTC communications connections	We have CTC connections from each system to every other system. We now use both FICON® and ESCON® CTC channels on all of our CPCs. **Note:** All of our z/OS images use both CTCs and coupling facility structures to communicate. This is strictly optional. You might choose to run with structures only, for ease of systems management. We use both structures and CTCs because it allows us to test more code paths. Under some circumstances, XCF signalling using CTCs is faster than using structures. See *S/390 Parallel Sysplex Performance* for a comparison.
	Coupling facility channels	We use a combination of ISC, ICB, and IC coupling facility channels in peer mode. We use MIF to logically share coupling facility channels among the logical partitions on a CPC. We define at least two paths from every system image to each coupling facility, and from every coupling facility to each of the other coupling facilities.
	ESCON channels	We use ESCON channels and ESCON Directors for our I/O connectivity. Our connections are “any-to-any”, which means every system can get to every device, including tape. (We do not use any parallel channels.)
	FICON channels	We have FICON native (FC) mode channels from all of our CPCs to our Enterprise Storage Servers and our 3590 tape drives through native FICON switches. (See *FICON Native Implementation and Reference Guide*, SG24-6266, for information about how to set up this and other native FICON configurations.) We maintain both ESCON and FICON paths to the Enterprise Storage Servers and 3590 tape drives for testing flexibility and backup. Note that FICON channels do not currently support dynamic channel path management. We have also implemented FICON CTCs, as described in the IBM Redpaper *FICON CTC Implementation* available on the IBM Redbooks™ Web site.
DASD	Enterprise Storage Server® (ESS, 2105-F20, 800, DS6000, DS8000)	All volumes shared by all systems; about 90% of our data is SMS-managed. We currently have four IBM TotalStorage® Enterprise Storage Servers, of which two are FICON only, and two that are attached with both ESCON and FICON. **Note:** Do not run with both ESCON and FICON channel paths from the same CPC to a control unit. We have some CPCs that are ESCON-connected and some that are FICON-connected.
Tape	3490E tape drives	16 IBM 3490 Magnetic Tape Subsystem Enhanced Capability (3490E) tape drives that can be connected to any system.
	3590 tape drives	4 IBM TotalStorage Enterprise Tape System 3590 tape drives that can be connected to any system.
Automated tape library (ATL)	3495 Model L40 with 16 additional 3490E tape drives and 12 3590 tape drives	All tape drives are accessible from all systems.
Virtual Tape Server (VTS)	3494 Model L10 with 32 virtual 3490E tape drives and 12 ESCON- and FICON-attached 3590 tape drives	All tape drives are accessible from all systems.

Exploiting 64k cylinder logical volumes

To fulfill our customers requirements for more space, more UCBs and our desire to simplify system administration with fewer devices, we implemented 3390 LVS (large volume support, greater then 32K cylinders volumes) on our IBM DS6000 and IBM DS8000. The first thing we did was to ensure that our ESS licensed internal code (LIC) and our z/OS system software supported this enhancement.

Our large volumes are SMS managed and were initialized with larger VTOCs, as follows:

```
//ICKDSF      EXEC PGM=ICKDSF
//SYSPRINT   DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//SYSIN           DD *
  INIT UNITADDRESS(6D00) DEVICETYPE(3390) PURGE NORECLAIM NOCHECK SG -
               NOVERIFY VOLID(LVS001) VTOC(2,0,120) INDEX(1,0,15)
```

We exploit dynamic Parallel Access Volume (PAV) and on average have a 3:1 ratio (3alias' to 1 base). For our large volumes we defined 12 aliases per base in the ESS SPECIALIST.

We use DFSMShsm to manage space and availability for the data on these volumes, and on all our DASD.

Applications in the zSeries Integration Test environment are exploiting these larger volumes. The applications include: HFS, zFS, DB2, CICS®, IMS, JES, HSM ML1, SVC dumps as well as stand-alone dumps (SAD).

Testing greater than 16 CPU Support

With the combination of z/OS V1R6, z990 servers, and the recommended APARs listed below, customers can now define a single z/OS LPAR image with up to 32 CPs. This function provides the customer more flexibility in choosing the way they want to grow:

- Horizontally with Parallel Sysplex, or
- Vertically using the greater than 16 CP support.

This function is released in two phases:

1. The general availability of z/OS V1R6 in combination with the z990 servers, support up to 24 processors.
2. Later in 2005, this function will be extended to support up to 32 processors.

We installed the following APARs before starting our test:

- OA08993
- OA05907
- OA07200
- OA07857
- OA09340
- OA09688

Our testing occurred in two phases:

1. **Phase 1:** One LPAR, dedicated CPs.

   On a z990 server, we defined only one z/OS LPAR, with dedicated CPs. On this image, we ran a high stress level of our IMS, CICS, DB2, MQ, OMVS and WebSphere workloads with a constant number of transactions. We started with 16 CPs online, and then varied CPs online in groups of 8, up to 32 total CPs, while maintaining the same level of transactions. We observed the CPU utilization value and found that it scaled as expected.
2. **Phase 2:** Two LPARs, shared CPs

   For this phase we defined two z/OS LPARs, each with 32 shared CPs. On one LPAR we ran the high stress IMS, CICS, DB2, MQ, OMVS and WebSphere

workloads. On the other LPAR we ran low priority workloads (batch and OMVS). The initial weights for the two LPARs were set differently according to the workloads (one higher and one lower).

On the high stress LPAR, we did a staging run where we gradually increased the number of transactions that the workloads were running. We started at low levels and slowly increased them until the LPAR was using more than 80% of the total CPU resource. This was done in order to see how WLM and Intelligent Resource Manager (IRD) would manage processor resources. When the high stress LPAR reached 80% CPU utilization, we observed that processors were taken away from the low priority LPAR and that the weights of both LPARs partitions were adjusted accordingly. The number of processors on the high stress LPAR remained at 32 as expected (the maximum allowed).

Here are examples of the RMF™ Monitor III screens that show the number of processors:

```
HARDCOPY     RMF V1R5   CPC Capacity                    Line 1
 Command ===>
0Samples: 120     System: J80   Date: 03/11/05  Time: 12.12.00  Range: 120
0Partition:   J80        2084 Model 332
 CPC Capacity:    1365   Weight % of Max: ****       4h MSU Average:   163
 Image Capacity:  1365   WLM Capping %:   ****       4h MSU Maximum:   234
0Partition  --- MSU ---  Cap  Proc    Logical Util %   - Physical Util % -
              Def   Act  Def   Num    Effect   Total   LPAR  Effect  Total
0*CP                                                    0.9    25.5   26.5
 J80            0   330  NO   32.0      23.8    24.1    0.3    23.8   24.1
 Z1             0    24  NO   32.0       1.7     1.8    0.0     1.7    1.8
 PHYSICAL                                               0.6            0.6
```

CPU Utilization reached 80% at J80:

```
Time gap from 03/11/05 12.27.00 to 03/11/05 12.27.40.
Samples: 60      System: J80   Date: 03/11/05  Time: 12.26.00  Range: 60
Partition:   J80        2084 Model 332
CPC Capacity:    1365   Weight % of Max: ****       4h MSU Average:   192
Image Capacity:  1365   WLM Capping %:   ****       4h MSU Maximum:  1009
Partition  --- MSU ---  Cap  Proc    Logical Util %   - Physical Util % -
             Def   Act  Def   Num    Effect   Total   LPAR  Effect  Total
*CP                                                    0.8    94.5   95.3
J80            0  1139  NO   31.0      85.5    86.1    0.5    82.9   83.4
Z1             0   159  NO   23.0      16.2    16.2    0.0    11.6   11.7
PHYSICAL                                               0.3            0.3
```

On/Off Capacity On Demand Testing

As part of IBM's On Demand strategy we tested On/Off Capacity On Demand with the following scenarios:

- z800 Concurrent upgrade testing
- z890 Capacity On Demand testing

z800 Concurrent upgrade testing: We converted our 2066 processor from a model 004 to a model A02. This model conversion was disruptive as you can not concurrently downgrade a z800 processor. At the time of this test we only had one LPAR (J80) on our z800. With all our standard workloads running we concurrently upgraded from a model A02 to a model 002, then to a model 003 and finally back to a model 004. We observed no disruption to our workloads. We did observe an expected decrease in CP utilization while at model types A02, 002, and 003. See Appendix B, "Some of our RMF reports," on page 333 for our RMF screen shots.

z890 Capacity on Demand Testing: With all our normal workloads running, we concurrently downgraded our 2086 processor from a model 370 to a model 360. At the time of this test we had two LPARs on our 2086(JG0,JH0). Each LPAR had two

general purpose processors and one zAAP processor configured. This downgrade decreased our MSU value from 158 to 91. This drop in MSU capacity only affected our general purpose processors, not our zAAP processors. We ran our 2086 as a model 360 for about two months until we concurrently upgraded our 2086 back to a model 370. During the duration of this test we had no disruptions or problems related to the concurrent downgrade or upgrade.

Our Parallel Sysplex software configuration

We run the z/OS operating system along with the following software products:

- CICS Transaction Server (CICS TS) V3R1
- IMS V9 (and its associated IRLM)
- DB2 UDB for z/OS and OS/390 V8 (and its associated IRLM)
- WebSphere for z/OS V6.0.2
- WebSphere MQ for z/OS V6
- Websphere Message Broker V6

We also run z/OS.e in one partition on our z890 server. z/OS.e supports next-generation e-business workloads; it does not support traditional workloads, such as CICS and IMS. However, z/OS.e uses the same code base as z/OS and invokes an operating environment that is identical to z/OS in all aspects of service, management, reporting, and zSeries functionality. See *z/OS.e Overview*, GA22-7869, for more information.

Note that we currently only run IBM software in our sysplex.

A word about dynamic enablement: As you will see when you read *z/OS and z/OS.e Planning for Installation*, z/OS is made up of base elements and optional features. Certain elements and features of z/OS support something called *dynamic enablement*. When placing your order, if you indicate you want to use one or more of these, IBM ships you a tailored IFAPRDxx parmlib member with those elements or features enabled. See *z/OS and z/OS.e Planning for Installation* and *z/OS MVS Product Management* for more information about dynamic enablement.

A note about IBM License Manager

In z/OS V1R1, IBM introduced a new base element called IBM License Manager (ILM). IBM has since decided not to deliver the IBM License Manager tool for zSeries. Therefore, when you run z/OS on a z800, z900, or z990 server, you must ensure that the ILMMODE parameter in IEASYS*xx* is set to `ILMMODE=NONE`.

Overview of our software configuration

Figure 2 on page 16 shows a high-level view of our sysplex software configuration.

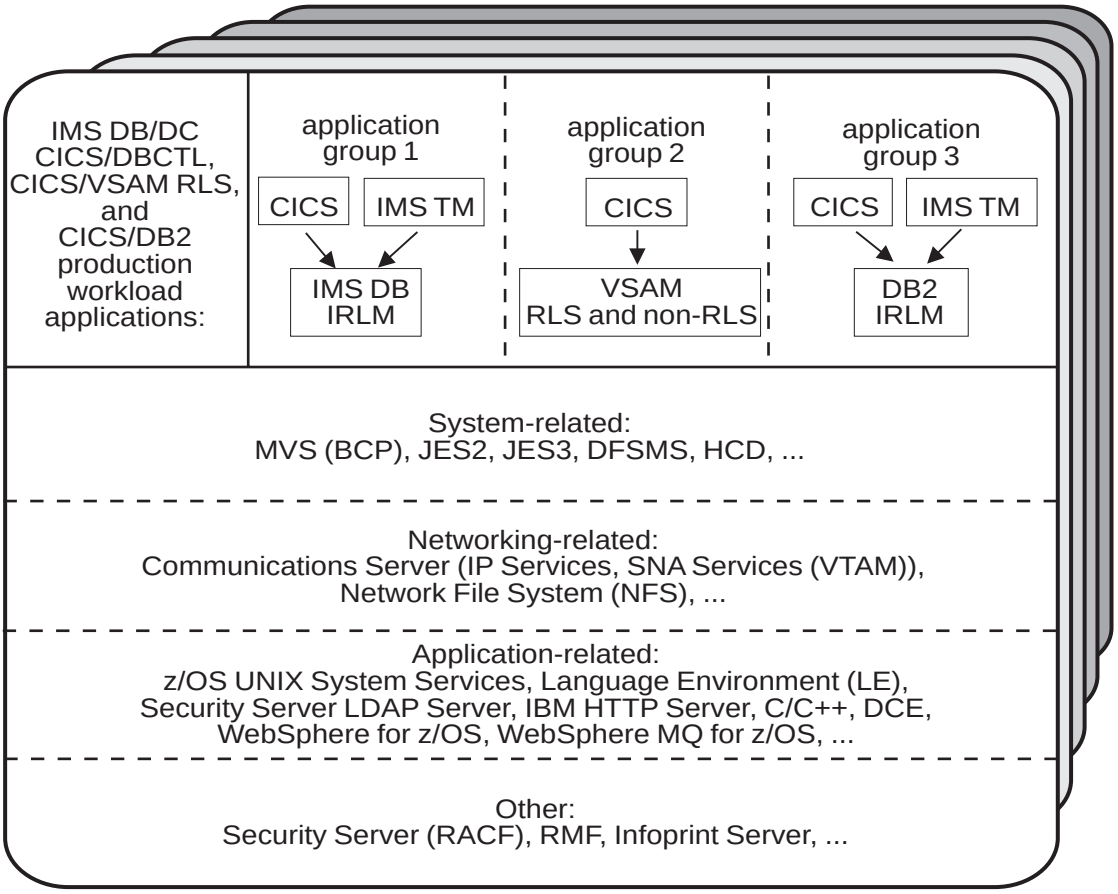


Figure 2. Our sysplex software configuration

We run three separate application groups in one sysplex and each application group spans multiple systems in the sysplex. Table 7 provides an overview of the types of transaction management, data management, and serialization management that each application group uses.

Table 7. Our production OLTP application groups

Application groups	Transaction management	Data management	Serialization management
Group 1	• CICS • IMS TM	IMS DB	IRLM
Group 2	• CICS	VSAM	VSAM record-level sharing (RLS)
Group 3	• CICS • IMS TM	DB2	IRLM

Our December 1995 edition describes in detail how a transaction is processed in the sysplex using application group 3 as an example. In the example, the transaction writes to both IMS and DB2 databases and is still valid for illustrative purposes, even though our application group 3 is no longer set up that way. For more information about the workloads that we currently run in each of our application groups, see “Database product OLTP workloads” on page 24.

About our naming conventions

We designed the naming convention for our CICS regions so that the names relate to the application groups and system names that the regions belong to. This is important because:

- Relating a CICS region name to its application groups means we can use wildcards to retrieve information about, or perform other tasks in relation to, a particular application group.
- Relating CICS region names to their respective z/OS system names means that subsystem job names also relate to the system names, which makes operations easier. This also makes using automatic restart management easier for us — we can direct where we want a restart to occur, and we know how to recover when the failed system is back online.

Our CICS regions have names of the form CICS*grsi* where:

- *g* represents the application group, and can be either 1, 2, or 3
- *r* represents the CICS region type, and can be either A for AORs, F for FORs, T for TORs, or W for WORs (Web server regions)
- *s* represents the system name, and can be 0 for system Z0, 8 for J80, 9 for J90, and A for JA0 through G for JG0
- *i* represents the instance of the region and can be A, B, or C (we have 3 AORs in each application group on each system)

For example, the CICS region named CICS2A0A would be the first group 2 AOR on system Z0.

Our IMS subsystem jobnames also correspond to their z/OS system name. They take the form IMS*s* where *s* represents the system name, as explained above for the CICS regions.

Our networking configuration

For a detailed description of our networking configuration, see Chapter 11, “About our networking and application enablement environment,” on page 125.

Our VTAM configuration

Figure 3 on page 18 illustrates our current VTAM® configuration.

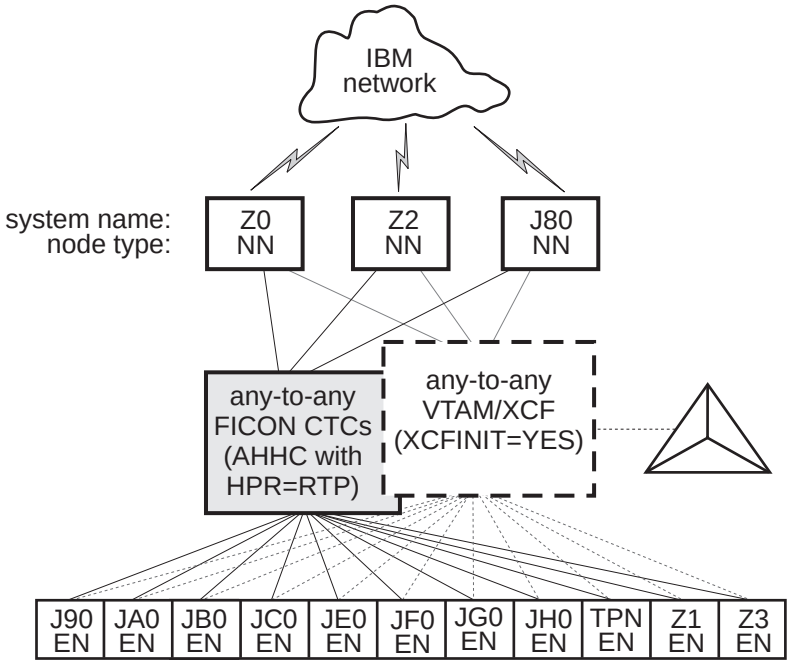


Figure 3. Our VTAM configuration

TPNS runs on our system TPN and routes CICS logons to any of the other systems in the sysplex (except JH0, which runs z/OS.e and does not support CICS).

Our VTAM configuration is a pure any-to-any AHHC. Systems Z0, Z2, and J80 are the network nodes (NNs) and the remaining systems are end nodes (ENs).

We also have any-to-any communication using XCF signalling, where XCF can use either CTCs, coupling facility structures, or both. This is called dynamic definition of VTAM-to-VTAM connections.

We are configured to use both AHHC and XCF signalling for test purposes.

Our workloads

We run a variety of workloads in our pseudo-production environment. Our workloads are similar to those that our customers use. In processing these workloads, we perform many of the same tasks as customer system programmers. Our goal, like yours, is to have our workloads up 24 hours a day, 7 days a week (24 x 7). We have workloads that exercise the sysplex, networking, and application enablement characteristics of our configuration.

Table 8 on page 19 summarizes the workloads we run during our prime shift and off shift. We describe each workload in more detail below.

Table 8. Summary of our workloads

Shift	Base system workloads	Application enablement workloads	Networking workloads	Database product workloads
Prime shift	• Automatic tape switching • Batch pipes • JES2/JES3 printer simulators	• Enterprise Identity Mapping (EIM) • IBM HTTP Server • LDAP Server • Kerberos Server • z/OS UNIX Shelltest (rlogin/telnet) • z/OS UNIX Shelltest (TSO) • WebSphere Application Server for z/OS • WebSphere MQ for z/OS • WebSphere Message Broker	• AutoWEB • FTP workloads • MMFACTS for NFS • NFSWL • Silk Test NFS video stream • TCP/IP CICS sockets • TN3270	• CICS DBCTL • CICS/DB2 • CICS/QMF online queries • CICS/RLS batch • CICS/RLS online • CICS/NRLS batch • CICS/NRLS online • DB2 Connect™ • DB2 online reorganization • DB2/RRS stored procedure • IMS AJS • IMS/DB2 • IMS full function • IMS SMQ fast path • QMF™ batch queries
Off shift	• Random batch • Automatic tape switching • JES2/JES3 printer simulators	• Enterprise Identity Mapping (EIM) • IBM HTTP Server • LDAP Server • Kerberos Server • z/OS UNIX Shelltest (rlogin/telnet) • z/OS UNIX Shelltest (TSO) • WebSphere Application Server for z/OS • WebSphere MQ for z/OS • WebSphere Message Broker	• FTP workloads • Silk Test NFS video stream • MMFACTS for NFS	• CICS /DBCTL • CICS/DB2 • CICS/RLS batch • CICS RLS online • CICS/NRLS batch • CICS/NRLS online • DB2 DDF • DB2 utility • IMS/DB2 • IMS utility • MQ/DB2 bookstore application • QMF online queries

Base system workloads

We run the following z/OS base (MVS) workloads:

BatchPipes®: This is a multi-system batch workload using BatchPipes. It drives high CP utilization of the coupling facility.

Automatic tape switching: We run 2 batch workloads to exploit automatic tape switching and the ATS STAR tape sharing function. These workloads use the Virtual Tape Server and DFSMSrmm™, as described in our December 1998 edition, and consist of DSSCOPY jobs and DSSDUMP jobs. The DSSCOPY jobs copy particular data sets to tape, while the DSSDUMP jobs copy an entire DASD volume to tape.

Both workloads are set up to run under Tivoli Workload Scheduler (TWS, formerly called OPC) so that 3 to 5 job streams with hundreds of jobs are all running at the same time to all systems in the sysplex. With WLM-managed initiators, there are no system affinities, so any job can run on any system. In this way we truly exploit the capabilities of automatic tape switching.

Tivoli Workload Scheduler (TWS) EXIT 51 tip:

Due to changes in JES2 for z/OS V1R7, TWS has made a new EXIT called EXIT51. TWS will only support TWS 8.1 or higher for z/OS V1R7 users. If you have z/OS V1R7 and use TWS 8.1 or higher you will need to:

- compile and linkedit your usual JES2/TWS EXITS
- compile and linkedit the new EXIT51.

EQQXIT51 is provided in the SEQQSAMP Lib. You will also need to add the following to both your JES2 PARM and existing OPCAXIT7 statement:

```
LOAD(TWSXIT51)
EXIT(51) ROUTINES=TWSENT51,STATUS=ENABLED
```

Once EXIT51 was installed and enabled we found no problems with our normal use of TWS 8.1.

JES2/JES3 printer simulators: This workload uses the sample functional subsystem (FSS) and the FSS application (FSA) functions for JES2 and JES3 output processing.

Random batch: This workload is a collection of MVS test cases that invoke many of the functions (both old and new) provided by MVS.

Application enablement workloads

We run the following application enablement workloads:

Enterprise Identity Mapping (EIM)

This workload exercises the z/OS EIM client and z/OS EIM domain controller. It consists of a shell script running on a z/OS image that simulates a user running EIM transactions.

HFS/zFS FILESYSTEM RECURSIVE COPY/DELETE

This TPNS driven workload copies over 700 directories from one large filesystem to another. It then deletes all directories in the copy with multiple remove (rm) commands.

IBM HTTP Server

These workloads are driven from AIX/RISC workstations. They run against various HTTP server environments, including the following:

- HTTP scalable server
- HTTP standalone server
- Sysplex distributor routing to various HTTP servers

These workloads access the following:

- MVS datasets
- FastCGI programs
- Counters
- Static html pages
- Static pages through an SSL connection
- REXX Exec through GWAPI
- Protection through RACF userid
- Sysplex Distributor
- Standalone http server
- Scalable http server

ICSF

This workload runs on MVS. It is run by submitting a job through TSO. This one job kicks off 200+ other jobs. These jobs are set up to use ICSF services to access the crypto hardware available on the system. The goal is to keep these jobs running 24/7.

LDAP Server

LDAP Server consists of the following workloads:

- Segue Silk Performer - is setup on a remote Windows NT machine. The workload is setup to run a Performer Script for 20 users. The script is designed to issue several LDAP commands (ldapsearch, ldapadd, ldapdelete) issued to the z/OS LDAP server. At the start of the workload simulation, each virtual user is setup to have a 15 second delay between executing the script, thus making the simulation more ″customer like″. This workload simulation is then executed on a 24/7 basis.
- Tivoli Access Manager - Tivoli Access Manager uses z/OS LDAP to store user information. The workload that is executed is a shell script that consists of several TAM user admin commands that places stress on the TAM/LDAP environment.
- Mindcraft Workload Simulator - The DirectoryMark benchmark is designed to measure the performance of server products that use LDAP, We have this product installed on a Windows server machine. Scripts generated by DirectoryMark are run against z/OS LDAP on a 24/7 basis.
- Authentication - This workload is driven from an AIX/RISC workstation. It runs against the IBM HTTP Server on z/OS and Apache on Linux to provide LDAP authentication when accessing protected resources.

NAS (kerberos)

This workload runs from the shell as a shell script. It uses both the z/OS LDAP and z/OS EIM client to bind through kerberos with EIM and LDAP.

z/OS UNIX Shelltest (rlogin/telnet)

In this workload, users log in remotely from an RS/6000® workstation to the z/OS shell using either rlogin or telnet and then issue commands.

z/OS UNIX Shelltest (TSO)

In this workload, simulated users driven by the Teleprocessing Network Simulator (TPNS) logon to TSO/E and invoke the z/OS UNIX shell and issue various commands. The users perform tasks that simulate real z/OS UNIX users daily jobs, for example:

- Moving data between the HFS and MVS data sets.
- Compiling C programs.
- Running shell programs.

WebSphere Application Server for z/OS

We run a number of different Web application workloads in our test environment on z/OS. Generally, each workload drives HTTP requests to Web applications that consist of any combination of static content (such as HTML documents and images files), Java™ Servlets, JSP pages, and Enterprise JavaBeans™ (EJB) components. These Web applications use various connectors to access data in our DB2, CICS, or IMS subsystems.

Our Web application workloads currently include the following:

- J2EE applications (including persistent (CMP and BMP) and stateless session EJB components) that:

  - Access DB2 using JDBC
  - Access CICS using the CICS Common Client Interface (CCI)
  - Access IMS using the IMS Connector for Java CCI
  - Access WebSphere MQ using Java Message Service (JMS)
  - Access Websphere MQ and the Websphere Message Broker
- Non-J2EE applications (only static resources, Servlets, and JSP pages) that:
  - Access DB2 using JDBC
  - Access CICS using CICS CTG
  - Access IMS using IMS Connect
- Other variations of the above applications, including those that:
  - Access secure HTTPS connections using SSL
  - Perform basic mode authentication
  - Use HTTP session data
  - Use connection pooling
  - Use persistent messaging
  - Use RACF or LDAP for Local OS security
  - Use WebSphere Network Deployment (ND) configuration(s)
  - Utilize Sysplex Distributor
  - Use HTTP Server / J2EE Server clustering
  - Use DB2 Legacy RRS / DB2 UDB JCC driver(s)

WebSphere MQ for z/OS workloads

Our WebSphere MQ environment includes one WebSphere MQ for z/OS queue manager on each system in the sysplex. We have two queue sharing groups: one with three queue managers and another with four queue managers.

Our workloads test the following WebSphere MQ features:
- CICS Bridge
- Distributed queueing with APPC, SSL, and TCP/IP channels
- Large messages
- Shared queues
- Clustering
- Transaction coordination with RRS

We use the following methods to drive our workloads (not all workloads use each method):
- Batch jobs
- Web applications driven by WebSphere Studio Workload Simulator
- TPNS TSO users running Java programs through z/OS UNIX shell scripts

The batch-driven workloads that use WebSphere MQ for z/OS include the following:

MQ batch stress for non-shared queues: This workload runs on one system and stresses WebSphere MQ for z/OS by issuing MQI calls. These calls include a variety of commands affecting local queues.

MQ batch stress for shared queues: This workload runs on one system and stresses WebSphere MQ for z/OS by issuing MQI calls. These calls include a variety of commands affecting shared queues. Workload parameters control the number of each type of call.

DQM and DQMssl: These workloads test the communication between z/OS queue managers using SSL TCPIP channels and non-SSL APPC channels. The application puts messages on remote queues and waits for replies on its local queues.

MQCICS: This workload uses the MQ CICS bridge to run a transaction that updates a DB2 parts table. The CICS bridge request and reply queues are local queues that have persistent messages. We also have a non-Web version of MQCICS that uses shared cluster queues with persistent messages. We defined a separate coupling facility structure for this application.

MQLarge: This workload tests various large message sizes by creating temporary dynamic queues and putting large messages on those queues. Message sizes vary from 1MB to 100MB starting in increments of 10MB. The script running the application randomly chooses a message size and passes this to the mqLarge program. mqLarge then dynamically defines a queue using model queues that have their maxmsgl set to accommodate the message.

WebSphere Message Broker

Our WebSphere Message Broker environment consists of five message brokers: three on test systems, and two on production systems. All are running Websphere Message Broker v6.0. We will refer to this broker version as WMB. We use the following methods to drive our workloads (not all workloads use each method):

- Web applications driven by WebSphere Studio Workload Simulator
- Batch jobs
- TPNS TSO users running Java programs through z/OS UNIX shell scripts

The Web applications consist of html pages, java servlets, and message flows to process the messages. These Java-based workloads have recently been converted to use Websphere Application Server 5.1 instead of the IBM HTTP Server with the WebSphere V4.0 plugin.

Retail_IMS: This workload tests message manipulation by taking a message, extracting certain fields from it, and adding an IMS header.

Retail_Info: This workload tests inserting and deleting fields from a message into a simple DB2 table.

Retail_Wh: This workload tests inserting and deleting an entire message (using a data warehouse node) into a LOB DB2 table.

We have two batch-driven workloads:

Sniffer: This workload tests basic MQ and broker functionality using persistent and non-persistent messages. It is based on SupportPac™ IP13: Sniff test and Performance on z/OS. (See http://www-306.ibm.com/software/integration/support/supportpacs/category.html#cat1)

Football: This workload tests basic broker publish/subscribe functionality. Using the Subscribe portion of the workload, a subscription is registered with the broker. The Publish portion publishes messages to the broker, which then routes them to the matching subscribers. Like the Sniffer workload, this workload is based on SupportPac IP13.

We have one TPNS workload that uses WMB:

Retail_TPNS: This workload is another version of Retail_IMS, but rather than being driven by WebSphere Studio Workload Simulator, it is driven by TPNS through z/OS UNIX shell scripts.

Networking workloads

We run the following networking workloads:

FTP workloads:

- **FTPHFS/DB2:** This client/server workload simulates SQL/DB2 queries through an FTP client.
- **FTPHFS(Linux):** This workload simulates users logging onto a Linux client through telnet or FTP and simulates workloads between the z/OS servers and the LINUX client.
- **FTP TPNS:** This workload uses TPNS to simulate FTP client connections to the z/OS server.
- **FTPWL:** This client/server workload automates Linux clients performing FTP file transfers across Token Ring and Ethernet networks. This workload also exercises the z/OS Domain Name System (DNS). Files that are transferred reside in both z/OS HFS and MVS non-VSAM data sets. Future enhancements to this workload will exploit the z/OS workload manager DNS.

MMFACTS for NFS: This client/server workload is designed to simulate the delivery of multimedia data streams, such as video, across the network. It moves large volumes of randomly-generated data in a continuous, real-time stream from the server (in our case, z/OS) to the client. Data files can range in size from 4 MB to 2 Gigabytes. A variety of options allow for variations in such things as frame size and required delivery rates.

NFSWL: This client/server workload consists of shell scripts that run on our AIX clients. The shell script implements reads, writes, and deletes on an NFS mounted file system. We mount both HFS and zFS file systems that reside on z/OS. This workload is managed by a front end Web interface.

AutoWEB: This client/server workload is designed to simulate a user working from a Web Browser. It uses the following HTML meta-statement to automate the loading of a new page after the refresh timer expires:

```
<meta http-equiv='Refresh' content='10; url=file:///filename.ext'>
```

This workload can drive any file server, such as LAN Server or NFS. It also can drive a Web Server by changing the URL from `url=file:///filename.ext` to `url=http://host/filename.ext`.

Silk Test NFS video stream: This client/server workload is very similar to that of MMFACTS except that it sends actual video streams across the network instead of simulating them.

TCP/IP CICS sockets: This TPNS workload exercises TCP/IP CICS sockets to simulate real transactions.

TN3270: This workload uses TPNS to simulate TN3270 clients which logon to TSO using generic resources. This workload exploits Sysplex Distributor.

Database product workloads

Database product OLTP workloads

Our sysplex OLTP workloads are our mission critical, primary production workloads. Each of our 3 application groups runs different OLTP workloads using CICS or IMS as the transaction manager:

- Application group 1—IMS data sharing, including IMS shared message queue
- Application group 2—VSAM record level sharing (RLS) and non-RLS
- Application group 3—DB2 data sharing (four different OLTP workloads, as well as several batch workloads).

Note that our OLTP workloads, which are COBOL, FORTRAN, PL1, or C/C++ programs, are Language Environment® enabled (that is, they invoke Language Environment support).

IMS data sharing workloads: In application group one, we run three IMS data sharing workloads:
- CICS/DBCTL
- IMS SMQ Fast Path
- IMS SMQ full function
- IMS automated job submission (AJS)

Highlights of our IMS data sharing workloads include:
- Full function, Fast Path, and mixed mode transactions
- Use of virtual storage option (VSO), shared sequential dependent (SDEP) databases, generic resources, and High Availability Large Databases (HALDB)
- Integrity checking on INSERT calls using SDEP journaling
- A batch message processing (BMP) application to do integrity checking on REPLACE calls
- A set of automatically-submitted BMP jobs to exercise the High-Speed Sequential Processing (HSSP) function of Fast Path and the reorg and SDEP scan and delete utilities. This workload continuously submits jobs at specific intervals to run concurrently with the online system. We enhanced this workload based on recent customer experiences to more closely resemble a real-world environment.

VSAM/RLS data sharing workload: In application group 2, we run one OLTP VSAM/RLS data sharing workload. This workload runs transactions that simulate a banking application (ATM and teller transactions). The workload also runs transactions that are similar to the IMS data sharing workload that runs in application group 1, except that these transactions use VSAM files.

VSAM/NRLS workload: Also in application group 2, we added two new workloads. One uses transactions similar to our VSAM/RLS workload but accessing VSAM non-RLS files. The other is a very I/O-intensive workload that simulates a financial brokerage application.

DB2 data sharing workloads: In application group 3, we run four different DB2 data sharing OLTP workloads. These workloads are also similar to the IMS data sharing workload running in application group 1.

In the first of the DB2 workloads, we execute 8 different types of transactions in a CICS/DB2 environment. This workload uses databases with simple and partitioned table spaces.

In the second of our DB2 workloads, we use the same CICS regions and the same DB2 data sharing members. However, we use different transactions and different databases. The table space layout is also different for the databases used by the second DB2 workload—it has partitioned table spaces, segmented table spaces, simple table spaces, and partitioned indexes.

Our third workload is a derivative of the second, but incorporates large objects (LOBs), triggers, user defined functions (UDFs), identity columns, and global temporary tables.

The fourth workload uses IMS/TM executing 12 different transaction types accessing DB2 tables with LOBs. It also excercises UDFs, stored procedures and global temporary tables.

Database product batch workloads

We run various batch workloads in our environment, some of which we will describe here. They include:

- IMS Utility
- RLS batch (read-only) and TVS batch
- DB2 batch workloads

We run our batch workloads under TWS control and use WLM-managed initiators. Our implementation of WLM batch management is described in our December 1997 edition.

DB2 batch workloads: Our DB2 batch workloads include:

- DB2 Online reorganization
- DB2/RRS stored procedure
- QMF batch queries
- DB2 utilities
- DB2 DDF

Our DB2 batch workload has close to 2000 jobs that are scheduled using TWS, so that the jobs run in a certain sequence based on their inter-job dependencies.

WebSphere MQ / DB2 bookstore application

Our multi-platform bookstore application lets users order books or maintain inventory. The user interface runs on AIX, and we have data in DB2 databases on AIX and z/OS systems. We use WebSphere MQ for z/OS to bridge the platforms and MQ clustering to give the application access to any queue manager in the cluster. See our December 2001 edition for details on how we set up this application.

Creating a split plex for production and test

Due to the robust test environment we provide we have been participating in an increasing number of projects which by their nature introduce increased instability and outages during the early testing phase. This is placing pressure on achieving one of our primary goals which is 24 * 7 availability. Down time for businesses in the real world costs money in terms of lost revenue and unhappy customers. For us, down time results in idle or unproductive time for the different test teams utilizing our environment. Creating a flexible environment which can accommodate disruptive testing and also provide high availability is a requirement to meet our current and future commitments. So we needed to look at our environment and see where we have been and where we need to go.

The following sections describe:

- “Our plex history” on page 27
- “Splitting the plex” on page 27
- “Planning and defining our future production and test plexes” on page 27
- “Executing the steps for our plex split” on page 29

Our plex history

After our production plex was created back in 1995 the need arose to have an MVS image where IPL tests and changes could be executed without affecting the production images. System Z1 (seen in Figure 1 on page 6) was created to fill this role. As the application environment grew over time the need to create a "sub-plex" to provide an environment for separate data sharing groups, but still contained within the same sysplex became a necessity. Systems Z2 and Z3 (also seen in Figure 1 on page 6) were created to fill this role. Today Z1, Z2, and Z3 provide an environment where we have a logical separation between test and production applications but do not provide the production images with true isolation.

Splitting the plex

We determined that by taking the Z1/Z2/Z3 "sub-plex" to the next level of isolation (a separate plex) we would be able to provide the flexible environment needed to meet our expanding commitment base. For the rest of this section Plex 1 is the current plex which would lose the test images and Plex 2 is the new plex in which the test images will reside.

This setup and its use actually reflects how many customers handle similar issues. Most large customers support multiple plex's in their environments today. The intent of one of these plex's is for the introduction of new hardware/software, new functions, and to test proposed changes. This is how we intend to use this smaller plex.

So how did we go about splitting our plex? Well the first thing we did was build a project plan to track the tasks and activities. We broke the list of items up into pre split tasks found in "Planning and defining our future production and test plexes," the actual split itself found in "Executing the steps for our plex split" on page 29, and then post split tasks found in "Executing post plex split steps" on page 30. Here is a copy of the task plan that we utilized.

Note: Our databases for IMS and DB2 were already separate from a test and production standpoint so we did not have to address the breaking of the databases which can be an issue for customers. There was some work required from an application standpoint to break out dataset names, and so forth.

Planning and defining our future production and test plexes

We had to define how we were going to use our second smaller plex (Plex 2). This included:

Changing our hardware and software implementation

Systems Z1, Z2, and Z3 will be used as early introduction points for new hardware, software, function, and workload implementation. The criteria for moving to production will be established on a project by project basis. For many of our software upgrades this is how we utilize these systems today. **The plex is not designed to be a high stress environment.**

Software service, ++apars, and usermods will be tested first in Plex 2. After such time that we deem this new service to be acceptable (typically 2 days), we will propagate it to Plex 1.

Configuring our second plex (Plex 2)

Next we needed to decide what the configuration of the Plex 2 would be.

Our configuration of Plex 2 consists of:

- 2 CFs and 3 z/OS images
- JES2 only (We can setup JES3 when needed.)
- A shared DASD pool between both plex's for common program products
- All subsystems, workloads, tools that currently reside on Z1, Z2, and Z3
- All hardware functions (Crypto, OSA, zAAP, and others) that are available currently to Z1, Z2, and Z3
- All systems will continue to be connected to all devices. We will utilize MVS to vary the required devices offline
- DFSMS/HSM environments will be unique to both plex's

We then decided where the z/OS images and coupling facilities would reside. Refer to Figure 1 on page 6. We left Z1 and Z3 on our z9, Z2 on our z990, and added a CF to each of those processors. The systems will be capped so that they don't take resource away from the production environments.

Defining our build strategy for both plexes

Our SMP/E environment is managed by a separate support team. We had to come up with a strategy that would use only one SMP/E environment for both plexes so that we did not put additional requirements on that support team. We could easily share the sysres between the 2 plexes, but we needed to have separate version root filesystems. This is for data integrity, because the version root filesystem is able to be mounted R/W. Our strategy at a high level is simple. We restore from dump and customize the version root twice (onto different sets of DASD) - once for each plex.

The following tasks were performed just prior to ipling the first image in the new plex. Changes or updates were frozen.

- Defined/copied RACF databases – this was completed the day before we split the plex
- Defined and preloaded the CFRM Policy
- Moved/converted filesystems to z/FS – We could have copied HFSs over but we decided to convert our HFSs over to z/FS on Plex 2.

Software setup steps for both plexes

This work was completed over the course of several months. Both plexes have separate master catalogs and user catalog structures except for one shared user catalog between both plex's which is for non SMS managed data that resides in the shared pool.

The following are the steps we took in preparing for the split:

- Identified the application test data that would either be shared between the plexes or isolated to Plex 1 and Plex 2. Duplicate high level qualifiers were identified.
- Identified required system datasets for Plex 2. System datasets that were currently being utilized by the Z1, Z2, and Z3 images needed to be copied over

to the Plex 2 specific DASD or the new shared pool. These included LPA, Linklist, and Procs (datasets within procs). As a result of these assessments the following tasks were performed:

  - Defined and created the shared pool space requirements
  - Defined and propogated Plex 2 usercats (user catalogs) with aliases for Z1, Z2, and Z3
  - Defined Plex 2 Storage Groups to the Plex 1 ACS routine. We then either copied/renamed Plex 2 libraries (system or application) to the Plex 2 specific volumes or moved the datasets that needed to be shared to the share pool
- Separated our JES2 MAS weeks before we split the plex and created a new JES2 node. We started with a fresh spool and cold started to bring in the new MAS and Checkpoints. We ipled one image at a time. Our daily spool cleanup on Plex 1 eventually cleaned up any output hanging around from the Plex 2 images after the MAS split although the image definitions still exist.
- Created 2 new CFs for Plex 2. One dedicated CP for each
- Developed our proclib concatenation strategy for both plexes
- Created a new parmlib for Plex 2 called Sys1.Plex 2.Parmlib to be concatenated ahead of SYS1.PARMLIB which will be shared.
- Defined new sysplex name for Plex 2
- Defined new Plex 2 generic resource name
- Copied, moved test libraries and data to Plex 2 specific volumes or the shared pool
- Defined couple datasets. This included ARM, SFM, Sysplex, CFRM, WLM, and Logger

The following tasks were performed just prior to ipling the first image in the new plex. Changes or updates were frozen.

- Defined/copied RACF databases – this was completed the day before we split the plex
- Defined and preloaded the CFRM Policy
- Moved/converted filesystems to z/FS – We decided to convert our HFSs over to z/FS on Plex 2 so we could have more exposure with both filesystem types.

Executing the steps for our plex split

We executed the following steps one at a time. Unless otherwise specified all subsequent tasks were performed on Plex 2. Following are the steps we took:

- Activated the Plex 2 Coupling Facilities
- Ipled the first image (not in monoplex mode since we were able to predefine and load the CFRM Policy). We also predefined and loaded the ARM and SFM policies
- Started all MVS Base component subsystems , LLA, VTAM, RMF, TSO, SDSF, and others. through the Sys1.Plex 2.Parmlib(COMMNDxx) member
- Loaded, activated, and verified the following policies:
  - WLM policy from Plex 1
  - Logger policy
- Re-IPL'd our first image to verify all policies activated successfully at IPL time
- IPL'd the remaining two images
- Activated subsystems (for example; DB2, CICS and IMS) one at a time and any startup problems were resolved.

- Brought up workloads

Executing post plex split steps

Following are the post plex split steps we took:

- Created our Netview/SA environment. Prior to activation (or implementation) we ran several weeks in manual mode to ensure all applications started successfully. These products do, however, provide the ability to set up policies ahead of time.
- Created and activated our TWS environment
- Defined and activated RMM
- Created and activated new HSM environment
- Updated our IODF to remove test images from chpids on Plex 1
- Cleaned up both plexes Sys1.Parmlib
- Reviewed and updated procedures for Operations
- Cleaned up Plex 1 and Plex 2 Policies and the TWS daily plan
- Separated Plex 1 and Plex 2's DASD
- Defined the promotion process for changes from Plex 2 to Plex 1.

Results of our plex split

We have now taken the separation of our test and production environments to the next level. This allows us to be more flexible with additional testing requests, yet still maintain our focus on high availability. Overall the split of the one sysplex into two went very smooth.

Chapter 2. About our security environment

In this chapter we describe our security computing environment, including information about:

- "Our Integrated Cryptographic Service Facility (ICSF) configuration"
- "RACF Security Server mixed case password support"
- "Testing the IBM Encryption Facility for z/OS" on page 32

Our Integrated Cryptographic Service Facility (ICSF) configuration

z/OS Integrated Cryptographic Service Facility (ICSF) is a software element of z/OS that works with the hardware cryptographic features and the Security Server (RACF) to provide secure, high-speed cryptographic services in the z/OS environment. ICSF provides the application programming interfaces by which applications request the cryptographic services. The cryptographic feature is secure, high-speed hardware that performs the actual cryptographic functions.

The available cryptographic hardware features are dependent on the server.

In our sysplex, we are currently running ICSF, FMID HCR7730, on top of z/OS V1R7. Because we have many types of servers in our environment, we run with various cryptographic hardware features. Following is a list of cryptographic hardware features we currently have:

- z9-EC:
  - Crypto Express2 Accelerator (CEX2A)
  - Crypto Express2 Coprocessor (CEX2C)
- Z990:
  - PCI Cryptographic Accelerator (PCICA)
  - PCI X Cryptographic Coprocessor (PCIXCC)
  - Crypto Express2 Coprocessor (CEX2C)
- z9-BC:
  - Crypto Express2 Coprocessor (CEX2C)
- Z900:
  - Cryptographic Coprocessor Feature (CCF)
  - PCI Cryptographic Accelerator (PCICA)

Since our goal is to run a customer-like environment, we have various products running customer-like scenarios using SSL. SSL will in turn use ICSF and any of the Cryptographic Features that we have, as needed. The products that use SSL in our environment are z/OS WebSphere Application Server, FTP, HTTP, LDAP, and CICS. We also have an ICSF specific workload that runs 16 hours a day, 7 days a week and exercises the cryptographic services available through the ICSF API.

RACF Security Server mixed case password support

With the release of V1R7, RACF Security Server now supports mixed case passwords. The maximum length of the password remains at eight(8) characters. To turn on mixed case support, a MVS security administrator would enter the following command:

```
SETROPTS PASSWORD(MIXED)
```

to turn it off, the command is

```
SETROPTS PASSWORD(NOMIXED)
```

You should only turn this on if you really need this support and you are sure that all of your applications support mixed case passwords. If for some reason you have to turn mixed case support off, all passwords that were created during the time period that support was on will have to be reset.

After implementing MIXEDCASE, if a password is SET (through ADDUSER) without any lower-case characters, then RACF will not require exact case matching. If the user then changes their password to one without any lower-case characters, RACF still won't enforce exact case matching.

Once a user changes their password to include lower-case characters, RACF will enforce case matching.

Currently, the following CS/390 clients and servers now support mixed case passwords.

FTP
TN3270
POP
USS Telnet
TSO

Testing the IBM Encryption Facility for z/OS

We recently brought the IBM Encryption Facility for z/OS into our environment to test. The Encryption Facility provides encryption and decryption processing of data for exchange between different systems and platforms and for archiving purposes. It makes use of hardware compression and encryption and relies on a centralized key management based on the z/OS Integrated Cryptographic Service Facility (ICSF). Encryption Facility consists of the following optional features:

- Encryption Facility Encryption Services for z/OS
- Encryption Facility DFSMSdss/DFSMShsm Encryption

A licensed Java reference program called Encryption Facility for z/OS Client is also downloadable from the Web.

We used the IBM Encryption Facility for z/OS: User's Guide throughout our testing, located at:

```
http://publibz.boulder.ibm.com/epubs/pdf/csda1101.pdf
```

Using the Encryption Facility Encryption Services for z/OS feature, we encrypted data to both 3390-type DASD and 3590 tapes using a header with enough information to recover and decrypt the data. We used the CSDFILEN batch program to encrypt the data and the CSDFILDE batch program to decrypt the data. Sample jobs to encrypt and decrypt data can be found in the *IBM Encryption Facility for z/OS: User's Guide*.

Using the DFSMSdss Encryption feature and DFSMSdss DUMP command, we encrypted and decrypted data to 3590 tape and 3390 DASD. DFSMShsm is an additional feature of DFSMSdss Encryption. It allowed us to encrypt our full volume dumps created through the *hsm* BACKVOL DUMP Command. We indicated whether or not encryption is to be performed through the dump class definition in

our SYS1.PARMLIB(ARCCMDxx). Sample jobs to run DFSMSdss Encryption features can found in the *IBM Encryption Facility for z/OS: User's Guide*.

For your viewing pleasure, we've provided samples of the JCL we used during our test. The samples can be found in the samples section of this web site at:

`http://www.ibm.com/servers/eserver/zseries/zos/integtst/samples.html`

For additional information, there is an article in the January 2006, Issue 14 of the HOT TOPICS Newsletter which gives the reader a high level overview of why you would want to use the Encryption Facility, what the features entail and how to use them. This Newsletter can be found at:

`http://www.ibm.com/zseries/zos/bkserv/hot_topics.html`

Chapter 3. Migrating to and using z/OS

This chapter describes our experiences with migrating to new releases of the z/OS operating system.

Overview

The following sections describe our most recent migration activities:

- "Migrating to z/OS V1R7"
- "Migrating to z/OS.e V1R7" on page 38
- "Migrating to z/OS V1R6" on page 41
- "Migrating to z/OS.e V1R6" on page 43

We primarily discuss our sysplex-related base operating system experiences. This includes the enablement of significant new functions and, if applicable, performance aspects. Detailed test experiences with major new functions beyond migration appear in subsequent chapters.

We discuss our networking and application-enablement environment and test experiences in Part 2, "Networking and application enablement," on page 121.

You can read about our migration experiences with earlier releases of z/OS and OS/390 in previous editions of our test report, available on our Web site:

For migration experiences with...	See...
z/OS V1R6	our December 2003 edition
z/OS.e V1R5	our December 2003 edition
z/OS V1R5	our December 2003 edition
z/OS V1R4	our December 2003 edition
z/OS V1R3	our December 2002 edition
z/OS V1R1 and V1R2	our December 2001 edition

Migrating to z/OS V1R7

This section describes our migration experiences with z/OS V1R7.

z/OS V1R7 base migration experiences

In this section we described our experiences with our base migration to z/OS V1R7, without having implemented any new functions. It includes our high level migration process along with other migration activities and considerations.

Our high-level migration process for z/OS V1R7

The following is an overview of our z/OS V1R7 migration process.

Before we began: We reviewed the migration information in *z/OS and z/OS.e Planning for Installation*, GA22-7504 and *z/OS Migration*.

Table 9 on page 36 shows the high-level process we followed to migrate the members of our sysplex from z/OS V1R6 to z/OS V1R7.

Table 9. Our high-level migration process for z/OS V1R7

Stage	Description
Updating parmlib for z/OS V1R7	We created SYS1.PETR17.PARMLIB to contain all the parmlib members that changed for z/OS V1R7 and we used our LOAD*xx* member for migrating our systems one at the time. (See “Using concatenated parmlib” on page 43 for more about our use of concatenated parmlib and see our December 1997 edition for an example of how we use LOAD*xx* to migrate individual systems.)
Applying coexistence service	We applied the necessary coexistence service (also known as compatibility or toleration PTFs) to position our systems for the migration. See the coexistence service requirements in *z/OS and z/OS.e Planning for Installation* and make sure you install the fixes for any APARs that relate to your configuration before you migrate.
IPLing our first z/OS V1R7 image	We brought up z/OS V1R7 on our Z2 test system and ran it there for a couple of weeks.
Updating the RACF templates	To test the RACF dynamic template enhancement, we IPLed the first z/OS V1R7 image without first running the IRRMIN00 utility with PARM=UPDATE. As expected, the following message appeared: `ICH579E RACF TEMPLATES ON DATABASE ARE DOWNLEVEL` RACF initialization still completed successfully. We then ran IRRMIN00 with PARM=UPDATE to dynamically update the templates on all six RACF data sets without the need for an IPL. (See *z/OS Security Server RACF System Programmer's Guide*, SA22-7681 for details about RACF templates.)
IPLing additional z/OS V1R7 images	We continued to bring up additional z/OS V1R7 images across our sysplex, as follows: • We brought up z/OS V1R7 on our on JC0 production system and ran with it for a couple of months. • Next we migrated one test system, Z1, and ran for a couple of weeks. • Next, we migrated an additional production system, J80, and ran with it for a couple of days. • At this point, we took all of the V1R7 images back down to V1R6. This is part of our increased focus on migration testing and fallback. We ran for a full day and experienced no fallback issues. • Next we migrated an additional test system, Z3, and two more productions systems, JF0 and TPN, and ran for a week. • Next we migrated four additional production systems, JA0, JB0, JE0, and JH0, and ran for a couple of weeks. • We then migrated the remaining production system, Z0, to V1R7.

Due to special testing that needed to be done with images on V1R7, the migration for our Sysplex took longer that it would normally take. This time we had 2 images on V1R7 for a couple of months before we migrated the rest of the Sysplex.

More about our migration activities for z/OS V1R7

This section highlights additional details about some of our migration activities.

Running with mixed product levels: During our migration, we successfully ran our sysplex with mixed product levels, including the following:

- z/OS V1R6 and z/OS V1R7
- z/OS V1R6 and z/OS.e V1R7
- z/OS V1R6 JES2 and z/OS V1R7 JES2
- z/OS V1R6 JES3 and z/OS V1R7 JES3.

Using concatenated parmlib: We continue to use concatenated parmlib support to add or update parmlib members for z/OS V1R7. Appendix A, "Some of our parmlib members," on page 331 summarizes the additions and changes we made by parmlib member. Also see our Web site for examples of some of our parmlib members.

This is a good use of concatenated parmlib because it isolates all of the parmlib changes for z/OS V1R7 in one place and makes it easier to migrate multiple systems. Rather than change many parmlib members each time we migrate another system to V1R7, we just add the PARMLIB statements at the appropriate places in SYS0.IPLPARM(LOAD*xx*) to allow that system to use SYS1.PETR17.PARMLIB.

Recompiling REXX EXECs for automation: We recompiled our SA OS/390 REXX EXECs when we migrated to z/OS V1R7. We discuss the need to recompile these REXX EXECs in our our December 1997 edition.

Migrating JES2 large spool datasets: JES2 for z/OS V1R7 supports spool datasets larger than 65K track, if you are in a MAS that has no pre-z/OS V1R7 members.

We implemented this support in our sysplex, using the *z/OS JES2 Initialization and Tuning Guide*, SA22-7532.

After we had completely migrated our entire MAS for z/OS V1R7, and were confident that we would not fall back to a pre-z/OS V1R7 level on any member of the MAS, we enabled JES2 large dataset support with the *$T SPOOLDEF,LARGEDS=ALLOWED* command. Once this command was completed, a COLD START would have been necessary to fall back to a pre-V1R7 JES2.

We then chose a large (32K Cylinder) volume, and allocated a large spool dataset with 491,220 tracks. We used the DSNTYPE=LARGE keyword in our JCL.

```
//SPOOL    EXEC  PGM=IEFBR14
//SYSPRINT DD SYSOUT=*
//*
//SP1200 DD DISP=(,KEEP),SPACE=(TRK,(491220),,CONTIG),
//    DCB=(DSORG=PSU),DSNTYPE=LARGE,
//    DSN=SYS1.HASPACE,UNIT=3390,VOL=SER=SPOLJ5
```

Then we adjusted our TGSPACE=MAX= value to ensure we could add additional TGs (Track Groups), again using the *$T SPOOLDEF* command. In our MAS, we have three tracks per TG, so we needed to ensure we could add 163,740 TGs.

Once the dataset was allocated we formatted and started the spool dataset with the *$S SPL(SPOLJ5),FORMAT* command.

The following example shows what this looks like in SDSF:

Note: The number of TGs for SPOLJ5 shows at 163T, where the T represents Thousands.

```
SDSF SPOOL DISPLAY  J80    23%  ACT 379915 FRE 292216  LINE 1-12 (12)
COMMAND INPUT ===>                                        SCROLL ===> PAGE
NP   VOLUME Status   TGPct TGNum TGUse Command  SAff  Ext LoTrk    HiTrk    Trk
     SPOLJ5   ACTIVE      12 163T  19940          ANY    01 00000001 00077ED4
     SPOLJ8   ACTIVE      39 50025 19822         ANY    07 00000001 00024A3B
     SPOLJM  ACTIVE      28 16615  4747          ANY    0B 00000001 0000C2B5
     SPOLJW ACTIVE      29 16615  4830          ANY    0C 00000001 0000C2B5
     SPOLJ0   ACTIVE      28 16615  4739          ANY    05 00000001 0000C2B5
     SPOLJ1   ACTIVE      29 16615  4869          ANY    08 00000001 0000C2B5
     SPOLJ2   ACTIVE      28 16615  4789          ANY    03 00000001 0000C2B5
     SPOLJ3   ACTIVE      28 16615  4783          ANY    09 00000001 0000C2B5
     SPOLJ4   ACTIVE      28 16615  4777          ANY    04 00000001 0000C2B5
     SPOLJ6   ACTIVE      28 16615  4748          ANY    0D 00000001 0000C2B5
     SPOLJ7   ACTIVE      29 16615  4831          ANY    0A 00000001 0000C2B5
     SPOLJ9   ACTIVE      29 16615  4824          ANY    06 00000001 0000C2B5
```

Migrating to z/OS.e V1R7

This section describes our migration experiences with z/OS.e V1R7.

z/OS.e V1R7 base migration experiences

This section describes our experiences with migrating one system image (JH0) from z/OS.e V1R6 to z/OS.e V1R7. Here we only cover our experiences with our base migration to z/OS.e V1R7, including our high-level migration process and other migration activities and considerations.

Our high-level migration process for z/OS.e V1R7

The following is an overview of our z/OS.e V1R7 migration process.

Before we began: We reviewed the information in *z/OS and z/OS.e Planning for Installation*, GA22-7504, which covers both z/OS V1R7 and z/OS.e V1R7.

Important notice about cloning and software licensing

As discussed in *z/OS and z/OS.e Planning for Installation*, you might find that sharing system libraries or cloning an already-installed z/OS or z/OS.e system is faster and easier than installing z/OS or z/OS.e with an IBM installation package such as ServerPac. Most Parallel Sysplex customers are already aware of the concept of cloning and the benefits it provides.

However, prior to sharing or cloning z/OS or z/OS.e, **you must have a license for each z/OS and z/OS.e operating system that you run.** If you don't have the appropriate license or licenses, you must contact IBM. Any sharing or cloning of z/OS or z/OS.e without the appropriate licenses is not an authorized use of such programs. On a z800 server, if you want to run both z/OS and z/OS.e, z/OS requires the appropriate license for the machine on which it runs and z/OS.e requires a license for the number of engines on which it runs.

For more information about z/OS.e licensing, see z800 Software Pricing Configuration Technical Paper at www.ibm.com/servers/eserver/zseries/library/techpapers/pdf/gm130121.pdf.

Table 10 on page 39 shows the high-level process we followed to migrate our z/OS.e V1R6 system to z/OS.e V1R7.

Table 10. Our high-level migration process for z/OS.e V1R7

Stage	Description
Obtaining licenses for z/OS.e	You need a license for the appropriate number of engines on the z800 or z890 server on which you intend to run z/OS.e (and, you would also need a license to run z/OS on the z800 or z890, if you intend to install it there). We use an internal process to do this; however, you must use the official process stated in *z800 Software Pricing Configuration Technical Paper*.
Updating the z800 or z890 LPAR name	z/OS.e must run in LPAR mode and the LPAR name must be of the form `ZOSE`*xxxx*, where *xxxx* is up to 4 user-specified alphanumeric characters. The name of the LPAR in which we run z/OS.e is `ZOSEJH0`. (We used HCD to set this when we first installed z/OS.e V1R3.)
Updating parmlib for z/OS.e V1R7	z/OS.e requires the `LICENSE=Z/OSE` statement in the IEASYS*xx* parmlib member. We used the same SYS1.PETR17.PARMLIB data set that we created for z/OS V1R7. We then have separate IEASYS*xx* and IFAPRD*xx* members in SYS1.PARMLIB that we tailored specifically for z/OS.e. See "Updating system data sets for z/OS.e" on page 40 for details.
Updating our LOAD*xx* member	During our initial installation of z/OS.e V1R3, we updated the entry for our system JH0 in our LOAD*xx* member in SYS0.IPLPARM to point to our new IEASYS02 parmlib member and to reflect the new LPAR name. Therefore, we did not need to change it for V1R7.
Updating our IEASYMPT member	During our initial installation of z/OS.e V1R3, we updated the entry for our system JH0 in our IEASYMPT member in SYS1.PETR13.PARMLIB to point to our new IFAPRD*xx* parmlib member and to reflect the new LPAR name. Therefore, when we created our new SYS1.PETR17.PARMLIB, we carried the change along for V1R7.
IPLing the z/OS.e V1R7 image	We brought up z/OS.e V1R7 on our JH0 production system.

More about our migration activities for z/OS.e V1R7

This section highlights additional details about some of our migration activities.

About our z890 LPAR environment: z/OS.e must run in LPAR mode on a zSeries 800 or 890 mainframe server; it cannot run in basic mode. In addition, the name of the LPAR in which z/OS.e runs must be of the form `ZOSE`*xxxx*, where *xxxx* is up to four user-specified alphanumeric characters. The name of our z/OS.e z890 LPAR is ZOSEJH0.

Note: You can only run z/OS.e in a partition named ZOSE*xxxx*. You cannot IPL a z/OS system in a partition named ZOSE*xxxx*.

We currently run z/OS.e (JH0) as our only LPAR in a z890 server.

Note: Don't let the fact that z/OS.e only runs on a z800 or z890 server confuse you. These are fully functional zSeries servers and, in addition to z/OS.e, theyt supports all of the same zSeries operating systems as a z900 or z990 server.

Updating system data sets for z/OS.e: We continue to use concatenated parmlib support to add or update parmlib members for z/OS.e V1R7. We use the same SYS1.PETR17.PARMLIB data set as we do for our z/OS V1R7 systems.

Below are examples of our parmlib customizations to accommodate z/OS.e V1R7. Appendix A, "Some of our parmlib members," on page 331 summarizes the changes we made by parmlib member.

Example: We have a separate IEASYS*xx* member, IEASYS02, which specifies the `LICENSE=Z/OSE` statement that z/OS.e requires.

The entry for our z/OS.e system (JH0) in our LOAD*xx* member in SYS0.IPLPARM points to our IEASYS02 parmlib member and specifies the name of our z/OS.e LPAR, as follows:

```
⋮
HWNAME   z800name
LPARNAME  ZOSEJH0
PARMLIB  SYS1.PETR17.PARMLIB
SYSPARM   02
⋮
```

Example: We have a separate IFAPRD*xx* member, IFAPRD02, which specifies the product ID value `5655-G52` for z/OS.e. There is no change to the product name value for z/OS.e (the product name value remains `Z/OS`).

Below is an example of one of the entries from our IFAPRD02 member:

```
⋮
PRODUCT OWNER('IBM CORP')
        NAME(Z/OS)
        ID( 5655-G52 )
        VERSION(*) RELEASE(*) MOD(*)
        FEATURENAME(Z/OS)
        STATE(ENABLED)
⋮
```

We also have an entry for our system JH0 in our IEASYMPT member in SYS1.PETR17.PARMLIB to point to our new IFAPRD02 parmlib member and to reflect the z/OS.e LPAR name, as follows:

```
⋮
SYSDEF HWNAME(z800name)
       LPARNAME( ZOSEJH0 )
       SYSNAME(JH0)
       SYSCLONE(JH)
⋮
       SYMDEF(&PROD= '02' )
⋮
```

Using current z/OS.e levels of JES2 and LE: As required, we are using the level of JES2 and Language Environment (LE) that comes with z/OS.e V1R7. z/OS.e does not permit the use of a lower level JES2 (or JES3) or LE.

Updating the ARM policy: You must ensure that your automation policies, such as ARM, do not try to use a z/OS.e image to start products that z/OS.e does not

support. For example, do not identify a z/OS.e image as a restart target in a Parallel Sysplex that contains a mix of z/OS.e and z/OS images where the z/OS images run IMS, CICS, or DB2 with a requirement for CICS. CICS, IMS, or DB2 that uses CICS cannot restart on a z/OS.e image, but must restart on a z/OS image. If, for example, a CICS region attempts to start on z/OS.e, the region will start but the applications will fail with a U4093 abend.

Back when we installed z/OS.e V1R3, we removed our z/OS.e image, JH0, as a restart target for the unsupported subsystems mentioned above.

Removing z/OS.e from participation in MNPS: In our environment, CICS is the only exploiter of multiple node persistent sessions (MNPS) support. Because CICS cannot run on z/OS.e, there is no reason for the VTAM on z/OS.e to connect to the MNPS structure, ISTMNPS. We removed our z/OS.e image from participating in MNPS by coding the `STRMNPS=NONE` statement in our VTAM start member, ATCSTR*xx*, in SYS1.VTAMLST.

Removing z/OS.e from participation in TSO generic resource groups: Since TSO on z/OS.e only allows a maximum of eight concurrent sessions, we removed our z/OS.e image from participating in TSO generic resource groups. You can do this by coding the `GNAME=NONE` parameter—either in a separate TSOKEY*xx* member in parmlib or on the START command that starts the terminal control address space (TCAS).

In our case, we use a single TSOKEY*xx* member that has a symbolic value for the GNAME parameter. We then set that symbol to `NONE` for our JH0 image in our IEASYMPT member.

Other experiences with z/OS.e V1R7

Our testing of z/OS.e V1R7 included the following workloads or scenarios:

- z/OS UNIX System Services
- DB2 UDB
- IBM HTTP Server in scalable server mode
- WebSphere Application Server for z/OS
- CICS Transaction Gateway (CTG) to access CICS regions running in z/OS images on the same CPC and other CPCs
- DB2 access from Linux guests under z/VM on the same CPC
- our Bookstore application transactions

Migrating to z/OS V1R6

This section describes our migration experiences with z/OS V1R6.

z/OS V1R6 base migration experiences

In this section we only describe our experiences with our base migration to z/OS V1R6, without having implemented any new functions. It includes our high-level migration process along with other migration activities and considerations.

Our high-level migration process for z/OS V1R6

The following is an overview of our z/OS V1R6 migration process.

Before we began: We reviewed the migration information in *z/OS and z/OS.e Planning for Installation*, GA22-7504 and *z/OS Migration*.

Table 11 shows the high-level process we followed to migrate the members of our sysplex from z/OS V1R5 to z/OS V1R6.

Table 11. Our high-level migration process for z/OS V1R6

Stage	Description
Updating parmlib for z/OS V1R6	We created SYS1.PETR16.PARMLIB to contain all the parmlib members that changed for z/OS V1R6 and we used our LOAD*xx* member for migrating our systems one at a time. (See “Using concatenated parmlib” on page 43 for more about our use of concatenated parmlib and see our December 1997 edition for an example of how we use LOAD*xx* to migrate individual systems.)
Applying coexistence service	We applied the necessary coexistence service (also known as compatibility or toleration PTFs) to position our systems for the migration. See the coexistence service requirements in *z/OS and z/OS.e Planning for Installation* and make sure you install the fixes for any APARs that relate to your configuration before you migrate.
IPLing our first z/OS V1R6 image	We brought up z/OS V1R6 on our Z1 test system and ran it there for about one week.
Updating the RACF templates	To test the RACF dynamic template enhancement, we IPLed the first z/OS V1R6 image without first running the IRRMIN00 utility with PARM=UPDATE. As expected, the following message appeared: `ICH579E RACF TEMPLATES ON DATABASE ARE DOWNLEVEL` RACF initialization still completed successfully. We then ran IRRMIN00 with PARM=UPDATE to dynamically update the templates on all six RACF data sets without the need for an IPL. (See *z/OS Security Server RACF System Programmer's Guide*, SA22-7681 for details about RACF templates.)
IPLing additional z/OS V1R6 images	We continued to bring up additional z/OS V1R6 images across our sysplex, as follows: • We brought up z/OS V1R6 on our Z2 and Z3 test systems and ran for a couple of days. • Next, we migrated one production system, JF0, and ran for about a week. • Next, we migrated an additional test system, Z1, and two production systems, JG0 and JH0, and ran for another week. • At this point, we took all of the V1R6 images back down to V1R5. This is part of our increased focus on migration testing and fallback. We ran for a full day and experienced no fallback issues. • Next, we migrated an additional test system, Z0, and three production systems, TPN, JB0, and JC0, and ran for a couple of days. • Next, we migrated two more production systems, JA0 and JE0, and ran for about a week. • We then migrated the remaining two systems, J80 and J90.

The total time for our migration was approximately a month.

More about our migration activities for z/OS V1R6

This section highlights additional details about some of our migration activities.

Running with mixed product levels: During our migration, we successfully ran our sysplex with mixed product levels, including the following:

- z/OS V1R5 and z/OS V1R6
- z/OS V1R5 and z/OS.e V1R6
- z/OS V1R5 JES2 and z/OS V1R6 JES2
- z/OS V1R5 JES3 and z/OS V1R6 JES3

Using concatenated parmlib: We continue to use concatenated parmlib support to add or update parmlib members for z/OS V1R6. Appendix A, "Some of our parmlib members," on page 331 summarizes the additions and changes we made by parmlib member. Also see our Web site for examples of some of our parmlib members.

This is a good use of concatenated parmlib because it isolates all of the parmlib changes for z/OS V1R6 in one place and makes it easier to migrate multiple systems. Rather than change many parmlib members each time we migrate another system to V1R6, we just add the PARMLIB statements at the appropriate places in SYS0.IPLPARM(LOAD*xx*) to allow that system to use SYS1.PETR16.PARMLIB.

Recompiling REXX EXECs for automation: We recompiled our SA OS/390 REXX EXECs when we migrated to z/OS V1R6. We discuss the need to recompile these REXX EXECs in our our December 1997 edition.

Migrating to z/OS.e V1R6

This section describes our migration experiences with z/OS.e V1R6.

z/OS.e V1R6 base migration experiences

This section describes our experiences with migrating one system image (JH0) from z/OS.e V1R5 to z/OS.e V1R6. Here we only cover our experiences with our base migration to z/OS.e V1R6, including our high-level migration process and other migration activities and considerations.

Our high-level migration process for z/OS.e V1R6

The following is an overview of our z/OS.e V1R6 migration process.

Before we began: We reviewed the information in *z/OS and z/OS.e Planning for Installation*, GA22-7504, which covers both z/OS V1R6 and z/OS.e V1R6.

Important notice about cloning and software licensing

As discussed in *z/OS and z/OS.e Planning for Installation*, you might find that sharing system libraries or cloning an already-installed z/OS or z/OS.e system is faster and easier than installing z/OS or z/OS.e with an IBM installation package such as ServerPac. Most Parallel Sysplex customers are already aware of the concept of cloning and the benefits it provides.

However, prior to sharing or cloning z/OS or z/OS.e, **you must have a license for each z/OS and z/OS.e operating system that you run.** If you don't have the appropriate license or licenses, you must contact IBM. Any sharing or cloning of z/OS or z/OS.e without the appropriate licenses is not an authorized use of such programs. On a z800 server, if you want to run both z/OS and z/OS.e, z/OS requires the appropriate license for the machine on which it runs and z/OS.e requires a license for the number of engines on which it runs.

For more information about z/OS.e licensing, see *z800 Software Pricing Configuration Technical Paper* at www.ibm.com/servers/eserver/zseries/library/techpapers/pdf/gm130121.pdf.

Table 12 shows the high-level process we followed to migrate our z/OS.e V1R5 system to z/OS.e V1R6.

Table 12. Our high-level migration process for z/OS.e V1R6

Stage	Description
Obtaining licenses for z/OS.e	You need a license for the appropriate number of engines on the z800 or z890 server on which you intend to run z/OS.e (and, you would also need a license to run z/OS on the z800 or z890, if you intend to install it there). We use an internal process to do this; however, you must use the official process stated in *z800 Software Pricing Configuration Technical Paper*.
Updating the z800 or z890 LPAR name	z/OS.e must run in LPAR mode and the LPAR name must be of the form `ZOSE`*xxxx*, where *xxxx* is up to 4 user-specified alphanumeric characters. The name of the LPAR in which we run z/OS.e is `ZOSEJH0`. (We used HCD to set this when we first installed z/OS.e V1R3.)
Updating parmlib for z/OS.e V1R6	z/OS.e requires the `LICENSE=Z/OSE` statement in the IEASYS*xx* parmlib member. We used the same SYS1.PETR16.PARMLIB data set that we created for z/OS V1R6. We then have separate IEASYS*xx* and IFAPRD*xx* members in SYS1.PARMLIB that we tailored specifically for z/OS.e. See "Updating system data sets for z/OS.e" on page 45 for details.
Updating our LOAD*xx* member	During our initial installation of z/OS.e V1R3, we updated the entry for our system JH0 in our LOAD*xx* member in SYS0.IPLPARM to point to our new IEASYS02 parmlib member and to reflect the new LPAR name. Therefore, we did not need to change it for V1R6.

Table 12. Our high-level migration process for z/OS.e V1R6 (continued)

Stage	Description
Updating our IEASYMPT member	During our initial installation of z/OS.e V1R3, we updated the entry for our system JH0 in our IEASYMPT member in SYS1.PETR13.PARMLIB to point to our new IFAPRD*xx* parmlib member and to reflect the new LPAR name. Therefore, when we created our new SYS1.PETR16.PARMLIB, we carried the change along for V1R6.
IPLing the z/OS.e V1R6 image	We brought up z/OS.e V1R6 on our JH0 production system.

More about our migration activities for z/OS.e V1R6

This section highlights additional details about some of our migration activities.

About our z890 LPAR environment: z/OS.e must run in LPAR mode on a zSeries 800 or 890 mainframe server; it cannot run in basic mode. In addition, the name of the LPAR in which z/OS.e runs must be of the form `ZOSE`*xxxx*, where *xxxx* is up to four user-specified alphanumeric characters. The name of our z/OS.e z890 LPAR is ZOSEJH0.

Note: You can only run z/OS.e in a partition named ZOSE*xxxx*. You cannot IPL a z/OS system in a partition named ZOSE*xxxx*.

We currently run z/OS.e (JH0) in a mixed LPAR environment alongside LPARs running z/OS (JG0) on the same z890 server.

Note: Don't let the fact that z/OS.e only runs on a z800 or z890 server confuse you. These are fully functional zSeries servers and, in addition to z/OS.e, theyt supports all of the same zSeries operating systems as a z900 or z990 server.

Updating system data sets for z/OS.e: We continue to use concatenated parmlib support to add or update parmlib members for z/OS.e V1R6. We use the same SYS1.PETR16.PARMLIB data set as we do for our z/OS V1R6 systems.

Below are examples of our parmlib customizations to accommodate z/OS.e V1R6. Appendix A, "Some of our parmlib members," on page 331 summarizes the changes we made by parmlib member.

Example: We have a separate IEASYS*xx* member, IEASYS02, which specifies the `LICENSE=Z/OSE` statement that z/OS.e requires.

The entry for our z/OS.e system (JH0) in our LOAD*xx* member in SYS0.IPLPARM points to our IEASYS02 parmlib member and specifies the name of our z/OS.e LPAR, as follows:

```
⋮
HWNAME   z800name
LPARNAME ZOSEJH0
PARMLIB  SYS1.PETR16.PARMLIB
```

```
SYSPARM   02
:
```

Example: We have a separate IFAPRD*xx* member, IFAPRD02, which specifies the product ID value 5655-G52 for z/OS.e. There is no change to the product name value for z/OS.e (the product name value remains Z/OS).

Below is an example of one of the entries from our IFAPRD02 member:

```
:
PRODUCT OWNER('IBM CORP')
        NAME(Z/OS)
        ID( 5655-G52 )
        VERSION(*) RELEASE(*) MOD(*)
        FEATURENAME(Z/OS)
        STATE(ENABLED)
:
```

We also have an entry for our system JH0 in our IEASYMPT member in SYS1.PETR16.PARMLIB to point to our new IFAPRD02 parmlib member and to reflect the z/OS.e LPAR name, as follows:

```
:
SYSDEF HWNAME(z800name)
       LPARNAME( ZOSEJH0 )
       SYSNAME(JH0)
       SYSCLONE(JH)
:
       SYMDEF(&PROD= '02' )
:
```

Using current z/OS.e levels of JES2 and LE: As required, we are using the level of JES2 and Language Environment (LE) that comes with z/OS.e V1R6. z/OS.e does not permit the use of a lower level JES2 (or JES3) or LE.

Updating the ARM policy: You must ensure that your automation policies, such as ARM, do not try to use a z/OS.e image to start products that z/OS.e does not support. For example, do not identify a z/OS.e image as a restart target in a Parallel Sysplex that contains a mix of z/OS.e and z/OS images where the z/OS images run IMS, CICS, or DB2 with a requirement for CICS. CICS, IMS, or DB2 that uses CICS cannot restart on a z/OS.e image, but must restart on a z/OS image. If, for example, a CICS region attempts to start on z/OS.e, the region will start but the applications will fail with a U4093 abend.

Back when we installed z/OS.e V1R3, we removed our z/OS.e image, JH0, as a restart target for the unsupported subsystems mentioned above.

Removing z/OS.e from participation in MNPS: In our environment, CICS is the only exploiter of multiple node persistent sessions (MNPS) support. Because CICS cannot run on z/OS.e, there is no reason for the VTAM on z/OS.e to connect to the MNPS structure, ISTMNPS. We removed our z/OS.e image from participating in MNPS by coding the STRMNPS=NONE statement in our VTAM start member, ATCSTR*xx*, in SYS1.VTAMLST.

Removing z/OS.e from participation in TSO generic resource groups: Since TSO on z/OS.e only allows a maximum of eight concurrent sessions, we removed our z/OS.e image from participating in TSO generic resource groups. You can do this by coding the GNAME=NONE parameter—either in a separate TSOKEY*xx* member in parmlib or on the START command that starts the terminal control address space (TCAS).

In our case, we use a single TSOKEY*xx* member that has a symbolic value for the GNAME parameter. We then set that symbol to `NONE` for our JH0 image in our IEASYMPT member.

Other experiences with z/OS.e V1R6

Our testing of z/OS.e V1R6 included the following workloads or scenarios:

- z/OS UNIX System Services
- DB2 UDB
- IBM HTTP Server in scalable server mode
- WebSphere Application Server for z/OS
- CICS Transaction Gateway (CTG) to access CICS regions running in z/OS images on the same CPC and other CPCs
- DB2 access from Linux guests under z/VM on the same CPC
- our Bookstore application transactions

Migrating z/OS Images and a Coupling Facility to the z9

We migrated the following images from two other CPCs to the z9 server:

- Our J80 and Z3 z/OS images that were running on our z990 server
- Our JF0 and Z1 z/OS images that were running on our z900 server
- Our CF2 coupling facility that was running on our z990 server

We added the following zVM and Linux images to the z9 server:

- zVM images for Linux Distr01,Petlvs and Petlvs2
- Linux images Distr02 and Ticltst

Figure 4 summarizes the LPs that we migrated to the z9 server.

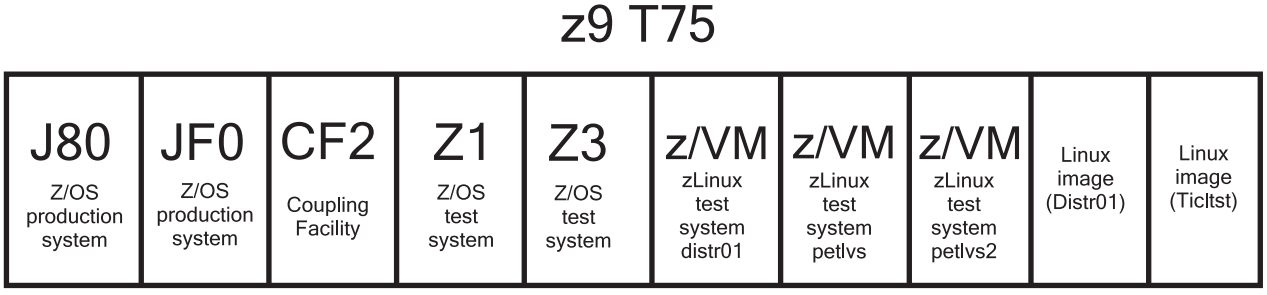


Figure 4. Summary of LPs that we migrated to the z9 server

Some of the features that differentiate our new z9 from our z990 are:

MIDAWS: z9 introduces a new type of channel program called a Modified Indirect Data Addressing Word (MIDAWS) for both Escon and Ficon. MIDAWS gives extended format VSAM data sets a considerable performance boost. The environment to most benefit from this support is one in which there is a great amount of extended format DFSMS data set activity. This includes DB2 database manager, CICS with extended format VSAM, and any environment that extensively uses extended format sequential data.

Managing ICF, IFL, and zAAPs independently: PUs defined as Internal Coupling Facility (ICF) processors, Integrated Facility for Linux (IFL) processors, or System

z9 Application Assist Processors (zAAPs) are now managed separately. In the past, ICF processors, IFL processors, and zAAPs were grouped together for allocation within and across the LPARs. The separate management of PU types enhances and simplifies capacity planning and management of the configured LPARs and their associated processor resources.

Improved LPAR weight management of CPs and zAAPs: For LPARs that have both CPs and zAAPs configured, a new zAAP weight specification is provided to allow a new unique LPAR weight specification for shared zAAPs to be defined. The existing LPAR shared processor weight specification is now applied only to the CPs configured to the LPAR. In the past, the existing shared processor weight specification was applied to both the shared CPs and to shared zAAPs configured to the LPAR. The ability to specify a separate LPAR weight for shared zAAPs helps to enhance and simplify capacity planning and management of the configured LPARs and their associated processor resources.

Multiple Subchannel Sets: Multiple Subchannel Set support enables constraint relief for subchannels. Two subchannel sets per LCSS will be implemented enabling a total of 63K subchannels in set-0 (was available with z990) and adding 64K-1 subchannels in set-1 with this function for z9.Subchannels for parallel devices will not be allowed in subchannel set-1, and initially only z/OS will support multiple subchannel sets with only Shark PAV devices in the second set.

z/OS performance

The performance of our z/OS systems is an important issue for us, just as it is for you. If we are to be customer-like, we must pay attention to meeting the goals in our service level agreements.

The following describes what we do in each phase of our testing, and what we plan to periodically report to you in our test reports:

- Monitor our performance in terms of our service level agreements

  Our goal for our sysplex workloads continues to be 90% CP utilization across the systems in the sysplex, with WLM goals such as 80% of CICS transactions completed in less than 0.6 seconds on those images where CICS runs. We fill in the remaining 10% with batch work and various additional types of users, such as z/OS UNIX users (such as WebSphere for z/OS), TSO users, and workstation clients.

  Note: This is not formal performance testing for purposes of publishing performance statistics for z/OS. It is a way for us to establish and report on reasonable goals for response times and transaction rates for the various types of workloads we run, just as a customer would do to create a service level agreement (SLA).
- Identify performance problems in our environment, find solutions to those problems, and report the information to you.
- Provide you with periodic performance snapshots of our environment, in the form of RMF reports, to provide pertinent information such as how many transactions we process per second and what our response times are for various workloads. You can find those reports in Appendix B, “Some of our RMF reports,” on page 333.

Chapter 4. Using zSeries Application Assist Processors (zAAPs)

As promised, we've updated our most recent testing of zAAP since the June 2004 report.

IBM @server zSeries 890 (z890) and zSeries 990 (z990) servers make available a new, optional feature—the zSeries Application Assist Processor (zAAP)—which provides a strategic z/OS Java execution environment for customers who desire the powerful integration advantages and traditional qualities of service of the zSeries platform.

A zAAP is similar in concept to a System Assist Processor (SAP). Unlike CPs, ICFs, and IFLs, zAAPs can do nothing on their own; they cannot perform an IPL and cannot run an operating system. zAAPs must operate along with general purpose CPs within logical partitions running z/OS; however, they are designed to operate asynchronously with the general purpose CPs to execute Java programming under control of the IBM Java Virtual Machine (JVM).

This chapter describes what we did to configure and to prepare to exercise the zAAP feature on our z990 server.

Prerequisites for zAPP

The following are prerequisites for zAAP and execution of the JVM processing cycles:

- the IBM Software Developer's Kit (SDK) for z/OS
- z/OS 1.6 (or z/OS.e 1.6)
- Java 2 Technology Edition V1.4.1 with PTF for APAR PQ86689
- and the Processor Resource/Systems Manager™ ((PR/SM™) must be enabled).

Subsystems and applications using SDK 1.4 that exploit zAAPs

The following subsystems and applications using SDK 1.4 exploit zAAPs:

- WebSphere Application Server 5.1
- CICS/TS 2.3
- DB2 V7, DB2 V8 (we tested with both)
- IMS V7, IMS V8, and IMS V9 (we tested with IMS V8)
- Websphere MQ 5.3.1
- Websphere Business Integration Message Broker 5.0

Setting up zAAP

First we executed the recommended zAAP Projection Tool for Java 2 Technology Edition SDK 1.3.1 against the available Java workload we had at the early stage of our testing. This provided us a baseline or a starting point for defining the minimum number of zAAP processors we would require on our z990 and z890 processors. Please reference the following for more details pertaining to the Projection Tool:

- "Installation of the zAAP Projection Tool Instrumented SDK in WebSphere for z/OS Version 5" available at:

  `http://www-1.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100431`

- z/OS Performance: Capacity Planning Considerations for zAAP Processors available at:

  `http://www-1.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100417`

As mentioned in the intro we recommend that you contact your hardware support for the latest hardware and software requirements. We licensed 1 CP on our z890 and 2 CPs on our z990 as zAAPs.

Configuring zAAPs

We configured two zAAPs on all our z/OS images on our z990 server and we configured one zAAP on our z890 server. When you configure the z/OS logical partitions, you simply specify how many logical zAAPs you want to configure for each partition, just as you do the number of standard CPs. When you IPL the system, z/OS determines how many zAAPs are configured and manages an additional dispatcher queue for zAAP-eligible work.

We did the following to configure the zAAPs:

1. Updated the image profiles for the Z2 and Z3 partitions to define two zAAPs to each partition.

   Example: Figure 5 shows an example of the image profile for our Z2 image with two zAAPs defined.

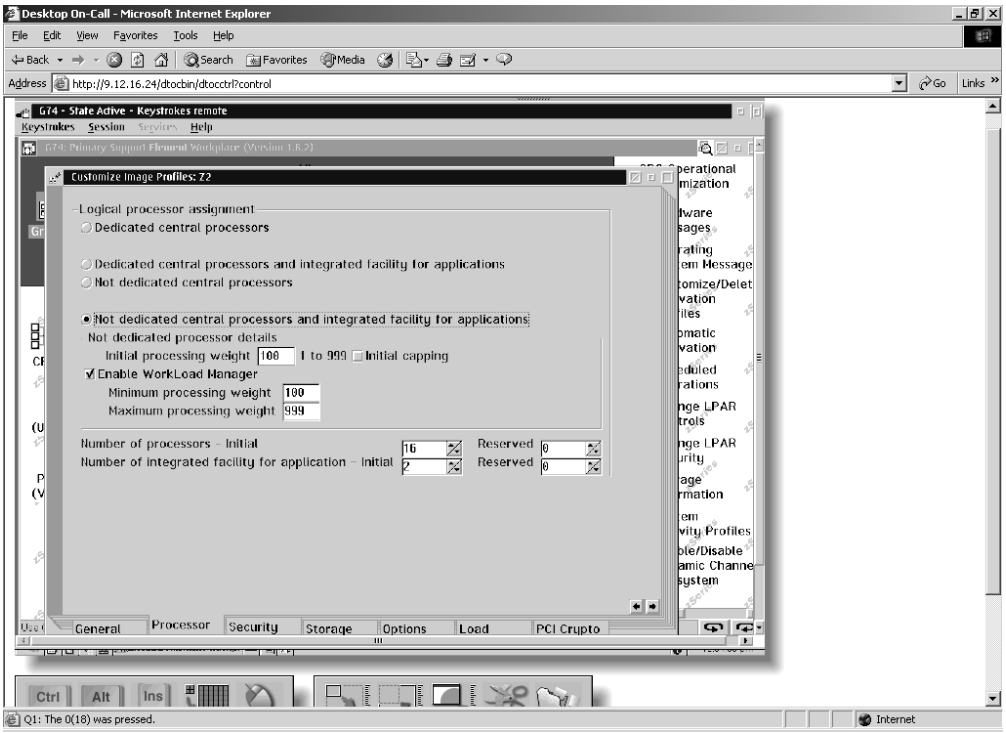


Figure 5. Example of the image profile for our Z2 image with two zAAPs defined.

2. Updated parmlib member IEAOPT*xx* for the z/OS partitions to specify the following options:

IFACROSSOVER=YES
zAAP-eligible work may execute on zAAPs or it can “cross over” and execute on standard processors.

IFAHONORPRIORITY=YES
Standard processors execute both Java and non-Java work in order of dispatching priority.

---

3. Deactivated and reactivated the z/OS partitions to bring the zAAPs online.

   You can use the D M=CPU command to display the status of the zAAPs. The zAAPs appear as assist processors in the response to the D M=CPU command.

   Example: The following is an example of the response to the D M=CPU command on system Z2:

```
IEE174I 15.34.28 DISPLAY M
PROCESSOR STATUS
ID  CPU                 SERIAL
00  +                   02B52A2084
01  +                   02B52A2084
02  +                   02B52A2084
03  +                   02B52A2084
04  +A                  02B52A2084
05  +A                  02B52A2084

CPC ND = 002084.D32.IBM.00.00000001B52A
CPC SI = 2084.325.IBM.00.000000000001B52A
CPC ID = 00
CPC NAME = G74
LP NAME = Z2         LP ID =  2
CSS ID  = 0
MIF ID  = 2
```

   Example: The following is an example of the response to the D M=CPU command on system Z3:

```
IEE174I 15.38.30 DISPLAY M
PROCESSOR STATUS
ID  CPU                 SERIAL
00  +                   24B52A2084
01  +                   24B52A2084
02  +                   24B52A2084
03  +                   24B52A2084
04  +A                  24B52A2084
05  +A                  24B52A2084

CPC ND = 002084.D32.IBM.00.00000001B52A
CPC SI = 2084.325.IBM.00.000000000001B52A
CPC ID = 00
CPC NAME = G74
LP NAME = Z3         LP ID = 24
CSS ID  = 2
MIF ID  = 4
```

Monitoring zAAP utilization

There is support in RMF (supplied by APAR OA05731) to provide information about zAAP utilization. This information is useful to determine if and when you need to add additional zAAP capacity.

SMF is another source of information. SMF type 72 records contain information about zAAP utilization. There are also new fields in SMF type 30 records to indicate

the amount of time spent on zAAP work as well as the amount of time spent executing zAAP-eligible work on standard processors for both crossover and honor priority execution modes.

Here is an example of our RMF Monitor III displaying the use of the zAAPs on our z990 processor highlighted in **bold**:

```
                        RMF V1R5   CPC Capacity                  Line 1 o
Command ===>                                                  Scroll ===>
Switched to option set WLMPOL01 on JA0.
Samples: 120     System: JA0   Date: 09/14/04  Time: 11.09.00  Range: 120

Partition:   JA0        2084 Model 325
CPC Capacity:    1114   Weight % of Max: 10.0       4h MSU Average:   175
Image Capacity:  1069   WLM Capping %:   ****       4h MSU Maximum:   205

Partition  --- MSU ---  Cap  Proc    Logical Util %   - Physical Util % -
             Def   Act  Def   Num    Effect   Total   LPAR  Effect  Total

*CP                                                    2.1    46.9   49.0
EBTELNX        0     1  NO    2.0       1.3     1.3   0.0     0.1    0.1
JA0            0   191  NO    7.0      60.3    61.2   0.3    16.9   17.1
JC0            0   158  NO    7.0      49.8    50.6   0.2    13.9   14.2
JE0            0   137  NO    8.0      37.9    38.6   0.2    12.1   12.3
Z2             0    27  NO    7.0       8.5     8.7   0.1     2.4    2.4
Z3             0    17  NO    4.0       9.3     9.5   0.0     1.5    1.5
PHYSICAL                                               1.3            1.3

*ICF                                                   1.2    60.6   61.8
CF2                         3.0      98.9    98.9    0.0    42.4   42.4
JA0                    NO   2.0      46.7    46.9    0.1    13.3   13.4
JC0                    NO   2.0       0.0     0.0    0.0     0.0    0.0
JE0                    NO   2.0       0.0     0.0    0.0     0.0    0.0
PETVM                  NO   2.0      15.0    15.6    0.2     4.3    4.4
Z2                     NO   2.0       2.0     2.1    0.0     0.6    0.6
Z3                     NO   2.0       0.1     0.1    0.0     0.0    0.0
PHYSICAL                                               1.0            1.0
```

Preparing our workloads to exercise the zAAP feature

Initially, we selected several of our current MQ Web workloads and rewrote them as base Java applications (non-Web) to exercise the zAAP feature. We also installed WebSphere Business Integration Message Broker 5.0 for zAAP testing. This product itself uses Java and will increase the utilization of the zAAP feature in addition to the workloads. .

In the near future, we plan to use the following MQ-based workloads (which run via TPNS scripts and TSO users) to test the zAAP feature:

- MQLARGE — This workload currently runs primarily on system JG0. Its purpose is to create temporary MQ queues and put large messages on them. We added a compute module to increase the application's CPU usage, as MQ itself does not consume much CPU resource.
- MQCICS — This workload runs on system JB0 and puts a request message on a CICS bridge queue to run a DB2 transaction.
- MQDQM — This is a communications test workload that puts and gets messages between a local MQ queue manager and a remote system's queue manager.
- MQDQLSSL — This workload is similar to the MQDQM workload but uses SSL channels to test security.

- RetailTPNS — This workload simulates a retail type of application and uses WebSphere MQ Integrator to put messages on a queue for the broker to process. We run a TPNS version in Java that will use the zAAP feature.

Other workloads we support are:

- DB2— In the past, Integration Test had implemented the NST (Native Stress Test) Version 6 workload for DB2, which is a TPNS driven, CICS based workload comprised of COBOL programs, stored procedures and user defined functions (UDFs). For z/OS 1.6 with the introduction of eServer™ zSeries Application Assist Processor, several of the stored procedures originally written in COBOL were converted to Java, and a new Workload Manager (WLM) address space to run the Java stored procedures was defined.
- IMS— The IMS Java workload is based on the IMS Java Dealership IVP application, which is documented in the ″IMS Java User's Guide″ (SC27-1296-00). The environment is IMS V8 running SDK 1.4.2. This application consists of one database and one transaction (with multiple methods). We modified the IMS V8 Dealership IVP application by:
  - Creating MFS screen formats so the transactions can be set up as an OLTP workload
  - Coded TPNS scripts to drive different dealership application methods.

We are currently running the following methods: FindACar, ListModels and ShowModelDetails.

We executed our zAAP testing with different configurations.The majority of our testing was completed with our zAAPs configured online. We also performed extensive testing with the zAAP processors configuring them online and offline with Java workload running.

Chapter 5. Migrating to CICS TS Version 3 Release 1

In this section, we describe our experiences with migrating to CICS Transaction Server Version 3 Release 1 (CICS TS 3.1). CICS TS 3.1 contains the following:

- HBDD110 CICS Application Migration Aid
- HCI6400 CICS - Base
- HCP3100 CICSPLEX System Manager - Base
- H0B5110 CICS REXX Runtime
- H0B7110 CICS REXX Runtime Development
- H0Z2110 CICS REXX Runtime Common
- JCI640D CICS - JAVA/IIOP
- JCI6401 CICS - COBOL language parts
- JCI6402 CICS - PL/1 language parts
- JCI6403 CICS - 'C' language parts
- JCP3102 CICS/Plex System Manager - SAS components

Applicable documentation: During the migration to CICS TS 3.1, we used the following publications:

- *Program Directory for CICS Transaction Server for z/OS, GI10-2586* located at:
  `http://publibz.boulder.ibm.com/epubs/pdf/i1025860.pdf`
- *CICS Transaction Server for z/OS Migration Guide, GC34-6425* located at:
  `http://publibfp.boulder.ibm.com/epubs/pdf/dfhe5b02.pdf`
- *CICS Transaction Server for z/OS Installation Guide, GC34-6426* located at:
  `http://publibfp.boulder.ibm.com/epubs/pdf/dfhe5b02.pdf`
- *CICS Transaction Server for z/OS Messages and Codes, GC34-6442* located at:
  `http://publibfp.boulder.ibm.com/epubs/pdf/dfhe5b02.pdf`
- *CICS Transaction Server for z/OS Operations and Utilities Guide, SC34-6431* located at:
  `http://publibfp.boulder.ibm.com/epubs/pdf/dfhe5b02.pdf`
- *CICS Transaction Server for z/OS CICSPlexSM Messages and Codes, GC34-6471* located at:
  `http://publibz.boulder.ibm.com/epubs/pdf/eyua1b01.pdf`

Overview of migrating to CICS TS 3.1

As always, our goal with this migration was to follow the path of a typical customer. We migrated slowly across the sysplex, and within the workloads on the sysplex. This created a thorough mixture of releases on a system as well as across the CICSPlex. We did this to uncover as many compatibility and operational problems as possible. After we tested the migration steps on our test CICSPlex, we were ready to migrate our production CICSPlex.

As we mentioned in our last report, we've added another workload to our CICSPlex. In addition to the original data-sharing workloads (IMS, VSAM-RLS, DB2), we now have a fourth workload, used primarily to test new z/OS Cryptographic hardware, ICSF, and Java. We call this our application group-C workload, which exploits the CICS-ICSF attach facility and the CICS-Java interface.

Figure 6 shows our CICS TS 3.1 and CPSM 3.1 latest configuration:

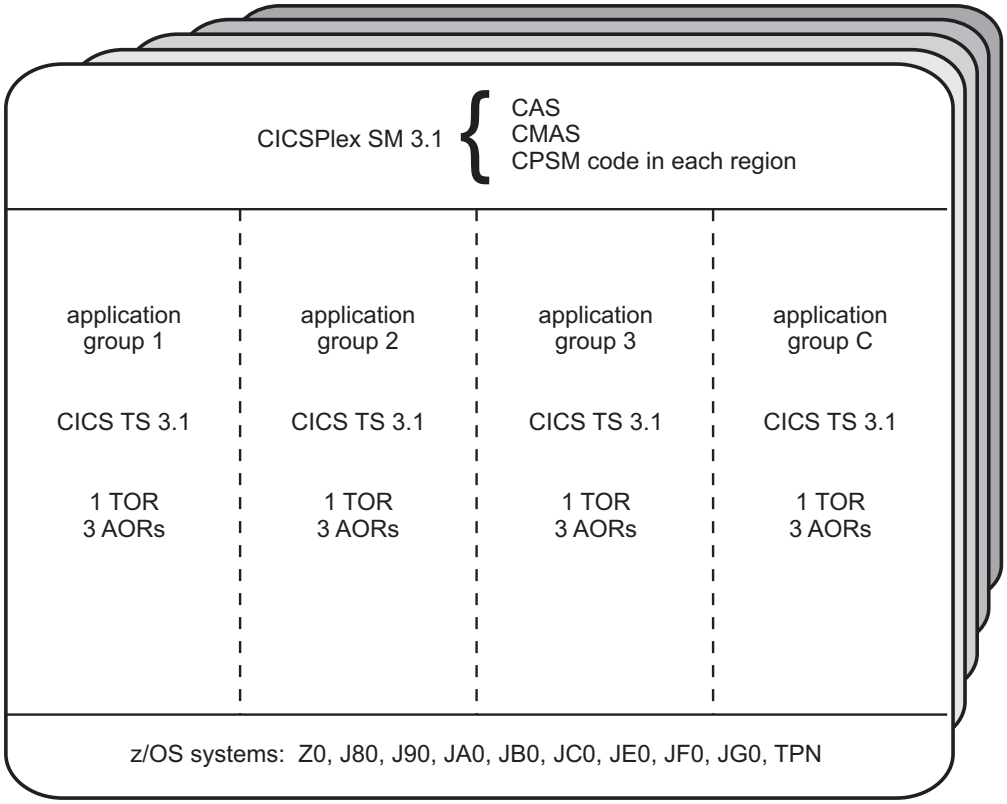


Figure 6. Our CICS TS 3.1 and CPSM 3.1 configuration

When all of our systems are up and running we have a total of 10 CASs/CMASs monitoring 120+ CICS regions (MASs). We stopped the migration when we were about half-way across the sysplex. At this point we had CTS23 CMASs communicating with CTS31 CMASs. Under the CTS31 CMASs, we had a mixture of CTS23 and CTS31 MASs. We did this to test the compatibility of the different releases working together and to further expose the CICSPlex to the z/OS testing already in progress. About three weeks later, we completed this migration.

Performing the migration to CICS TS 3.1

This section describes how we migrated to CICS TS 3.1.

Preparing for migration

Before we began the actual migration process, we did some preparatory work in the following areas:

Backing up our data: Even though we created new files and data sets, as a precaution, we first took backups of all of the CSDs and data repositories.

Alias's: We defined new aliases for CTS31.

Loading the CTS31 product libraries: We set up and ran the SMPE jobs to load the CTS31 product libraries. We then ran a copy job to bring the build libraries over to our production systems.

Allocating supporting PDS's: We created copies of all our supporting libraries (JCL, SYSIN, TABLEs, and so on). We reviewed these and updated accordingly with all the necessary CTS31 changes.

Customizing the CICS region data sets: We customized the jobs in *hlq*.SDFHINST(DFHDEFDS) and *hlq*.SEYUINST(EYUDEFDS) to define all of the region data sets and submitted them a number of times. Depending on your environment, you might need to alter the default file sizes. We increased the file sizes for the DFHGCD, DFHINTRA and DFHTEMP data sets.

Reviewing and reassembling tables: We reviewed and reassembled any tables we had modified. We stopped using a ″customized″ SRT, since all of our customizations are now included in the default SRT.

Updating SYS1.PARMLIB and APF-authorizing program libraries: In SYS1.PARMLIB, we updated LINK list and LPA list. We APF-authorized the following program libraries:

- *hlq*.SDFHAUTH
- *hlq*.SDFHLINK
- *hlq*.SDFJAUTH
- *hlq*.SEYUAUTH
- *hlq*.SEYULINK

In PROCLIB, we reviewed and updated all our procs for the CTS31 changes.

Migrating CICSPlex SM

If you are migrating to CICSPlex SM for the first time, CPSM consists of the following parts on each system:

- CAS (coordinating address space)
- CMAS (CICS-managed address space)
- CPSM code running in each CICS region (MAS), which communicates with the CMAS, sometimes referred to as the ″agent″ code. On any specific z/OS system, all three parts of CPSM must be at the same release level. As long as all CPSM components on any single z/OS system are at the same level, you can run mixed levels of CPSM on different systems within a sysplex.

Note: Remember, DFHIRP must be at the highest level of the code.

Migrating the CASs

We reviewed the steps documented in *CICS Transaction Server for z/OS Migration Guide* to migrate the CASs.

Steps for migrating the CASs

We did the following to migrate the CASs:

1. Defined a new BBIPARM parameter repository data set for CPSM 3.1 We reviewed the JCL in the EYUDEFDS member, customized the JCL statements that allocate the CAS parameter library (EYUIPARM) data set, and then submitted it. The BBIPARM DD name contains the cross-system definitions for CPSM.

   Note: CASs running at different levels cannot share the same BBIPARM data set.

2. Updated our TSO signon procedures to point to the new data sets for the new release of CPSM 3.1

3. Reviewed the JCL in the EYUCAS member for any changes to the CAS startup procedure.

   Because there were no changes, we simply made a copy of our old procedure and updated the data set names to use our new high-level qualifiers. (The EYUCAS member resides in the *hlq*.SEYUINST library.)

4. Started the CAS

5. Defined the CAS and updated the parameter repository (BBIPARM), as follows:
   a. From the TSO EUI address space (CPSM), selected option 1 to invoke the PLEXMGR view
   b. Invoked the CASDEF view, which put us into the browse mode
   c. Entered the EDIT command to change to edit mode
   d. Entered the C action command to select our CAS, which took us to the **Change CAS System Definition** panel
   e. Made the appropriate changes to our environment

   Note: When the CAS first comes up, it takes a default group name of EYUGR310. We changed our XCF group name to EYUGP310 for our production CASs, (EYUGT310 for test). Remember, as in any migration, CASs running at different release levels cannot communicate with each other.

Migrating the CMASs

Steps for migrating the CMASs

We did the following to migrate the CMASs:

1. Defined a new CSD

2. Updated the CSD with CPSM 3.1 level resource definitions and the CICS startup group list.

   We did this by running the DFHCSDUP utility with the UPGRADE command, as discussed in *CICS Operations and Utilities Guide*.

3. Updated the CICS system initialization table (SIT) overrides as follows:
   - changed GRPLIST parameter to point to the new CPSM 3.1 group list (EYU310L0)

4. Reviewed our CICS resource definition tables, which we updated earlier

5. Converted the CPSM data repository to the CPSM 3.1 level by running the EYU9XDUT utility, as discussed in *CICS Transaction Server for z/OS Installation Guide*

6. Reviewed the JCL in the EYUCMAS member for any changes to the CMAS startup procedure.

   Because there were no changes, we simply made a copy of our old procedure and updated the data set names to use our new high-level qualifiers.

7. Updated the MAS JCL to point to the new CPSM data sets, in order to identify the new CMAS code to the MAS regions.

Our CMASs were then ready to start.

Migrating the MASs

Steps for migrating the MASs

We reviewed the steps documented in CICS Transaction Server for z/OS Migration Guide to migrate the MASs. Many of the steps are similar to the steps we followed to migrate the CASs and CMASs.

We did the following to migrate the MASs:

1. Defined a new CSD and copied our application groups from the old CSD

2. Upgraded the CSD using ″UPGRADE USING(EYU964G1)″ for CPSM 3.1. We also removed groups for previous releases of CPSM from the group list

3. Reviewed our CICS resource definition tables, which were updated earlier

4. Copied the JCL for our MAS startup procedure and changed the library names to use our new high-level qualifiers;

   Reviewed the LE libraries we had in RPL concatenation.

5. Before release CTS23, JVM profiles were stored in a PDS member. In CICS TS 2.3 and later, they are stored in an filesystem directory pointed to by the JVMPROFILEDIR system initialization parameter. If you are migrating from a release prior to CTS23, you will need to make the appropriate JAVA changes.

   Note: We keep our JVM profiles outside the filesystem shipped with CTS31, so that they would not be overridden with a CTS31 filesystem at maintenance time.

Our MASs were then ready to start.

Migrating the Web User Interface (WUI)

As stated in the *CICS Transaction Server for z/OS Migration Guide*, both the Web User Interface and the CMAS it connects to must be at the highest level of CICSPlex SM within the CICSplex. This means that both must be at the same level as the maintenance point CMAS.

Since the CICS system that acts as the Web User Interface is just another MAS, use the same steps above for migrating a MAS:

1. Migrate the MAS that acts as the Web User Interface.
2. Update the CSD with the Web User Interface group (EYU310G1).
3. Migrate the contents of the Web User Interface server repository (EYUWREP) as documented in the *CICS Transaction Server for z/OS Migration Guide*.

Experiences with migrating to CICS TS 3.1

With the exception of the usual typos and unrelated sysplex issues, this migration went very well. As we did during the last migration, we stopped and compared CPSM views, across the different releases of CPSM. We did NOT find any discrepancies or problems. In fact, we did NOT find any problems during this migration.

Setting up CICS Java in our CICS TS 3.1 environment

This section describes our experiences with setting up CICS Java in our CICS TS 3.1 environment on z/OS. We found this a bit tricky as we were CICS programmers and not Unix System Services or Java programmers. This section is written from the CICS programmer's point of view who is new to Unix System Services and Java. You may need to contact your Unix System Services team (and various others from a security and infrastructure perspective) to complete these activities.

This section will describe the activities necessary for

- setting up CICS Java (JCICS class) in our existing CICS TS 3.1 environment
- setting up an filesystem for production CICS applications
- building the CICS samples for testing in our environment.

We used the following manuals as guides for setting this up.

- *CICS Transaction Server for z/OS Installation Guide*, GC34-6224
- *CICS Transaction Server for z/OS V3 R1 Java Applications in CICS – SC34-6440-00*
- z/OS library, Unix System Services bookshelf

The following is a list of activities we needed to perform in our environment to complete the install and testing:

1. "Setting up MVS components for CICS Java" on page 61
2. "Determining a filesystem naming convention for CICS TS 3.1, application code and CICS Java logs (stderr,stdout,stdin) directories" on page 61
3. "Setting up filesystems for CICS Java" on page 61
4. "Setting up BPXPRMxx to include new CICS Java filesystem definitions" on page 63
5. "Setting up our JVM Profile directory in the CICS Java application filesystem" on page 64
6. "Setting up our JVM Profile in our JVM Profile directory" on page 64
7. "Setting up our JVM properties file to be used by JVM profiles" on page 65
8. "Setting up the CICS system initialization table (SIT) for CICS/Java" on page 65
9. "Setting up Java samples to run in our environment" on page 65
10. "Setting up symlinks to manage our CICS TS 3.1 filesystem service activities" on page 63

Setting up MVS components for CICS Java

We set up the MVS components for CICS Java by adding *hlq*.SDFJAUTH to both the APF list and to the steplib for our CICS region start up proc or job. See *CICS Transaction Server for z/OS Installation Guide*, GC34-6224 for additional information.

Determining a filesystem naming convention for CICS TS 3.1, application code and CICS Java logs (stderr,stdout,stdin) directories

We used the following naming convention for our filesystems:

CICS TS 3.1 dumped copy of the SMP/E build file system
MVSBUILD.CICSTS31.HFS

Production CICS TS 3.1 filesystem
OMVS.CICSTS31.date.FS (where date was the date the filesystem was copied to our production environment).

Application CICS Java filesystem
OMVS.APPDEV.CICSAPPS.FS

CICS Java logs (stderr,stdout,stdin)
OMVS.CICSJAVA.LOGS.FS

Setting up filesystems for CICS Java

The following sections describe the filesystem setups we used in our environment for CICS Java:

- "Setting up and copying the CICS TS 3.1 filesystem from the build filesystem to our environment" on page 62
- "Setting up of a zFS for CICS Java application code" on page 62
- "Setting up of a zFS for CICS Java logs (stdin, stdout, and stderr)" on page 63

We created separate filesystems for directories holding our CICS Java application code so they would be maintained separate from the CICS Java product code. Additionally, we created another filesystem to hold the logging output, to help further isolate this output (Also see "Some CICS Java Hints and Tips" on page 66 concerning the output and this filesystem!).

Creating directories for mount points

We also set up the directories used for our new filesystem, CICS TS 3.1 filesystem, CICS Java application filesystem and CICSLOGS filesystem. You may need to contact your Unix System Services team and various others from a security and infrastructure perspective to get this activity completed.

The following directories were used for our set up:

- /cicsts/cicsts31 for our CICS TS 3.1 filesystem
- /appdev/cicsapps for our CICS Java
- /cicsjava/logs for our stderr, stdout, stdin logs

We first created the directories in our environment using a combination of OMVS and ISHELL.

From OMVS:

```
mkdir cicsts
cd /cicsts
mkdir cicsts31
chmod xxx cicsts31 (change permission bits to xxx (correct security settings for your env)
```

Setting up and copying the CICS TS 3.1 filesystem from the build filesystem to our environment

Our procedure for moving the build CICS TS 3.1 filesystem to our plex was to run the following jobs:

1. HFSCOPY, takes a copy of the dumped CICS TS 3.1 SMP/E build filesystem and moves it to our local DASD
2. HFSRESTR, restores the file and renames it to what we will be using in our environment.

We moved our Build CICS TS 3.1 filesystem to local a filesystem.

The following was the HFSCOPY job we used:

```
//HFSCOPY JOB ...
//STEP1    EXEC PGM=ADRDSSU,TIME=60,REGION=4096K
//SYSPRINT DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//IN DD DISP=SHR,VOL=SER=bldpak,UNIT=3390
//OUT DD DISP=SHR,VOL=SER=prdpak,UNIT=3390
//SYSIN    DD *
 COPY LOGINDDNAME(IN) OUTDD(OUT) -
 DATASET(INCLUDE(MVSBUILD.CICSTS31.HFS)) -
 ALLEXCP ALLDATA(*) TOL(ENQF) -
 CATALOG
```

The following is a JCL example that restores our filesystem to our production HFS:

```
//HFSRESTR JOB...
//RSTCICS EXEC PGM=ADRDSSU,TIME=60,REGION=4096K
//SYSPRINT DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//DDI      DD DSN=MVSBUILD.CICSTS31.HFS,DISP=(OLD,KEEP),
//         UNIT=3390,VOL=SER=prdpak
//SYSIN    DD *
 RESTORE DATASET(INCLUDE(OMVS.HFS.AQTS.MVSBUILD.CICSTS31)) -
      INDDNAME(DDI) -
      TOL(ENQF) -
      STORCLAS(yourstorclas) -
      MGMTCLAS(yourmgmtclas) -RENAMEU(OMVS.HFS.AQTS.MVSBUILD.CICSTS31, 'OMVS.CICSTS31.MAY2505.FS')  REPLACE CATALOG
```

From ISHELL, we mounted our CICS TS 3.1 filesystem (OMVS.CICSTS31.MAY2505.FS) as the /cicsts/cicsts31 directory.

Setting up of a zFS for CICS Java application code

We created our filesystem using the following JCL:

```
//ZFSALLOC  JOB ...
//*******************************************************************
//* FUNCTION: DEFINE A ZFS AGGREGATE
//*******************************************************************
//DEFINE  EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//AMSDUMP  DD SYSOUT=*
//SYSIN    DD *
 DEFINE CLUSTER (NAME(OMVS.CICSAPPS.FS) -
  LINEAR CYL(100 10) SHAREOPTIONS(2) -
  STORCLAS(yourstorclas))
//*******************************************************************
//* FUNCTION: FORMAT VSAM LINEAR DATASET AS A HFS COMPAT AGGREGATE
//*******************************************************************
//COMPAT  EXEC PGM=IOEAGFMT,REGION=0M,
// PARM=('-aggregate OMVS.CICSAPPS.FS   -compat')
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
```

```
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*
//*
```

We created /appdev/cicsapps through OMVS and as a mount point for the file system. We mounted 'OMVS.APPDEV.CICSAPPS.FS' at /appdev/cicsapps through ISHELL. For those familiar with mounting file systems, please use your normal procedures.

Setting up of a zFS for CICS Java logs (stdin, stdout, and stderr)

The following is the JCL we used to create our filesystem to be used as /cicsjava/logs:

```
//ZFSALLOC  JOB ...
//*******************************************************************
//* FUNCTION: DEFINE A ZFS AGGREGATE
//*******************************************************************
//DEFINE  EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//AMSDUMP  DD SYSOUT=*
//SYSIN    DD *
 DEFINE CLUSTER (NAME(OMVS.CICSJAVA.LOGS.FS) -
  LINEAR CYL(100 0) SHAREOPTIONS(2) -
  STORCLAS(yourstorclas))
//*******************************************************************
//* FUNCTION: FORMAT VSAM LINEAR DATASET AS A HFS COMPAT AGGREGATE
//*******************************************************************
//COMPAT  EXEC PGM=IOEAGFMT,REGION=0M,
// PARM=('-aggregate OMVS.CICSJAVA.LOGS.FS   -compat')
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*
//*
```

For CICS Java logs, we also created /cicsjava/logs through OMVS and set up a mount point as 'OMVS.CICSJAVA.LOGS.FS'.

Setting up BPXPRMxx to include new CICS Java filesystem definitions

We set up our BPXPRMxx member to include our new filesystem:

```
MOUNT FILESYSTEM('OMVS.CICSTS31.MAY2505.FS') TYPE(HFS)
   MODE(RDWR) MOUNTPOINT('/cicsts/cicsts31')
 MOUNT FILESYSTEM('OMVS.APPDEV.CICSAPPS.FS') TYPE(HFS)
   MODE(RDWR) MOUNTPOINT('/appdev/cicsapps')

MOUNT FILESYSTEM('OMVS.CICSJAVA.LOGS.FS') TYPE(HFS)
  MODE(RDWR) MOUNTPOINT('/cicsjava/logs')
PARM('FSFULL(85,5)')
```

Setting up symlinks to manage our CICS TS 3.1 filesystem service activities

We set up symlinks in our environment to make it easier for us to pick up service. We have a symlink set up in our /cicsts directory called *current* which will point to our latest CICS TS 3.1 service. We also used symlinks in our Java environment. In our examples, we used the most current Java 1.4 by specifying a pointer to /java/current14.

We restored each copy of the SMP/E build file system to a unique file system and mounted it at a unique location in the /cicsts/cicsts31 directory. For example, a service level may be copied to a file system named OMVS.CICSTS31.MAR07.FS and mounted at /cicsts/cicsts31/mar07.

To use the example service level above, our /cicsts/current symlink pointed to /cicsts/cicsts31/mar07. By using the /cicsts/current symlink in our configuration setups (for example; DFHJVMCD), only the symlink needs to be updated, rather than all of the individual files.

To create a symlink, we used the following command in OMVS:

```
ln -s /cicsts/cicsts31/mar07  /cicsts/current
```

Note: If the /cicsts/current symlink currently exists, you cannot alter it. To change it, you must first remove it and then recreate it.

Setting up our JVM Profile directory in the CICS Java application filesystem

We created a new directory in /appdev/cicsapps to hold our JVM profiles as we did not want them to be over written each time we picked up service. We went into OMVS and changed the directory to /appdev/cicsapps and issued *mkdir JVMProfiles* (notice the mixed case). Java set up is case sensitive and you must be very careful setting this up from a file systems perspective and in the CICS program definitions.

We then copied, updated, and moved our JVM profiles to our JVMPROFILE directory.

Setting up our JVM Profile in our JVM Profile directory

After our new directory was set up, we moved our sample DFHJVMxx members into the new directory. There is a job sent with the product to do this but we manually moved the files from XDFHENV to our filesystem: /appdev/cicsapps/JVMProfiles.

Following is an example of our default JVMProfile:

```
# DFHJVMPR
#
#
# ********* CICS-specific parameters ***********
#
WORK_DIR=/appdev/cicsapps
INVOKE_DFHJVMAT=NO
REUSE=RESET
#
# Specify the CICS and JVM install locations
#
CICS_DIRECTORY=/cicsts/current/
JAVA_HOME=/java/current14/
JVMPROPS=/appdev/cicsapps/props/jvmprops.ejb
LIBPATH=\
   /cicsts/current/lib:\
   /java/current14/bin/:\
   /java/current14/bin/classic:\
   /appdev/cicsapps/lib :\
   /cicsts/current/samples/dfjcics
#
STDIN=/cicsjava/logs/dfhjvmin.&APPLID.stdin
STDERR=/cicsjava/logs/dfhjvmerr.&APPLID.stderr
```

```
STDOUT=/cicsjava/logs/dfhjvmout.&APPLID.stdout
#
CLASSPATH=.
#
# ********* Java standard options **************
VERBOSE=NO
#
# ********* Java non-standard options **********
#
Xcheck=NO
Xdebug=NO
Xms=16M
Xmx=32M
Xnoclassgc=NO
Xoss=4M
Xss=512K
Xverify=none
```

Note: &APPLID is correlated to a CICS subsystem.

Setting up our JVM properties file to be used by JVM profiles

Following is an example of our JVM properties file in /appdev/cicsapps/props/jvmprops.ejb:

```
java.compiler=jitc
#java.compiler=none
ibm.jvm.shareable.application.class.path=/appdev/cicsapps/lib
com.ibm.CICS.iiop.PublishToJNDI=false
com.ibm.CICS.ejs.PublishToShelf=true
```

Setting up the CICS system initialization table (SIT) for CICS/Java

We added the following statements to our CICS TS 3.1. to support CICS/Java:

```
JVMPROFILEDIR=/appdev/cicsapps/JVMProfiles,
MAXJVMTCBS=5,     NEW FOR CICS TS  (this is the default)
```

We also created a new Java group to contain all our Java applications called JWORKLD and added them to our CICS TS 3.1 start up through CPSM BAS.

Setting up Java samples to run in our environment

As part of our testing of the set up, we installed a few CICS Java samples to verify the set up. We found this quite tricky as we were CICS programmers and not Unix System Services filesystem programmers and our Unix System Services programmer was not a CICS programmer. We used the "Build the JCICS sample programs" section of *'CICS Transaction Server for z/OS V3 R1 Java Applications in CICS'* book as a guide.

We first installed the DFJ$JVM group into our CICS regions (standard group install).

The first transaction we set up and ran was the JHE1 transaction which requires set up of the Java class (HelloWorld.java) and C language program(DFH$JSAM).

We compiled the C program using an existing C compiler proc.

To build the HelloWorld samples, we followed the instructions defined in the *'CICS Transaction Server for z/OS Java Applications in the CICS V3 R1'* book section named "Building the Java samples". We had some problems with the *make jvm* command mostly related to mixed case issues. Make sure you use the correct case. Using OMVS from the directory named /cicsts/cicsts31/samples/dfjCICS, we issued:

```
make -f HelloWorld.mak jvm
```

The following was the response:

```
154:/cicsts/cicsts31/samples/dfjcics $ make -f HelloWorld.mak jvm
javac -deprecation   -classpath .:/cicsts/cicsts31/lib/dfjcics.jar examples/HelloWorld/HelloWorld.java
javac -deprecation   -classpath .:/cicsts/cicsts31/lib/dfjcics.jar examples/HelloWorld/HelloCICSWorld.java
155:/cicsts/cicsts31/samples/dfjcics $
```

We checked and we now had .class files for both in the /cicsts/cicsts31/samples/dfjCICS/examples/HelloWorld/ directory as: HelloCICSWorld.class and HelloWorld.class

We ran the JHE1 transaction and it worked as indicated in *CICS Transaction Server for z/OS V3 R1 Java Applications in CICS – SC34-6440-00*.

We ran these samples as a test after each new CICS TS 3.1 filesystem install and did not want to go back and run the make file each time. We have since copied the HelloWorld and HelloCICSWorld java and class files to our appdev/cicsapps/lib/examples directory as:

```
/appdev/cicsapps/lib/examples/HelloWorld and /appdev/cicsapps/lib/examples/HelloCICSWorld
```

which is concatenated as part of our libpath in our jvmprofile. This also requires a program definition change so we copied the definitions from the DFH$JVM to another group called jworkld and modified the program definition for this change.

We set these transactions up as part of our CICS Gumbo TPNS workload to be run on a regular basis.

Some CICS Java Hints and Tips

Mixed case: Everything in a filesystem is mixed case so be very careful when executing and setting something up that you use the exact case specific wording. You must also set up your CICS definition in mixed case to match the name exactly.

Java packages and directories: Something we didn't know as we were new to Java; your Java code will have a package statement in it. This package statement is a path statement off of your libpath definition in your jvmprofile.

For example, the HelloWorld.java Java code has the following package statement: package examples.HelloWorld. Our libpath in our jvmprofile(DFHJVMPR) for HelloWorld is: /appdev/cicsapps/lib Therefore our HelloWorld class file must be in the following directory: /appdev/cicsapps/lib/examples/HelloWorld/ to execute successfully.

You will need /java/bin/classic in your libpath to pick up DLL libjvm.so. We had a little trouble finding this at first. If you don't have it in your libpath, you will get the following error:

```
DFHSJ0503 04/13/2006 13:38:45 CICSCA1A Attempt to load DLL libjvm.so has failed module not found.
```

CICS Java output (stdout / stderr): Some of our Java applications were encountering errors and the Java code was written such that output is written to stdout, stderr. We opened a problem on this and it was determined that this was working as designed. If a CICS transaction program can not write to an filesystem to output the stdout, stderr information requested by the program, the transaction will keep trying to write the data out until space is available in the filesystem. This kind of condition causes the CICS region to use excessive CPU cycles as CICS continually checks to see if data can be written and the transaction will not end until

the data can be written. The recommendation from development was to add an indicator in our BPXPRMxx so that we would be notified when the filesystem was 85% full. To better manage this, we added a separate filesystem to manage the stdout, stdin for each of our JVM profiles. This filesystem is set up in BPXPMRxx to notify the operator if the filesystem is 85% full so that we could react before the filesystem gets into a full condition.

CICS application dump suppression: When you are running Java applications you may want to consider suppressing your Java application abends during testing. We set up our regions to take special action for the following dump codes as they are application related much like you might do for an ap0001 abend. In our case, we suppress dumps for SJ0001,SM0002, KERNDUMP and have SR0001 set up for special handling due to a problem we had while trouble shooting. Following is an example of our cardin:

```
CEMT SET SYD(SJ0001) REM\
CEMT SET SYD(SJ0001) NOSYSDUMP NOSH MAX(0) ADD\
CEMT SET SYD(SM0002) REM\
CEMT SET SYD(SM0002) NOSYSDUMP NOSH MAX(0) ADD\
CEMT SET SYD(KERNDUMP) REM\
CEMT SET SYD(KERNDUMP) NOSYSDUMP NOSH MAX(0) ADD\
CEMT SET SYD(SR0001) REM\
CEMT SET SYD(SR0001) SYSDUMP NOSH MAX(1) ADD\
```

Chapter 6. Migrating to DB2 Version 8

This chapter addresses the processes and experiences encountered during the migration of the Integration Test production 12 way DB2 data sharing group DB1G from DB2 Version 7 to Version 8 (composed of members DBA1, DBB1, DBC1, DBD1, DBE1, DBF1, DBG1, DBH1, DBI1, DBZ1, DB81, and DB91).

We used the *DB2 Installation Guide* for our migration. Whenever we reference a **Migration Step** in **bold** in this chapter, we are referencing the same migration steps that are in the *DB2 Installation Guide*.

Migration considerations

Before you migrate to DB2 Version 8, note the following points:

- Migrations to DB2 Version 8 are only supported from subsystems currently running DB2 Version 7; unpredictable results can occur if a migration is attempted from another release of DB2.
- **Migration Step 24** is an optional step that is used to verify the DB2 Version 8 subsystem after it is in compatibility mode. For this step, only the following selected Version 7 IVP jobs can be executed:
  1. Version 7 phase 2 IVP applications
     a. DSNTEJ2A - All steps except the first two
     b. DSNTEJ2C - Only step PH02CS04, statement RUN PROGRAM(DSN8BC3) PLAN(DSN8BH61), is to be executed
     c. DSNTEJ2D - Only step PH02DS03, statement RUN PROGRAM(DSN8BD3) PLAN(DSN8BD61), is to be executed
     d. DSNTEJ2E - Only step PH02ES04, statement RUN PROGRAM(DSN8BE3) PLAN(DSN8BE61), is to be executed
     e. DSNTEJ2F - Only step PH02FS03, statement RUN PROGRAM(DSN8BF3) PLAN(DSN8BF61), is to be executed
     f. DSNTEJ2P - Execute step PH02PS05
  2. Version 7 phase 3 IVP applications
     a. ISPF-CAF applications, with the exception of DSNTEJ3C and DSNTEJ3P.

  If you want to run these IVPs as part of the verification of DB2 Version 8 compatibility mode, they must first be run under Version 7 in their entirety before you start the Version 8 migration process and must remain available for use after you complete the migration to Version 8 compatibility mode.
- Before you begin the migration process to DB2 Version 8, verify that if you are using CFCC levels 7 or 8, they are at the proper service levels - 1.06 and 1.03, respectively; to avoid the possibility of data corruption. Our three CFs are all running higher levels of service, as shown below:

```
CF1: CFCC RELEASE 14.00, SERVICE LEVEL 00.14
CF2: CFCC RELEASE 14.00, SERVICE LEVEL 00.14
CF3: CFCC RELEASE 13.00, SERVICE LEVEL 04.08
```

  Other than these requirements, no other CFCC service level requirements exist.
- Examining ″Migration Considerations″ of the *DB2 Installation Guide*, the following items are of particular interest:
  – Global temporary tables require a 16K buffer pool.

  - Declared temporary tables require at least one table space to have a page size of 8K or greater in the temporary database.
  - Support for DB2-established data spaces for cached dynamic statements is removed; you can no longer specify parameters EDMDSPAC and EDMDSMAX during migration.
  - Consider increasing IDBACK and CTHREAD subsystem parameters -- utilities might now require additional threads.
  - Support for DB2-established stored procedure address spaces is removed (that is you can no longer specify NO WLM ENVIRONMENT when creating or altering a stored procedure); existing stored procedures can continue to run in a DB2-established stored procedure address space, but you should migrate such procedures to a WLM environment as soon as possible.
  - A default DSNHDECP module (found in SDSNLOAD) is no longer shipped with DB2. DSNTIJUZ must be used to create a customized DSNHDECP, which is then placed in SDSNLOAD and SDSNEXIT.
- During the migration of the first member of a data sharing group to DB2 Version 8, other members of the data sharing group can be active, although they can experience delays or time-outs when accessing catalog objects as these objects might be locked because of the migration process. Upon completion of the migration process for all data sharing group members, you must update TSO and CAF logon procedures to reference the DB2 Version 8 libraries exclusively.

Premigration activities

Before migrating to DB2 Version 8, application of the fallback SPE to all members of the Version 7 data sharing group is necessary.

Also, ensure that the size of the work file database is sufficiently large enough to support the sorting of indexes when migration job DSNTIJTC is run.

After making a backup of the current logon procedure in use, we updated the procedure to reflect the following DB2 Version 8 concatenations before invoking the DB2 installation CLIST:

- DB2.DB2810.SDSNSPFM was concatenated to ISPMLIB.
- DB2.DB2810.SDSNSPFP was concatenated to ISPPLIB.
- DB2.DB2810.SDSNSPFS was concatenated to ISPSLIB.
- DB2.DB2810.SDSNSPFT was **not** concatenated to ISPTLIB, as DB2 online help was not installed.

In some instances it might be necessary to issue the following RACF command for the SDSNLOAD library:

```
ralter program * addmem('hlq.SDSNLOAD'/******/NOPADCHK)
```

This might prevent error messages like the following:

```
CEE3518S The module DSNAOCLI was not found in an authorized library.
CSV042I REQUESTED MODULE DSNAOCLI NOT ACCESSED. THE MODULE IS NOT PROGRAM CONTROLLED.
```

For our setup, we issued the RACF command on behalf of LDAP.

After we logged on with the updated logon procedure, we invoked the installation CLIST DSNTINST from the ISPF Command Shell by entering the following command:

```
ex 'db2.db2810.sdsnclst(dsntinst)' from ISPF option 6.
```

We filled in the first panel DSNTIPA1 as shown in Figure 7:

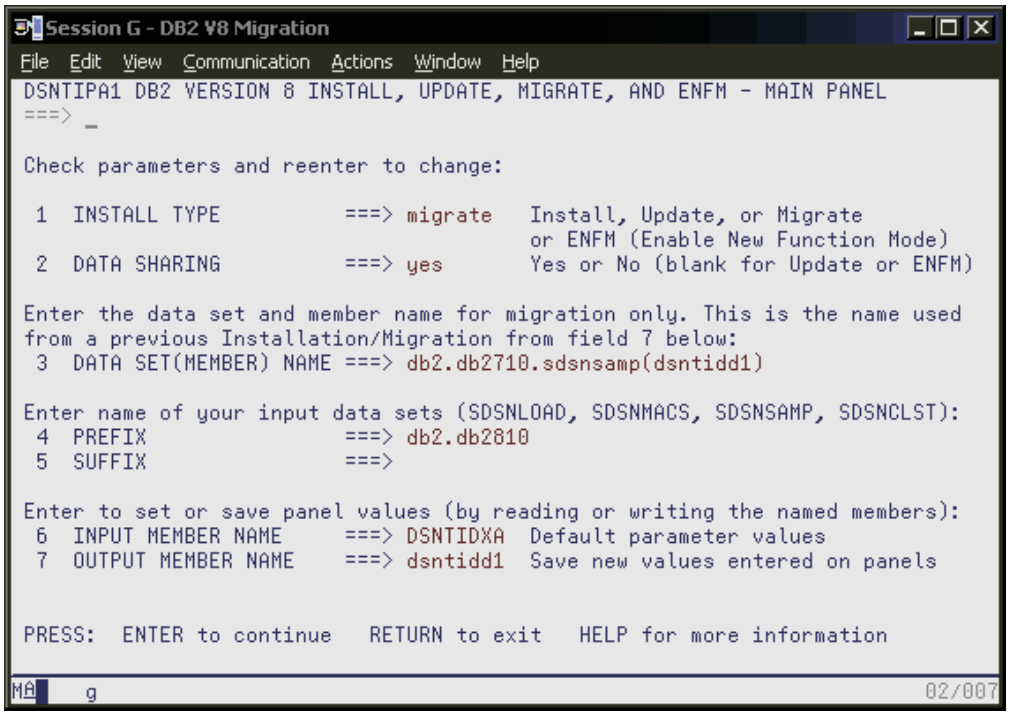


Figure 7. DSNTIPA1

When we pressed enter, the pop-up screen DSNTIPP2 appeared as shown in Figure 8 on page 72:

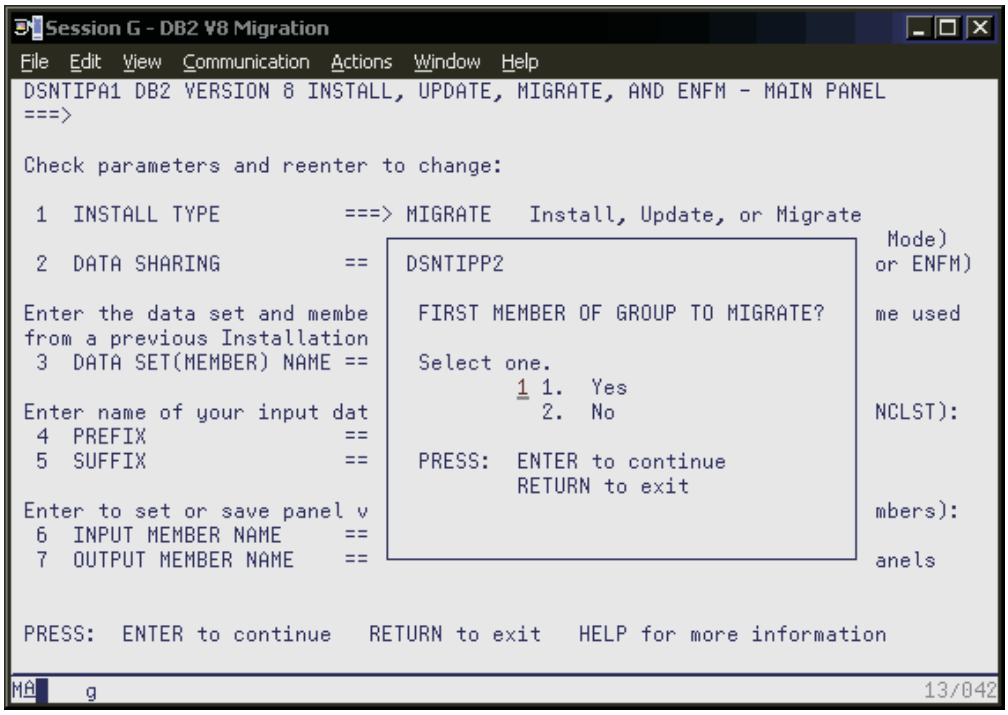


Figure 8. DSNTIPP2

We entered '1' to reflect that this was the first member of the data sharing group to be migrated to DB2 Version 8. From this point, we scrolled through the panels and accepted the existing values; upon completion, we placed the tailored CLISTs in DB2.DB2810.DBD1.SDSNSAMP and DB2.DB2810.NEW.SDSNTEMP, as shown in Figure 9 on page 73.

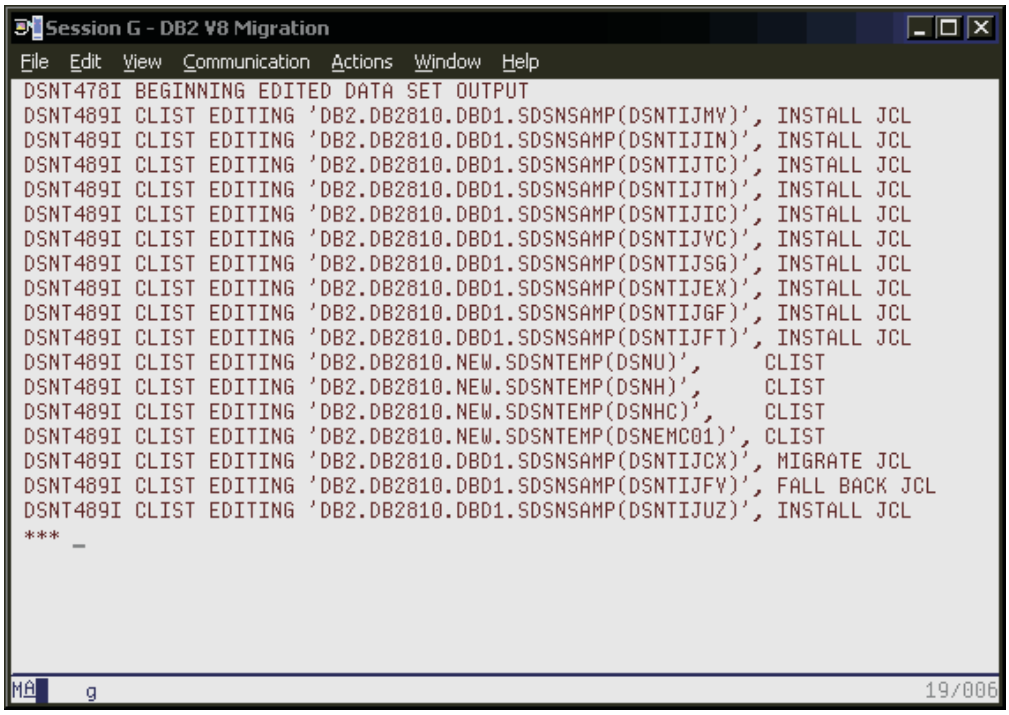


Figure 9. Tailored CLISTs placed in DB2.DB2810.DBD1.SDSNSAMP and DB2.DB2810.NEW.SDSNTEMP

Migrating the first member to compatibility mode

After we reviewed the topics outlined in **Migration Step 1**, we made the following observations:

- Ensured that the IVP jobs and sample database objects for DB2 Version 7 are still available for use. Failure to do so will prevent verifying that a successful migration to DB2 Version 8 compatibility mode has been made.
- Ensured that no utilities are running before migrating to DB2 Version 8. When the migration to Version 8 compatibility mode has been completed, any outstanding utilities that were started under Version 7 cannot be restarted or terminated under Version 8.
- EBCDIC and ASCII CCSID values must be nonzero. Note that this issue was addressed by several DB2 Version 7 PTFs (such as UQ74294), so ensure that the maintenance level for DB2 Version 7 is current before migrating to DB2 Version 8.

Migration Step 2 concerns the optional step of executing DSN1CHKR to verify the integrity of the DB2 directory and catalog table spaces that contain links or hashes. Before executing, we had to stop the following table spaces (through option `7, DB2 Commands` of the `DB2I Primary Option Menu`):

```
DSNDB06.SYSDBASE
DSNDB06.SYSDBAUT
DSNDB06.SYSGROUP
DSNDB06.SYSPLAN
DSNDB06.SYSVIEWS
DSNDB01.DBD01
```

DSN1CHKR was then executed without incident. The table spaces were restarted which had been stopped.

DB2 recommends running DSN1COPY with the CHECK option on all of the catalog and directory table spaces. To accomplish this, we created a job called CKDIRCAT and ran the job with DB2 down. No problems occurred and we brought DB2 back up.

Next, we ran the CHECK index utility against all catalog and directory indexes (we created a job called CKIDRCAT). Again, no major problems occurred (we received a RC = 4).

Finally, to ensure that there were no STOGROUPs defined with both specific and nonspecific volume ids, we ran the following query:

```
SELECT * FROM SYSIBM.SYSVOLUMES V1
  WHERE VOLID <> '*' AND
  EXISTS (SELECT * FROM SYSIBM.SYSVOLUMES V2
            WHERE V1.SGNAME = V2.SGNAME AND V2.VOLID='*');
```

The query did not return any rows.

Migration Step 3 is an optional step to determine which plans and packages are to be rendered not valid as a result of migrating to DB2 Version 8. To accomplish this, we ran the following queries:

```
SELECT DISTINCT DNAME
  FROM SYSIBM.SYSPLANDEP
  WHERE BNAME IN('DSNVVX01','DSNVTH01') AND
        BCREATOR = 'SYSIBM' AND
        BTYPE IN ('I','T')
  ORDER BY DNAME;


SELECT DISTINCT COLLID, NAME, VERSION
  FROM SYSIBM.SYSPACKDEP, SYSIBM.SYSPACKAGE
  WHERE BNAME IN('DSNVVX01','DSNVTH01')
    AND LOCATION = ' '
    AND BQUALIFIER = 'SYSIBM'
    AND BTYPE IN ('I','T')
    AND COLLID = DCOLLID
    AND NAME = DNAME
    AND CONTOKEN = DCONTOKEN
  ORDER BY COLLID, NAME, VERSION;
```

The first query did not produce any rows, while the second generated the results shown in Figure 10 on page 75.

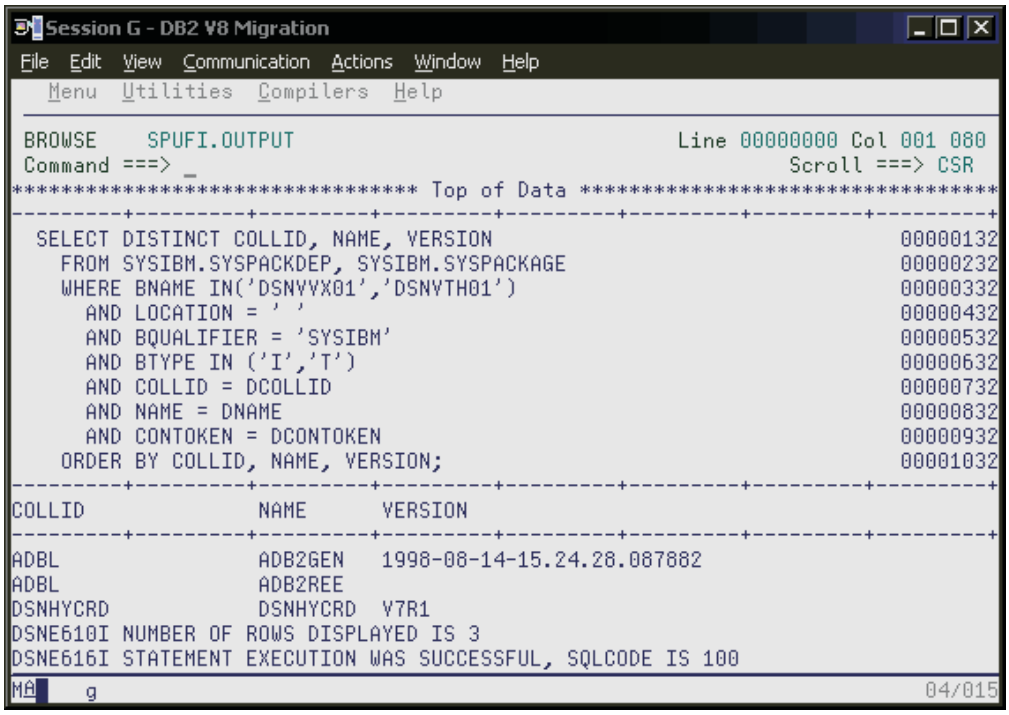


Figure 10. Output from query to find packages that will be invalidated when migrating to DB2 Version 8

Migration Step 4 is another optional step to check for consistency between catalog tables through running the queries contained in DB2.DB2810.SDSNSAMP(DSNTESQ). There are a total of 65 queries contained in this data set. We used the data set as input to SPUFI, it ran with no inconsistencies.

Migration Step 5 addresses performing an image copy of the catalog and directory in case of fallback. The *DB2 Installation Guide* recommends using the Version 7 job DSNTIJIC, however, this job has the following shortcomings:

- Does not place the catalog and directory table spaces in utility status to prevent updates while the image copies are being taken.
- Does not quiesce the catalog and directory spaces to provide a consistent point of recovery (the catalogs are referentially linked).

Using the groundwork laid in DSNTIJIC and before performing the image copies, we added steps to place all catalog and directory spaces in utility status and to quiesce them. In addition, we redirected the image copies to DASD generation data groups (GDGs) rather than to tape. Finally, we added a step to the job to restart the catalog and directory tables spaces read/write after the image copies. The job called IDIRCAT7 and was successfully executed.

In preparation for the post-migration image copy of the catalog and directory, we made similar modifications to the Version 8 job DSNTIJIC. The job was called IDIRCAT8 and included image copy steps for the new catalog table spaces **DSNDB06.SYSALTER** and **DSNDB06.SYSEBCDC**.

Migration Step 6 addresses the following steps necessary to connect DB2 to TSO:

- **Making DB2 load modules available to TSO and batch users** - SDSNEXIT and SDSNLOAD were added to linklist.

- **Making DB2 CLISTs available to TSO and batch users: DSNTIJVC** - Our logon proc QMFPROC must again be updated to add DB2.DB2810.NEW.SDSNCLST to the SYSPROC concatenation. We had to do this **after** we ran the installation job DSNTIJVC (the job that merges tailored CLISTs from prefix.NEW.SDSNTEMP with unchanged CLISTs from prefix.SDSNCLST and places the resulting set of CLISTs in the newly created data set prefix.NEW.SDSNCLST). Since we currently use fixed-block CLIST libraries (use the SYSPROC concatenation in logon proc QMFPROC), we had to modify DSNTIJVC as follows:
  - Changed the SYSIN DD to DUMMY.
  - Changed the allocation of prefix.SDSNCLST to match the data control block (DCB) attributes of our other CLIST libraries; this was accomplished by replacing the DCB attributes for DSNTIVB.SYSUT2 with **DCB=*.SYSUT1**.

  After DSNTIJVC successfully ran, we update logon proc QMFPROC to add DB2.DB2810.NEW.SDSNCLST to the SYSPROC concatenation.
- **Making panels, messages, and load modules available to ISPF and TSO** - We previously added SDSNSPFP, SDSNSPFM, and SDSNSPFS to the ISPF concatenations. In addition, we updated the logon proc QMFPROC to reflect the concatenation of the DB2 English DB2I panels as follows:
  - DB2.DB2810.SDSNPFPE concatenated to ISPPLIB.

Because IMS and CICS connections to DB2 had previously been established, we skipped **Migration Step 7** and **Migration Step 8**.

Migration Step 9 instructed us to stop all DB2 V7 activity or else fallback procedures can fail. Before stopping data sharing group DB1G, we insured that there were no incomplete utilities (**@DBD1 DISPLAY UTILITY(*)**), and that no databases were in restrict or advisory status (**@DBD1 DISPLAY DATABASE(*) SPACE(*) RESTRICT** and **@DBD1 DISPLAY DATABASE(*) SPACE(*) ADVISORY**, respectively); brought down all members of DB1G.

We skipped optional **Migration Step 10 (Back Up your DB2 Version 7 volumes)** and performed **Migration Step 11**, which defines DB2 initialization parameters through DSNTIJUZ. After modifying this job by removing the SMP/E step, we submitted it and it ran successfully.

As subsystem security had already been established, we skipped **Migration Step 12**.

Migration Step 13 defines DB2 V8 to MVS. We examined job DSNTIJMV to see which modifications to the MVS environment were required; they were implemented accordingly. DSNTIJMV performs the following actions:

- Updates IEFSSNxx, APF, and linklist members, which were deemed not necessary as they had been performed previously.
- Step RENAME renames the current DB2 procedures in proclib. We skipped this step, however. The DB2 startup procs for DBD1 are renamed manually (see below).
- Step DSNTIPM adds catalogued procedures to proclib; however rather than directing the output of this step to SYS1.PROCLIB, we directed it to a newly created data set, DB2.DB2810.PROCLIB.

We renamed the startup procs for DBD1 that reside in PET.PROCLIB (as per the RENAME step of DSNTIJMV). Next, we copied the new V8 startup procs for DBD1 from DB2.DB2810.PROCLIB.

For **Migration Step 14**, we successfully ran job DSNTIJIN to define system data sets.

For **Migration Step 15**, we ran the last two steps of job DSNTIJEX to assemble and link edit the access control authorization exit DSNXSXAC and user exit routine DSNACICX (invoked by stored procedure DSNACICS). We skipped the first and second steps that are used to assemble and link edit the signon (DSN3@SGN) and identify (DSN3@ATH) exits because they were not previously implemented.

Because we had previously IPLed the system to pick the V8 early code, we skipped **Migration Step 16**.

For **Migration Step 17**, the DBD1 member of data sharing group DB1G was started successfully. As shown in Figure 11, the level of the data sharing group is now 810 and in compatibility mode (**MODE(C)**). The DB2 level of DBD1 reflects that it is now running DB2 Version 8 code. The DISPLAY GROUP command shows the data sharing group is in compatibility mode and one member is running DB2 Version 8.

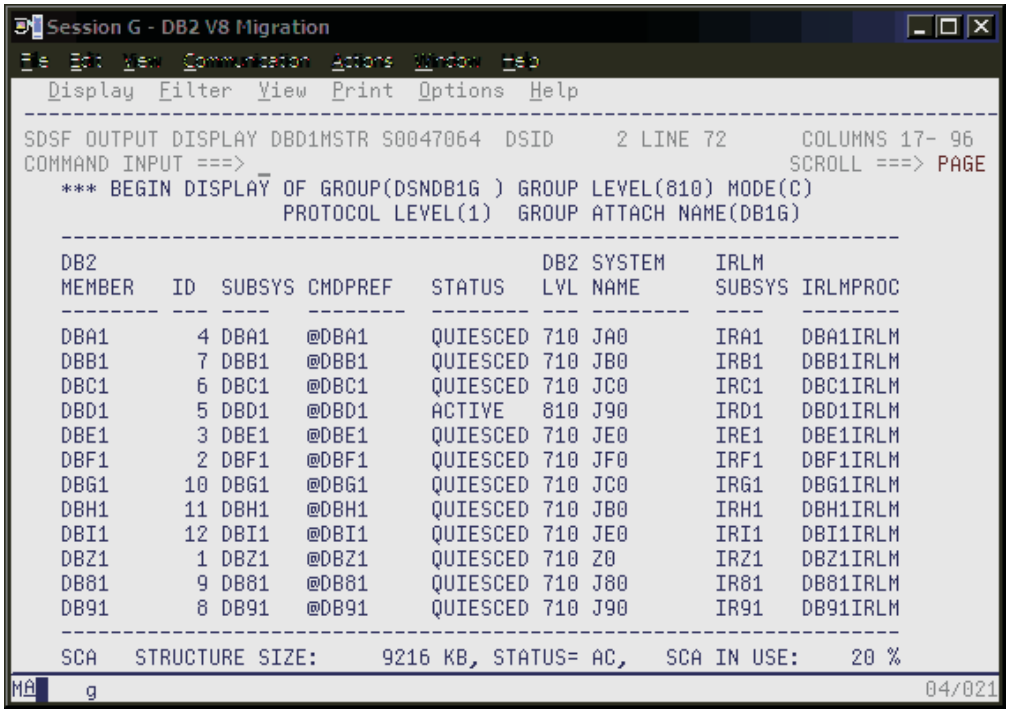


Figure 11. DISPLAY GROUP command

Up on deck next is CATMAINT, as outlined in **Migration Step 18**. We submitted and ran DSNTIJTC successfully. The job periodically issued message DSNU777I in SYSPRINT to indicate migration progress, as shown in Figure 12 on page 78:

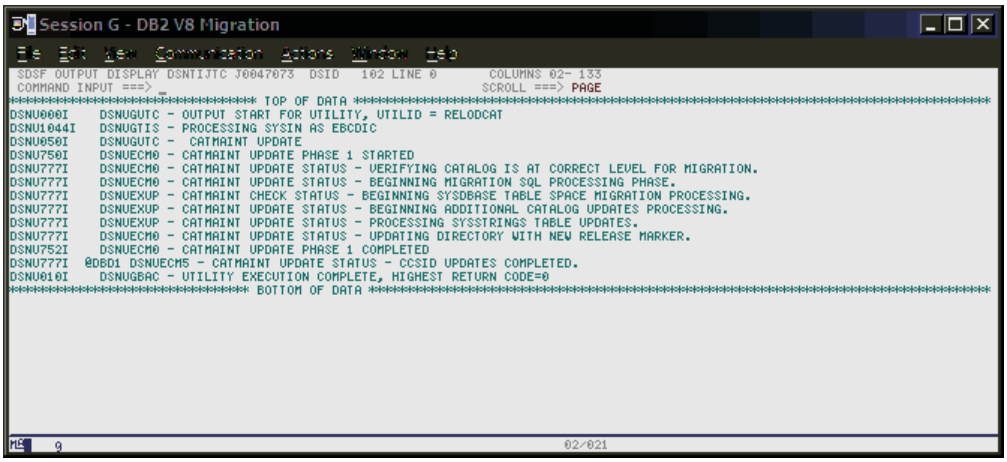


Figure 12. Message DSNU777I displays CATMAINT progress

Migration Step 19 is an optional step to ensure that there are no problems with the catalog and directory after running DSNTIJTC. We used the following steps:

- Ran DSNTIJCX to ensure the integrity of the catalog indexes. The first step produced a return code of 4 as a result of no indexes being found for table space DSNDB06.SYSALTER (these objects will be created during the enabling of New Function Mode). The remaining steps produced a return code of zero.
- Ran DSN1CHKR as in **Migration Step 2** to ensure there were no broken links and that it ran successfully.
- Examined the queries in SDSNSAMP member DSNTESQ for discrepancies; none found.
- Ran DSN1COPY with the CHECK option on the catalog and directory table spaces. The job completed with a return code of 4, the result of step SYSDBASE, which produced the following report:

```
DSN1999I START OF DSN1COPY FOR JOB CKDIRCAT STEP1   SYSDBASE
DSN1989I DSN1COPY IS PROCESSED WITH THE FOLLOWING OPTIONS:
         CHECK/NO PRINT/4K/NO IMAGECOPY/NON-SEGMENT/NUMPARTS=0/NO OBIDXLAT/NO VALUE/NO RESET/     /   /   /
         DSSIZE=   /PIECESIZ=    /
DSN1998I INPUT  DSNAME = DSNDB1G.DSNDBC.DSNDB06.SYSDBASE.I0001.A001  , VSAM
DSN1997I OUTPUT DSNAME = NULLFILE                                     , SEQ
DSN1991I UNCLUSTERED DATA DETECTED.  RID: '0000006D03'X   TABLE: SYSTABLESPACE INDEX KEY: WRKDBD1  DSN32K01
DSN1991I UNCLUSTERED DATA DETECTED.  RID: '0000006D02'X   TABLE: SYSTABLESPACE INDEX KEY: TMPDBD1  TEMPTS01
DSN1994I DSN1COPY COMPLETED SUCCESSFULLY,  00001250 PAGES PROCESSED
```

We submitted a reorg of DSNDB06.SYSDBASE to recluster the data; using DSN1COPY again with the CHECK option against DSNDB06.SYSDBASE then produced a return code of zero.

Before DB2 Version 8, if DBINFO was used to define external functions or procedures, the routine body must be recompiled and rebound to correctly reference ASCII or Unicode CCSID fields in DBINFO. **Migration Step 20** deals with this. The following query can be executed to identify those routines that might need to be recompiled and rebound because they reference ASCII or Unicode CCSID fields in DBINFO:

```
SELECT SCHEMA,NAME FROM SYSIBM.SYSROUTINES WHERE DBINFO='Y';
```

We had no such routines defined on our system.

Because we disabled change data capture on multiple DB2 catalog tables as a result of migrating to Version 8 compatibility mode, we performed **Migration Step 21** to re-enable change data capture on the appropriate catalog tables by issuing the following command:

```
ALTER TABLE SYSIBM.SYSTABLES DATA CAPTURE CHANGES;
```

We received an SQL code of -4700, which basically states that the query could not be executed because the data sharing group was not in New Function Mode. We postponed this step until after all members of the data sharing group were migrated to compatibility mode (see “Migrating to new function mode” on page 86).

We performed **Migration Step 22**, DSNTIJTM, assemble, link-edit, bind, and invoke DSNTIAD, and the job ran successfully.

We ran job DSNTIJSG according to the instructions specified in **Migration Step 23**. It completed successfully (RC=4). Note that when this job is executed, SPUFI can only be invoked from members of the data sharing group that have been migrated to V8 - those remaining on V7 will receive resource unavailable messages until they have migrated to V8, as shown in Figure 13:

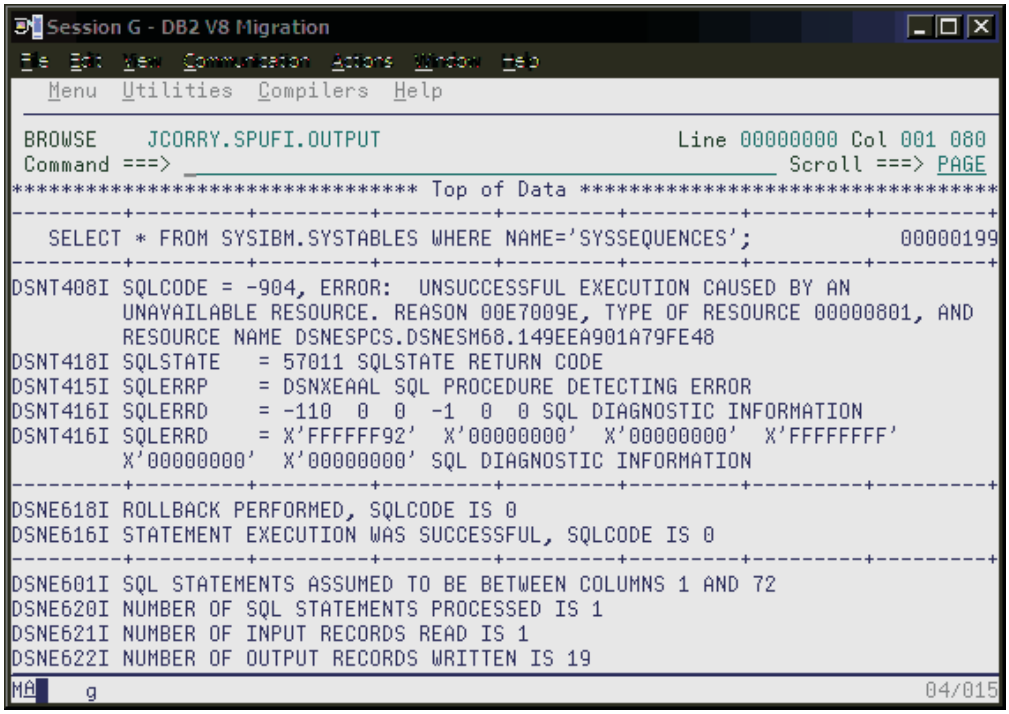


Figure 13. SPUFI is not available for use on DB2 Version 7 members after the execution of DSNTIJSG

Migration Step 24 describes how DSNTIJRX is used to optionally bind the packages for DB2 REXX language support. We did not perform this step because we do not have this feature available.

Because some views might have been marked with view regeneration errors during the migration to Version 8 compatibility mode, we performed **Migration Step 25** and identified the views with the following query:

```
SELECT CREATOR,NAME FROM SYSIBM.SYSTABLES
WHERE TYPE='V' AND STATUS='R' AND TABLESTATUS='V';
```

For those views found to have view regeneration errors, we issued the following command:

```
ALTER VIEW view_name REGENERATE;
```

In **Migration Step 26** we took another image copy of the directory and catalog after they were successfully migrated to V8, and submitted job IDIRCAT8; recall that this job is located in DB2.JOBS and is a modified version of DSNTIJIC (see **Migration Step 5** for details). Execution of IDIRCAT8 completed successfully.

Optional **Migration Step 27** verifies the DB2 Version 8 subsystem that is now in Compatibility Mode. For this step, only the following Version 7 IVP jobs can be submitted:

1. Version 7 Phase 2 IVP Applications
   a. DSNTEJ2A - All steps except the first two
   b. DSNTEJ2C - Only step PH02CS04, statement RUN PROGRAM(DSN8BC3) PLAN(DSN8BH61) is to be executed
   c. DSNTEJ2D - Only step PH02DS03, statement RUN PROGRAM(DSN8BD3) PLAN(DSN8BD61) is to be executed
   d. DSNTEJ2E - Only step PH02ES04, statement RUN PROGRAM(DSN8BE3) PLAN(DSN8BE61) is to be executed
   e. DSNTEJ2F - Only step PH02FS03, statement RUN PROGRAM(DSN8BF3) PLAN(DSN8BF61) is to be executed
   f. DSNTEJ2P - Execute step PH02PS05.
2. Version 7 Phase 3 IVP Applications
   a. ISPF-CAF applications, with the exception of DSNTEJ3C and DSNTEJ3P. Note that you must temporarily place the Version 7 SDSNSPFP panel library ahead of the Version 8 SDSNSPFP panel library in the ISPPLIB concatenation. This permits the plans that were migrated from Version 7 to be used.

Of the IVP jobs listed, we only issued the first, DSNTEJ2A, under Version 7. It is therefore the only IVP that we could use in Version 8 Compatibility Mode. For the record, if it is desired to execute the remaining IVPs as part of the verification of DB2 Version 8 Compatibility Mode, they must first be executed under Version 7 in their entirety before starting the Version 8 migration process and remain available for use after the migration to Version 8 Compatibility mode has been completed.

Before executing DSNTEJ2A, we performed the following actions:

- We updated JOBLIB statements to reflect the DB2 Version 8 load library.
- We edited proc DSN8EAE1 (used by the employee sample table) and copied it from the Version 7 exit library to the Version 8 exit library.

When we completed these tasks, we ran DSNTEJ2A and it produced a return code of 4 as the result of placing tables DSN8D71U.NEWDEPT and DSN8D71U.NEWPHONE in COPY PENDING status. This was as expected.

Finally, optional **Migration Step 28** deals with enabling WLM stored procedures by either executing the installation CLIST in MIGRATE mode or by editing and executing DSNTIJUZ. Additional information on enabling stored procedures is available in the *DB2 Installation Guide* under topic 2.6.24, ″Enabling stored procedures after installation″. Since we had already enabled WLM stored procedures under DB2 Version 7, this step was skipped.

DB2 V7 and V8 coexistence issues

We allowed the data sharing group to run in coexistence mode for several days while we tested various workloads and products for coexistence issues.

It is recommended that a data sharing group remain in coexistence mode for as brief a time period as necessary.

During this period we did not experience any problems.

Migrating the remaining members to compatibility mode

The next member to migrate in the data sharing group to DB2 Version 8 compatibility mode was DBG1. For us, this was a fairly simple process, which entailed the following steps:

1. Executing the installation CLIST.
2. Executing the resultant DSNTIJUZ job.
3. Replacing the Version 7 startup procs for the member being upgraded with their Version 8 equivalents. This is performed by executing DSNTIJMV step DSNTIPM.
4. Starting the member.

So, beginning with the installation CLIST, we ran DSNTINST from the ISPF Command Shell (ISPF option 6) by entering the following command:

```
ex 'db2.db2810.sdsnclst(dsntinst)'
```

We filled in the first panel as shown:

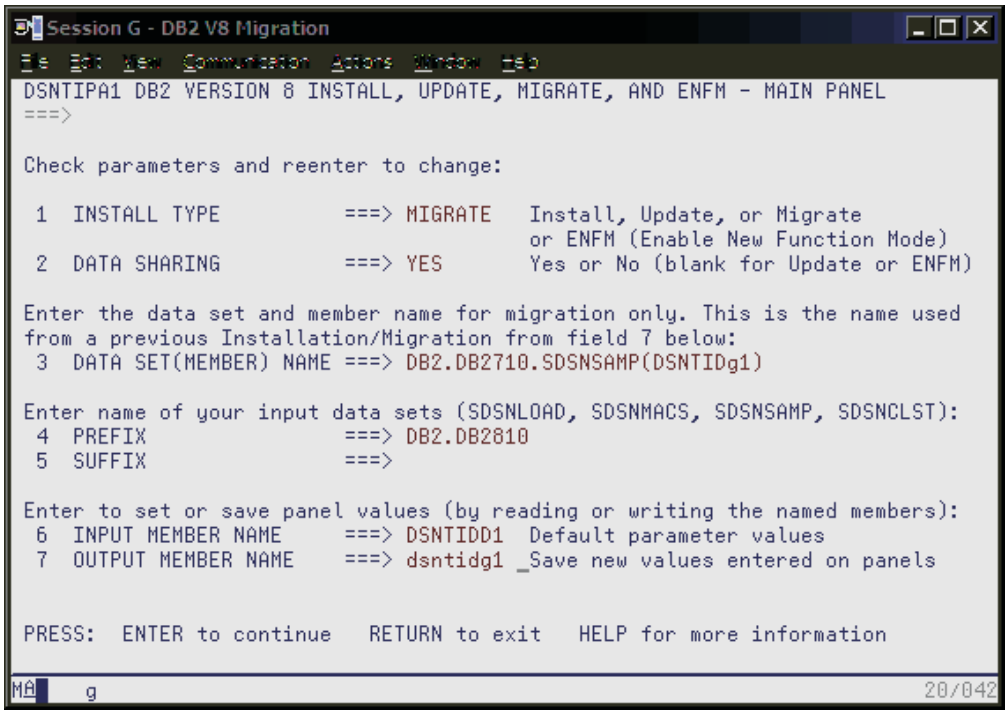


Figure 14. Executing DSNTINST in preparation for migrating the next member of the data sharing group

Pressing enter, we obtained the following pop-up screen:

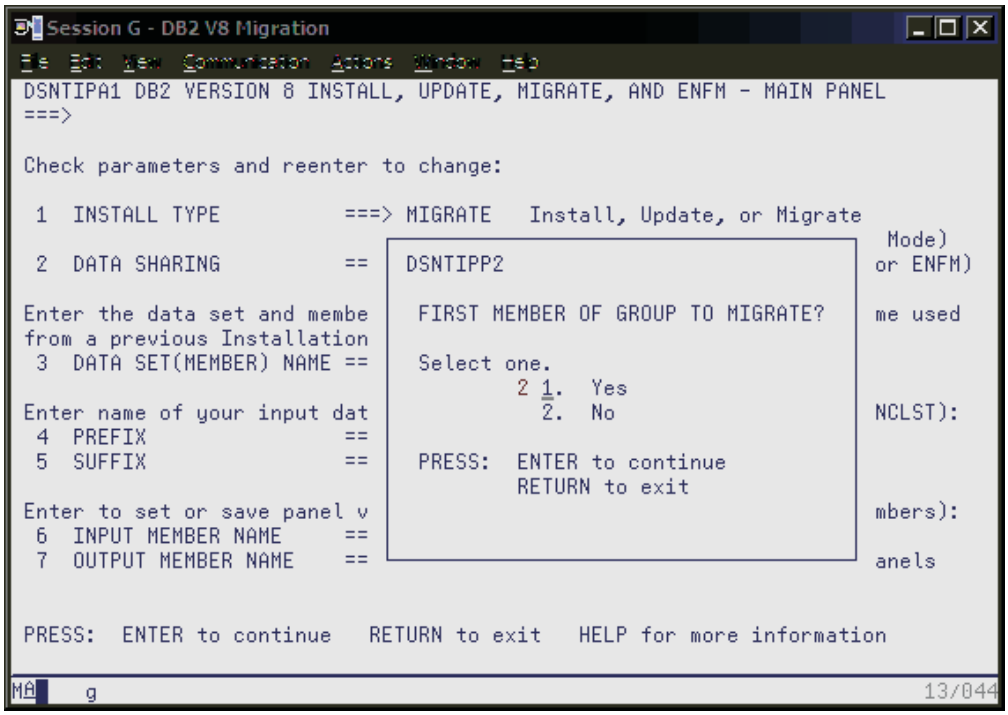


Figure 15. DSNTIPP2 pop-up screen

From this point, we scrolled the panels and accepted the existing values with the exception of the name of the sample library on panel DSNTIPT. We maintain a

separate sample library for each member of the data sharing group, so this field was updated accordingly to reflect DBG1, as shown in Figure 16.

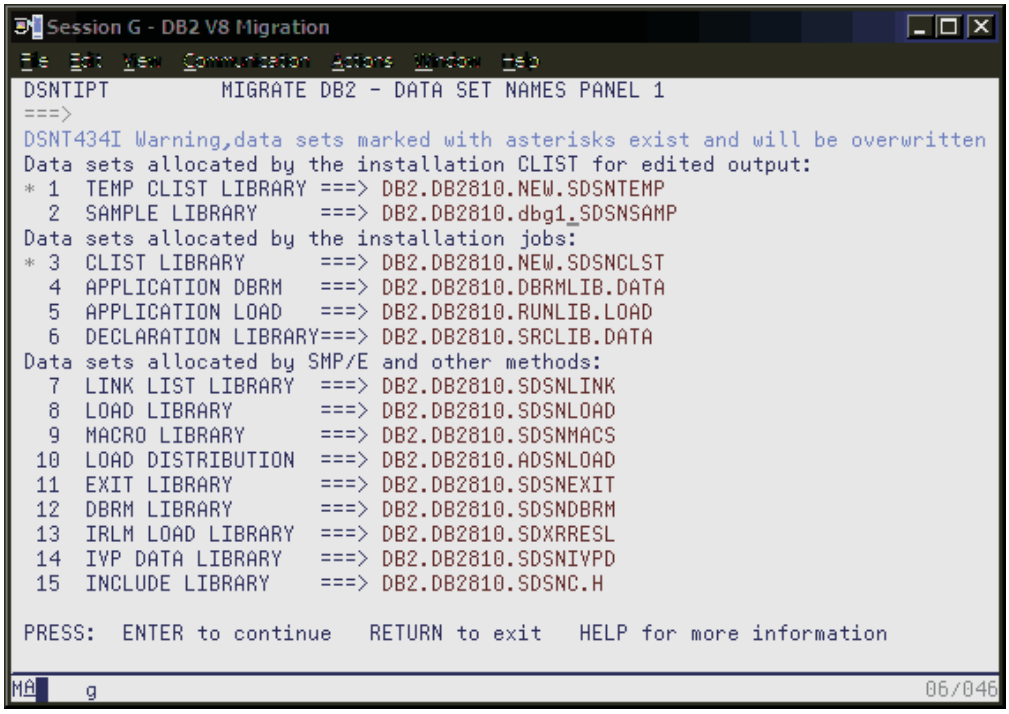


Figure 16. DSNTIPT - Data Set Names Panel 1

We placed the tailored migration JCL in DB2.DB2810.DBG1.SDSNSAMP as can be seen in Figure 17 on page 84:

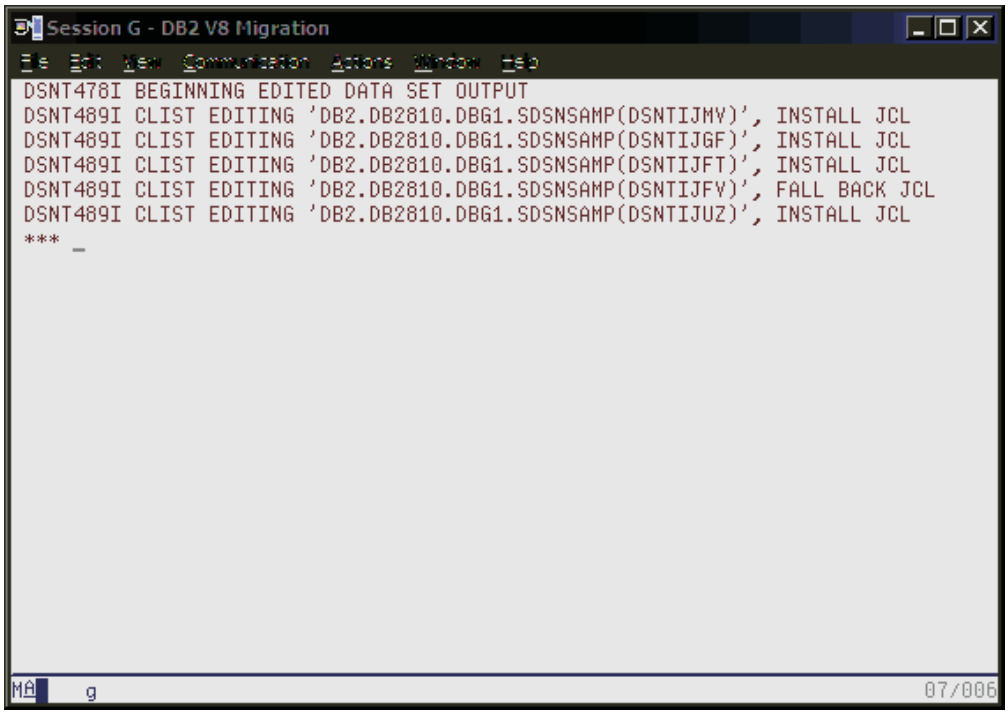


Figure 17. Tailored migration JCL placed in DB2.DB2810.DBG1.SDSNSAMP

Next, we removed the SMP/E step, brought DBG1 down, and ran DSNTIJUZ. DSNTIJUZ was submitted and ran successfully.

Next, we used steps RENAME and DSNTIPM of job DSNTIJMV to rename the existing Version 7 startup procedures for DBG1 and to add the new Version 8 startup procedures to proclib.

We then started DBG1 successfully in compatibility mode, as can be seen in Figure 18 on page 85:

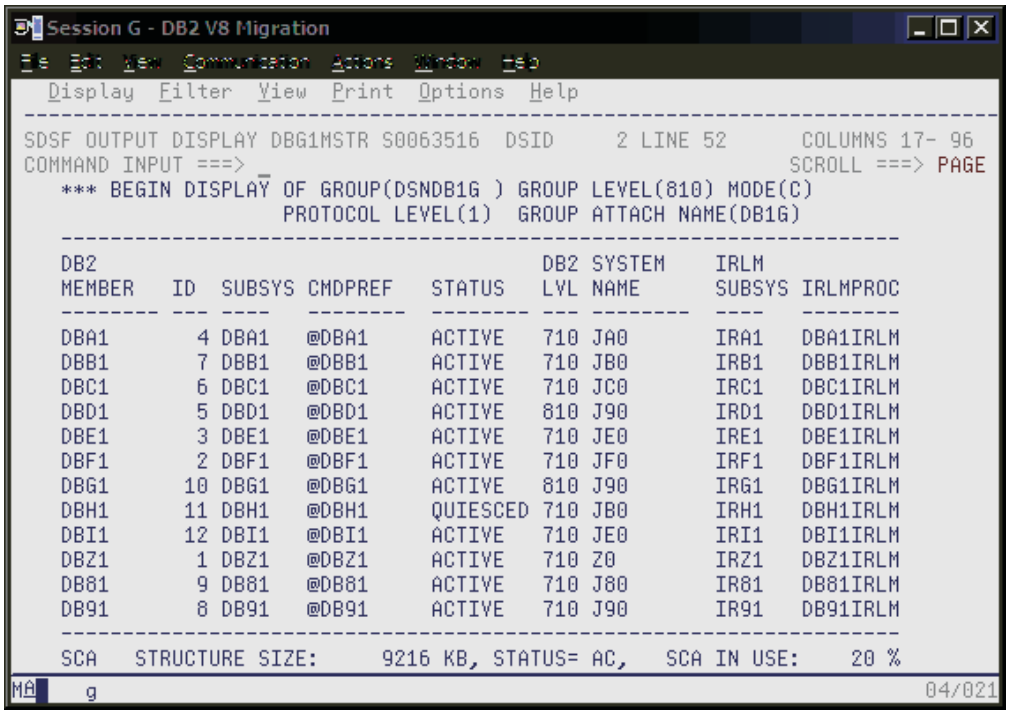


Figure 18. DBG1 started in compatibility mode

We followed the same process for the remaining ten members of the data sharing group, resulting in all members being in compatibility mode:

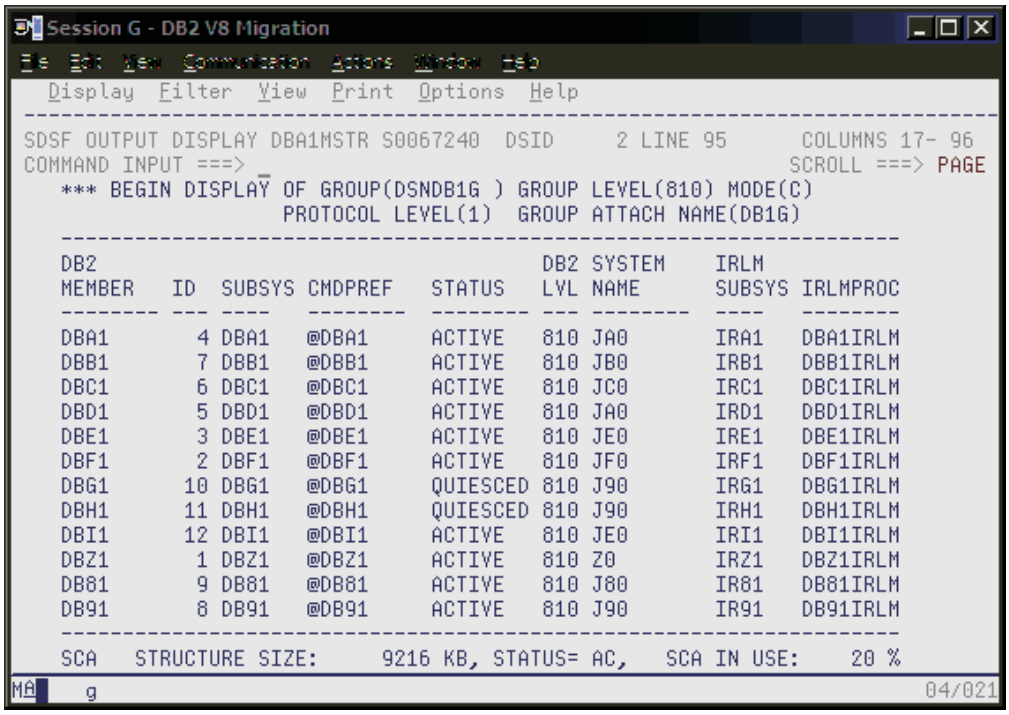


Figure 19. All members now in compatibility mode

Migrating to new function mode

After we migrated all members of the data sharing group to compatibility mode, we had to convert the DB2 catalog to exploit the new functions introduced by DB2 Version 8. The process is outlined below.

Preparing for new function mode

Before enabling-new-function mode, ensure that the following steps are taken.

- Ensure buffer pools BP8K0, BP16K0, BP32K exist, and in a data sharing environment that their corresponding group buffer pools have been defined (GBP8K0, GBP16K0, and GBP32K).
- An image copy of the catalog and directory must be taken before executing DSNTIJNE, which converts the DB2 catalog to Unicode for the first time.
- Increase the size of shadow data sets (panel DSNTIP01 of the installation CLIST) in order to support longer object names in the DB2 catalog; depending on their current size, you might have to increase the size of the catalog table space and index space as well.
- Run the installation CLIST using the ENFM option on panel DSNTIPA1.

After attending to the first three items, the installation CLIST was executed; panel DSNTIPA1 was completed as shown in Figure 20:

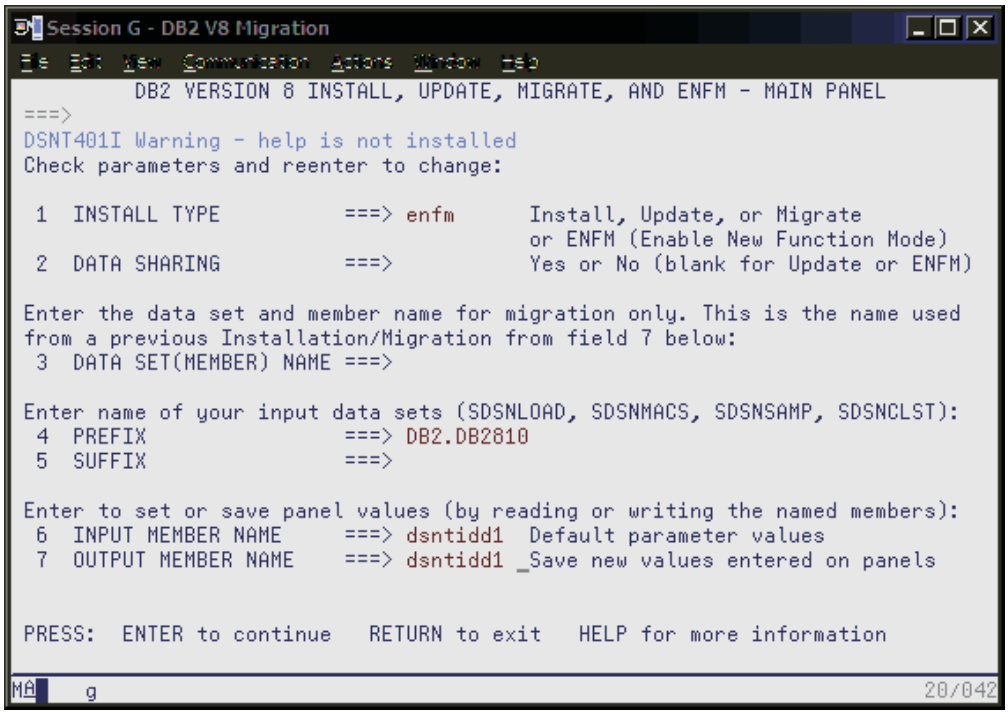


Figure 20. Executing DSNTINST in preparation for enabling-new-function-mode

Press enter twice to display panel DSNTIP00:

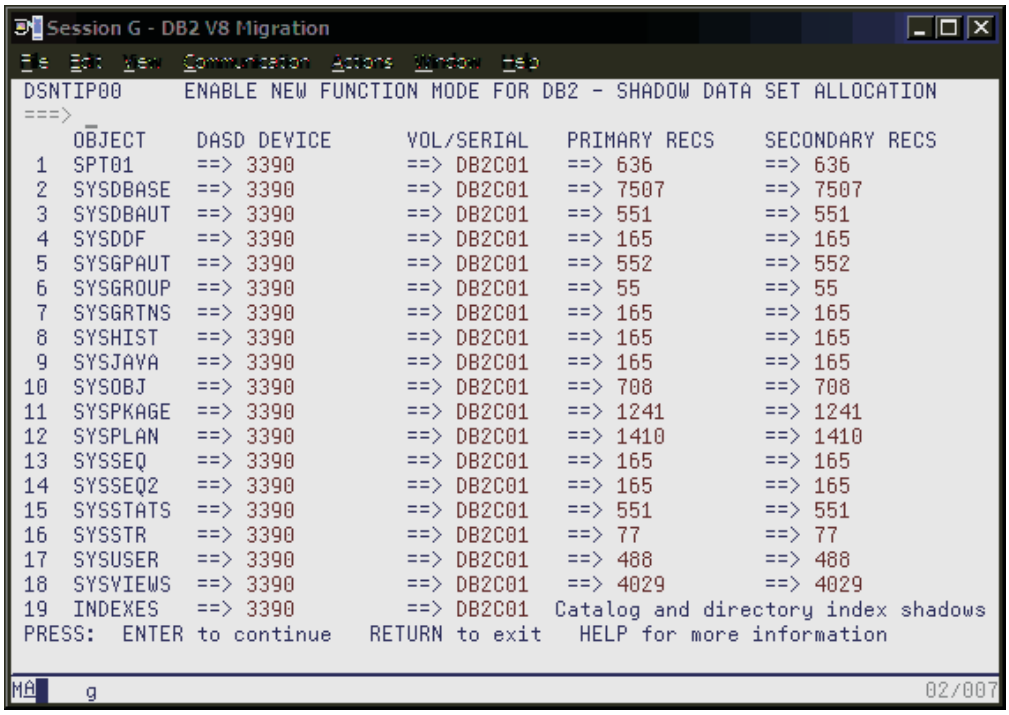


Figure 21. DSNTIP00 panel

We accepted calculated values and pressed enter to continue:

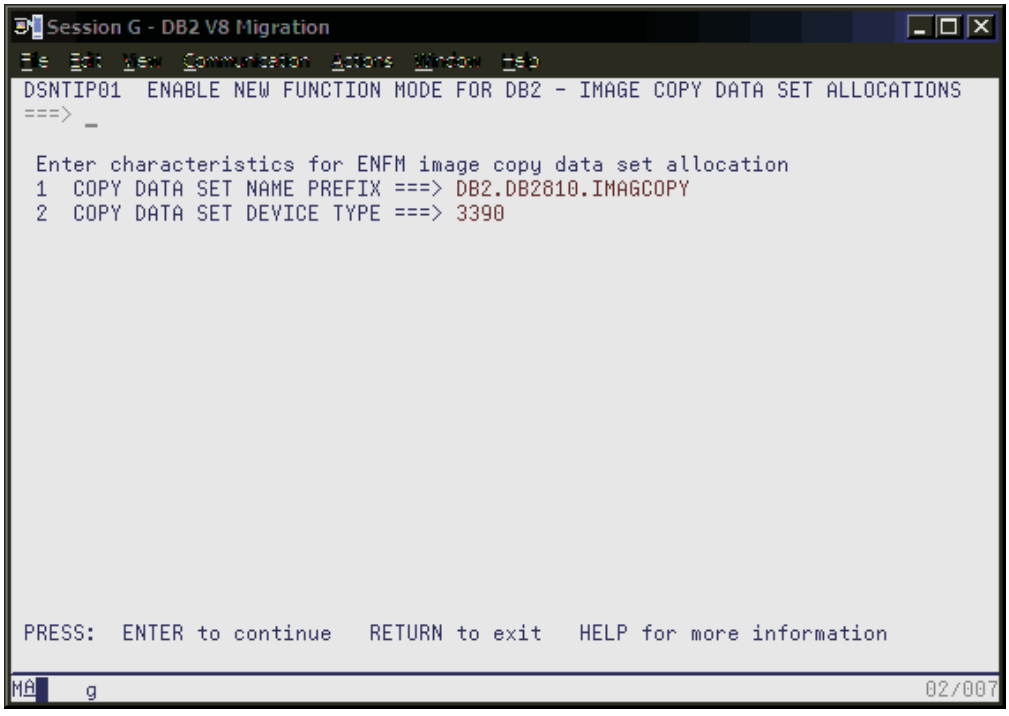


Figure 22. Image copy data set allocations on panel DSNTIP01

After updating the high-level-qualifier to be used for image copy data sets (`DB2.IC.DB1G`), we pressed enter and the following warning message appeared:

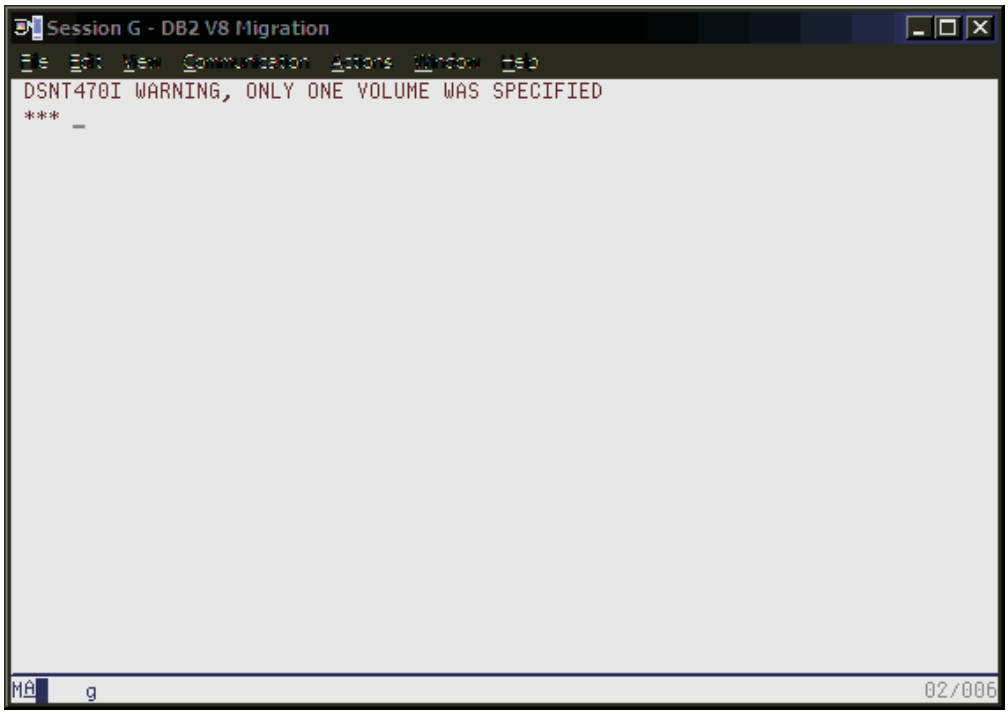


Figure 23. DSNT470I Warning message, only one volume was specified

After pressing enter, we obtained panel DSNTIP02:

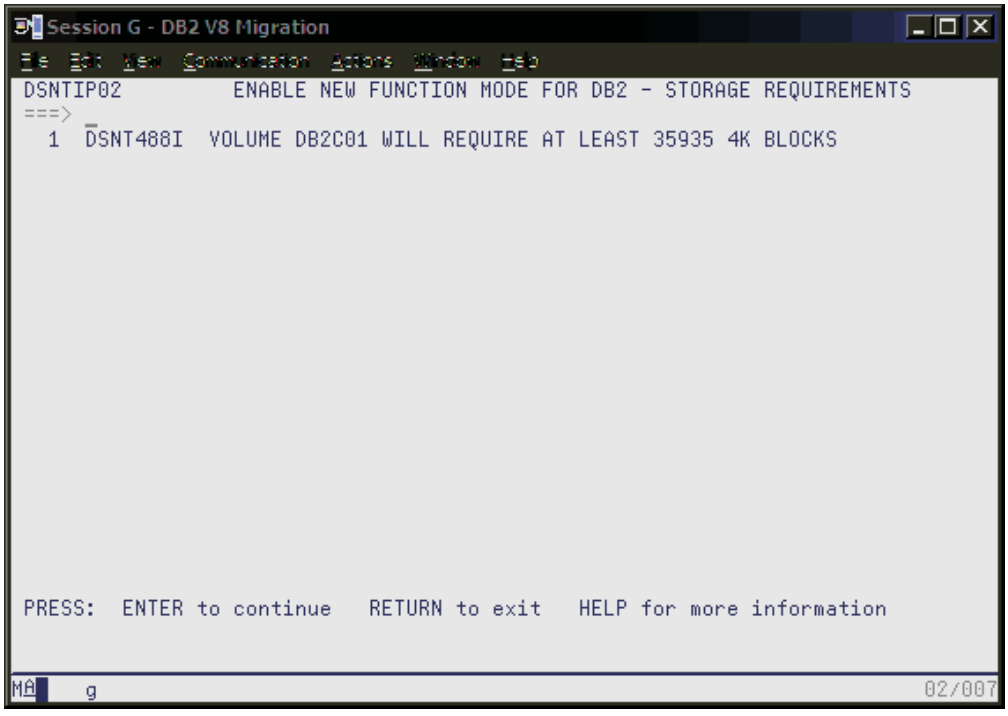


Figure 24. Message DSNT488I displayed on panel DSNTIP02

This was the last panel displayed. When we pressed enter, the generation of the enabling-new-function mode job along with the DB2 Version 8 sample jobs, occurred as shown in the following three screen images:

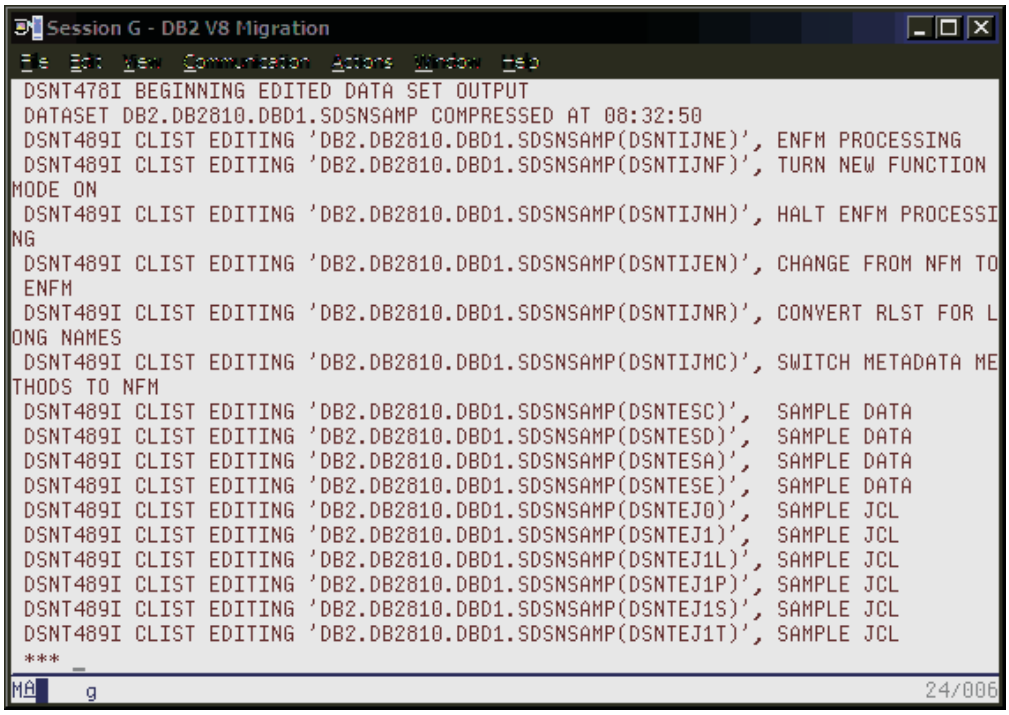


Figure 25. DSNT478I beginning data set output

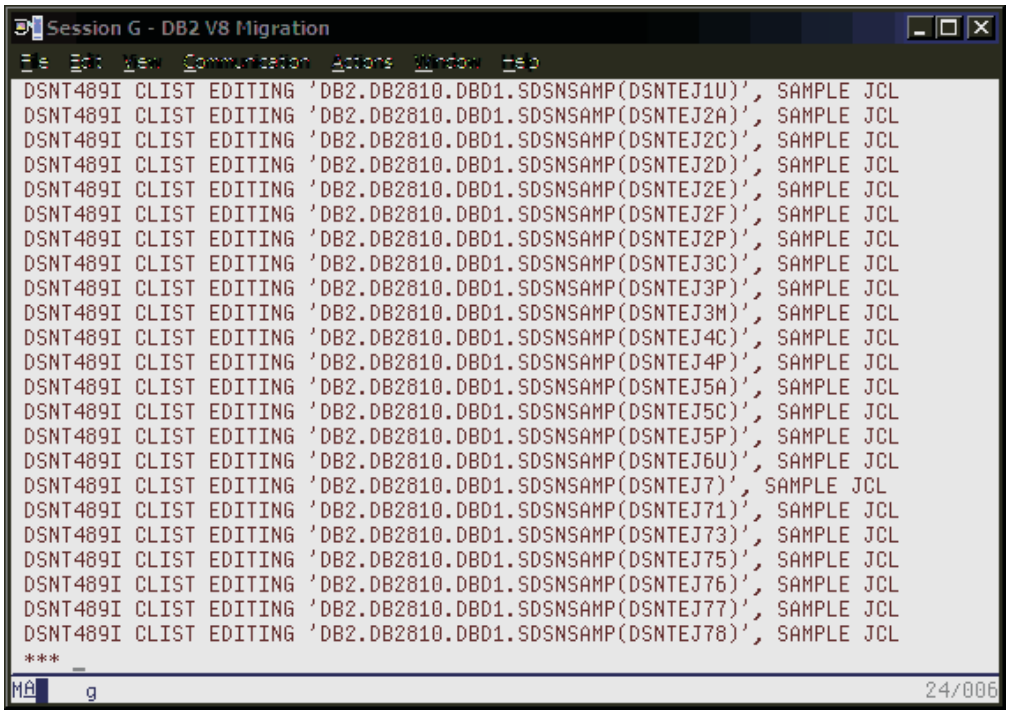


Figure 26. DSNT489I CLIST editing

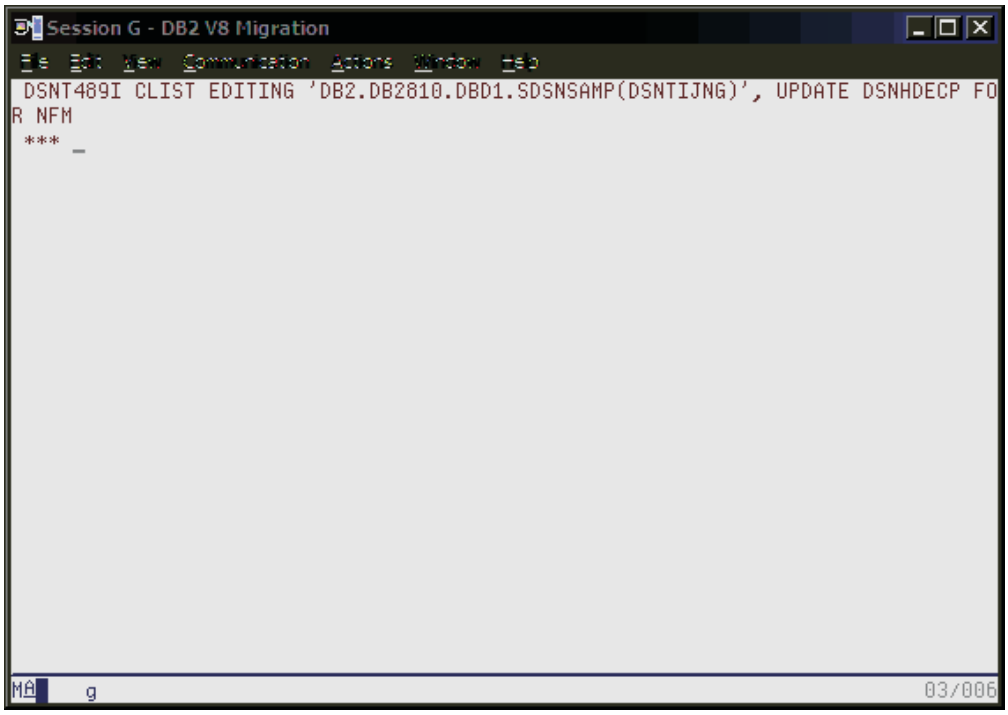


Figure 27. Screen showing completion of the preparation before enabling Version 8 new function mode

Enabling new function mode

Attention: Before proceeding, ensure that an image copy of the catalog and directory is taken.

The first step in enabling new function mode is to execute DSNTIJNE, which among other things converts the DB2 catalog to Unicode. The DISPLAY GROUP DETAIL command can be used during DSNTIJNE processing to determine how the enabling-new-function mode process is proceeding. It will display the names of the DB2 system table spaces and whether or not new function mode has been enabled yet, as shown in Table 13 on page 91:

Table 13. DB2 system table spaces and whether or not new function mode has been enabled yet.

Table Space	Enabled New Function Mode
SYSVIEWS	YES
SYSDBASE	YES
SYSDBAUT	YES
SYSDDF	NO
SYSGPAUT	NO
SYSGROUP	NO
SYSGRTNS	NO
SYSHIST	NO
SYSJAVA	NO
SYSOBJ	NO
SYSPKAGE	NO
SYSPLAN	NO
SYSSEQ	NO
SYSSEQ2	NO
SYSSTATS	NO
SYSSTR	NO
SYSUSER	NO
SPT01	NO

We submitted DSNTIJNE and then issued the DISPLAY GROUP DETAIL command throughout DSNTIJNE processing to view our progress. Note that the DISPLAY GROUP DETAIL output showed that we were now in enabling new function mode, as evidenced by `MODE(E)`.

Upon successful completion of DSNTIJNE, we took another image copy of the catalog and directory.

Next, we executed job DSNTIJNF, which is used to verify that the conversion of the DB2 catalog and directory. It completed successfully and produced a return code of zero. A DISPLAY GROUP DETAIL at this point reveals that we are now in new function mode (note `MODE(N)` in Figure 28 on page 92):

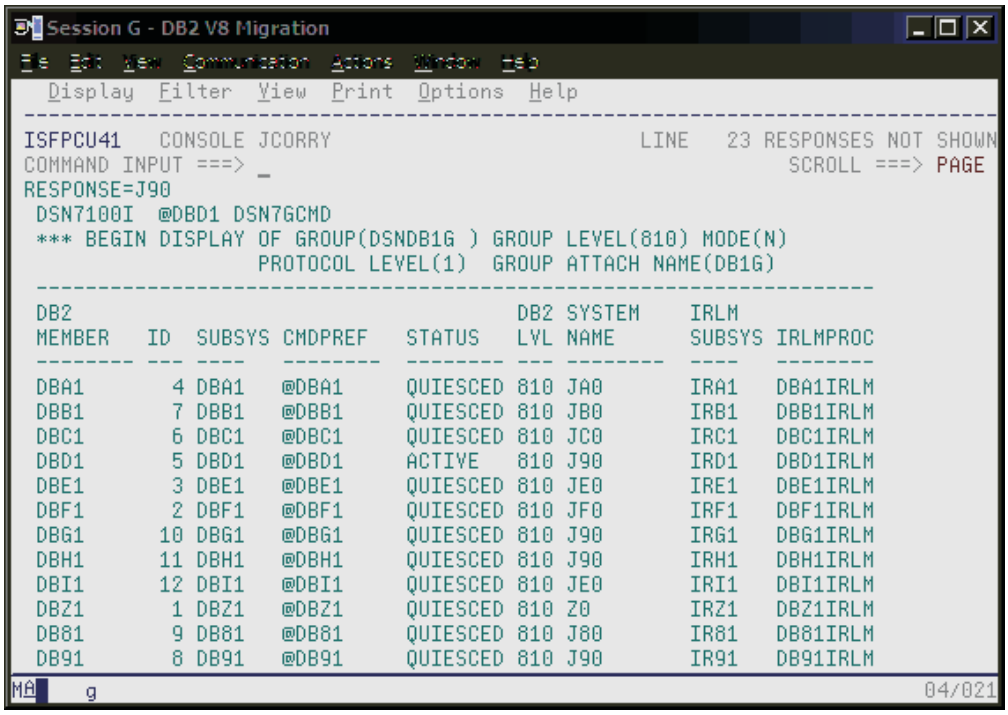


Figure 28. The DISPLAY GROUP command shows the data sharing group is now in new function mode

Running in new function mode

When we were in new function mode, we needed to run DSNTIJNG; it modifies DSNHDECP and allows new-function SQL statements introduced with Version 8 to be accepted by the DB2 precompiler by default. Note that in a data sharing environment where multiple DSNHDECP modules are in use, the jobs used to maintain the DSNHDECP modules must be updated to specify NEWFUN=YES. In our environment, only a single DSNHDECP module exists, therefore we ran DSNTIJNG successfully. (Note that we did not execute the last step, which deals with SMP/E.)

During the process of enabling-new-function mode, DATA CAPTURE was set to NONE on all of the DB2 catalog tables with the exception of SYSCOPY. Now that we are in new function mode, DATA CAPTURE must be re-enabled. This is a manual process and is performed by issuing the following ALTER command:

```
ALTER TABLE SYSIBM.SYSTABLES DATA CAPTURE CHANGES;
```

Also during enabling-new-function mode processing, two DB2 catalog tables that are not required by Version 8 were deleted. Once in new function mode, the corresponding VSAM data set for the index DSNKCX01 can be manually deleted.

An optional step can be executed once in new function mode that can be performed to convert `SYSIBM.DSNRLST01`, (the Resource Limit Specification Table, or RLST), to provide long name support for the authorization id (AUTHID), collection id (RLFCOLLN), and package id (RLFPKG) columns. Note that this step is only required if you would like to convert the Version 7 RLST or some other RLST (by

modifying the job) to support long names. We decided to enable long name support for the aforementioned columns of RLST and executed DSNTIJNR successfully.

Note that it is possible to create a larger BSDS once in new function mode, thereby providing support for up to 93 active log data sets and 10,000 archive log data sets per copy. Our BSDS was of a sufficient size for the time being, so we did not run the DSNJCNVB conversion utility as outlined in the *DB2 Installation Guide*.

Finally, once in new function mode, it is recommended to alter any frequently accessed buffer pools so that their pages are fixed in real storage, thereby avoiding the overhead involved for DB2 to fix and free pages each time an I/O operation is performed. For I/O intensive workloads, this processing time can amount to as much as 10%. To fix pages in storage, the PGFIX parameter of the ALTER BPOOL command is used as shown below:

```
ALTER BPOOL(buffer_pool_name) VPSIZE(virtual_page_size) PGFIX(YES)
```

Note that you should verify that sufficient real storage is available for fixing buffer pool pages before issuing the ALTER BPOOL command.

The following DSNDB06 indexes were placed in informational image copy status:

```
DSNDDX02
DSNDPX01
DSNDRX01
DSNDSX01
DSNDTX01
DSNDXX03
DSNPPH01
```

We image copied these indexes.

We also placed DSNRLST objects DSNRLS01 and DSNARL01 in advisory reorg status. Therefore, we reorganized table space DSNRLST.DSNRLS01.

Verifying the installation using the sample applications

Using the sample applications provided in DB2.DB2810.DBD1.SDSNSAMP, we performed verification of DBD1's migration to DB2 Version 8 as outlined below. Note that of the seven verification phases available, we ran only those phases and their associated jobs that applied to our specific environment.

Phase 0 is comprised of a single job, DSNTEJ0, that is used to free all objects that were created by running any of the seven verification phases. This permits the verification phases to be executed again in their entirety without the possibility of failure as a result of objects having been previously created.

Phases 1 through **3** are used to test the TSO and batch environments, including user-defined functions.

Phase 4 addresses IMS.

Phase 5 addresses CICS.

Phase 6 initializes sample tables and stored procedures for distributed processing.

Finally, **Phase 7** is used for the testing of DB2's Large Object feature (LOB) using sample tables, data, and programs.

We added the following JCLLIB statement after the JOB statement for all verification jobs that were executed:

```
//        JCLLIB ORDER=DB2.DB2810.PROCLIB
```

Recall that in **Migration Step 13** job DSNTIJMV was executed to add catalogued procedures to proclib; however, rather than directing the output of this step to SYS1.PROCLIB, it was directed to the newly created data set DB2.DB2810.PROCLIB. This library must be APF authorized (we dynamically added it to the APF authorization list before proceeding).

Beginning with **Phase 1**, which is used to create and load sample tables, we ran job DSNTEJ1. It produced the return codes as shown in Table 62 in the *DB2 Installation Guide*.

We prepared DSNTEP2 for DB2 Version 8 using job DSNTEJ1L, which produced a return code of zero. Job DSNTEJ1P was not executed as customization of DSNTEP2 was not required.

Job DSNTEJ1U installs DB2 Unicode support, but should only be run if the operating system is capable of supporting Version 7 Unicode. Referring to the program directory, Unicode support requires OS/390 Version 2 Release 9 or higher. Since our system is on z/OS Version 1 Release 6, we were able to execute DSNTEJ1U successfully.

Phase 2 tests the batch environment. Job DSNTEJ2A was executed to prepare assembler program DSNTIAUL. All steps produced the expected return codes.

Problems encountered:

When we ran job DSNTEJ2C next we ran into some problems. When we first ran this job, we received the following error:

```
IGYOS4003-E   INVALID OPTION PGMNAME(LONGUPPER) WAS FOUND AND DISCARDED
```

To correct this, we modified the parameters on the EXEC statement of steps PH02CS02 and PH02CS03 in job DSNTEJ2C. Originally, they were:

```
//PH02CS02 EXEC DSNHICOB,MEM=DSN8MCG,
//         COND=(4,LT),
//         PARM.PC=('HOST(IBMCOB)',APOST,APOSTSQL,SOURCE,
//         NOXREF,'SQL(DB2)','DEC(31)'),
//         PARM.COB=(NOSEQUENCE,QUOTE,RENT,'PGMNAME(LONGUPPER)'),
//         PARM.LKED='LIST,XREF,MAP,RENT'

.
.
.

//PH02CS03 EXEC DSNHICOB,MEM=DSN8BC3,
//         COND=(4,LT),
//         PARM.PC=('HOST(IBMCOB)',APOST,APOSTSQL,SOURCE,
//         NOXREF,'SQL(DB2)','DEC(31)'),
//         PARM.COB=(NOSEQUENCE,QUOTE,RENT,'PGMNAME(LONGUPPER)')
```

We updated them as shown below:

```
//PH02CS02 EXEC DSNHICOB,MEM=DSN8MCG,
//         COND=(4,LT),
//         PARM.PC=('HOST(IBMCOB)',APOST,APOSTSQL,SOURCE,
//         NOXREF,'SQL(DB2)','DEC(31)'),
//         PARM.COB=(NOSEQUENCE,APOST,RENT),
//         PARM.LKED='LIST,XREF,MAP,RENT'

.
.
.

//PH02CS03 EXEC DSNHICOB,MEM=DSN8BC3,
//         COND=(4,LT),
//         PARM.PC=('HOST(IBMCOB)',APOST,APOSTSQL,SOURCE,
//         NOXREF,'SQL(DB2)','DEC(31)'),
//         PARM.COB=(NOSEQUENCE,APOST,RENT)
```

Our modifications are highlighted above. Note that parameter PGMNAME(LONGUPPER) was removed altogether. After making these modifications, the job ran successfully and generated the expected return codes.

Because we do not use C, C++, or Fortran to access DB2, installation verification programs DSNTEJ2D, DSNTEJ2E and DSNTEJ2F were not executed.

We ran job DSNTEJ2P next to test PL/I program preparation procedures. It ran successfully.

Because we do not have access to C++ and did not complete the fields pertaining to C++ on the installation panel DSNTIPU, job DSNTEJ2U was not generated by the installation CLIST. This completed **Phase 2** of the installation verification procedures.

Phase 3 tests SPUFI, DRDA® access, dynamic SQL, and TSO connections to DB2 Version 8.

To test SPUFI, we used members DSNTESA, DSNTESC and DSNTESE of data set DB2.DB2810.DBD1.SDSNSAMP as input to SPUFI. We ran all these members successfully with the exception of DSNTESC, which failed on each of the insert statements . The DB2 Version 7 tables DSN8710.PLAN_TABLE, DSN8710.DSN_FUNCTION_TABLE, and DSN8710.DSN_STATEMNT_TABLE did not exist, causing the corresponding subselect to fail. When we commented out the insert statements, DSNTESC completed successfully.

We skipped running SPUFI at remote non-DB2 systems as none were available, and moved on to installation of the ISPF/CAF sample application. We removed the DSNHICOB parameter PGMNAME(LONGUPPER), and changed the QUOTE parameter to APOST before executing DSNTEJ3C. Next, we ran DSNTEJ3C successfully and generated the expected return codes. Finally, we ran DSNTEJ3P successfully and produced the designated return codes.

Chapter 7. Migrating to IMS Version 9

We migrated our IMS systems from IMS Version 8 Release 1 to IMS Version 9 Release 1. We completed the migration of our IMS systems in the following stages:

1. We migrated one of our IMS systems from IMS V8 to V9 on an IBM @server zSeries 990 server.
2. We migrated another IMS system to V9 on an IBM @server zSeries 800 server.
3. We migrated another IMS system to V9 on an IBM @serverzSeries 900 server
4. We migrated all remaining IMSs to Version 9 Release 1, except one, which needs to remain at Version 8 Release 1 for testing that is still in progress.

As soon as our IMS Version 8 Release 1 testing is completed, we will migrate the final IMS to Version 9 Release 1. We have been running with a mixed release IMSplex for approximately five months.

We used information from the following IMS V9.1 publications to perform the migration:

- IMS Version 9: Release Planning Guide, GC17-7831
- IMS Version 9: Installation Volume 1: Installation Verification , GC18-7822
- IMS Version 9: Installation Volume 2: System Definition and Tailoring, GC18-7823

We successfully performed the migration according to the instructions in the product publications. The following sections describe some of our experiences and observations.

Installing availability enhancements: IMS Version 9 provides two major enhancements for availability.

1. **z/OS Resource Manager Services:** IMS dynamically installs its Resource Manager cleanup routine; you do not need to install the DFSMRCL0 module as part of the IMS installation. Registration of the IMS Resource Manager cleanup routine with the operating system is done automatically during IMS startup.

   The Resource Manager is registered dynamically only for IMS Version 9 or later. If you use earlier IMS releases, or use both IMS Version 9 and earlier IMS releases, you must still install the DFSMRCL0 module as part of the IMS installation. The DFSMRCL0 module must be the highest version prior to IMS Version 9. When all IMSs (control region and batch regions) are IMS Version 9 or later, you can remove DFSMRCL0 from SYS1.LPALIB and remove the IEAVTRML CSECT of z/OS module IGC0001C. You must remove the name DFSMRCL0 from the IEAVTRML CSECT before you remove the module from SYS1.LPALIB. If the name is still in IEAVTRML but the module is not in SYS1.LPALIB, z/OS IPL will fail.

   Because we are running with both V8 and V9, we installed the V9 level of the DFSMRCL0 module. When we are running entirely on V9 we will uninstall the DFSMRCL0 module.
2. **DBRC Type-4 SVC Dynamic Install:** The Dynamic SVC (DFSUSVC0) utility dynamically updates the DBRC type-4 SVC module. Thus, you can apply maintenance to the DBRC type-4 SVC module without having to restart z/OS after each update. Before IMS Version 9, only the IMS type-2 SVC module could be dynamically updated. Any updates to the DBRC type-4 SVC required restarting z/OS.

You must define the IMS type-2 SVC and the DBRC type-4 SVC to z/OS before IMS starts. After you add the SVC definitions to IMS and restart z/OS, you can use the DFSUSVC0 utility to update the SVC routines without having to restart z/OS.

For more info see IMS Version 9: Release Planning Guide, GC17-7831

Using the same SVC numbers for different releases of IMS: IMS uses Type 2 supervisor calls (SVCs) in the range of 200 through 255 for batch, DBCTL, DCCTL, and DB/DC IMS control program functions, and a type 4 SVC in the same range for DBRC functions.

Note that, as in the past, the SVCs themselves are also downward compatible. For us, this means that during our migration we can use the V9 SVCs for both our IMS V9 and V8 systems.

Using the enhanced CHANGE.RECON command to upgrade the RECON data set without bringing systems down: Before migrating a system to IMS V9, we had to upgrade the RECON data set to the V9 level—you cannot migrate IMS until you do so. We used the enhanced CHANGE.RECON command, which ships as part of the V9 coexistence support SPE, to convert the RECON data set to the V9 level without shutting down active systems. This command must be run in batch mode because the RECON data set being upgraded to V9 still resides on a V8 system.

We performed the following steps to upgrade our RECON data set using the enhanced CHANGE.RECON command:

1. Installed the V9 coexistence support SPE PQ72840 and UQ82290.
2. Used the DBRC command utility, DSPURX00, to issue the CHANGE.RECON UPGRADE command. This command upgrades the RECON data set to the V9 level without shutting down all IMS activity. It also uses the DBRC I/O recovery algorithms to recover from failures during the upgrade.

   The following is the output from our DSPURX00 command utility job that issued the CHANGE.RECON UPGRADE command:

```
         IMS VERSION 9 RELEASE 1  DATA BASE RECOVERY CONTROL
  CHANGE.RECON UPGRADE
DSP0251I  RECON COPY 1 UPGRADE IS BEGINNING
DSP0252I  RECON COPY 1 UPGRADED SUCCESSFULLY
DSP0251I  RECON COPY 2 UPGRADE IS BEGINNING
DSP0252I  RECON COPY 2 UPGRADED SUCCESSFULLY
DSP0203I  COMMAND COMPLETED WITH CONDITION CODE 00
DSP0220I  COMMAND COMPLETION TIME 05.041 13:17:02.3
         IMS VERSION 9 RELEASE 1  DATA BASE RECOVERY CONTROL
DSP0211I  COMMAND PROCESSING COMPLETE
DSP0211I  HIGHEST CONDITION CODE = 00
```

Using a new JCL execution member for OLDS for coexistence: While we were running in coexistence mode with both IMS V9 and V8 systems, we needed two skeletal JCL execution members that kick off the online log data set (OLDS) archive—one for each release. The default skeletal JCL member for the log archive utility, ARCHJCL, points to the IMS V9 target library, RESLIB. For our V9 IMS systems, we created a new skeletal JCL execution member, ARCHJCL9, that points to the IMS V9 target library, SDFSRESL. To ensure that we invoke the new ARCHJCL9 execution member for our V9 systems, we pointed our VSPEC parameter to a new DFSVSM*xx* member of IMS.PROCLIB, which contains the following ARCHDEF statement:

```
ARCHDEF ALL MAXOLDS(1) MEMBER(ARCHJCL9)
```

Updating our change accumulation utilities, image copy utilities, and recovery jobs: We updated our change accumulation utilities, image copy utilities, and recovery jobs to point to the IMS V9 libraries. This is necessary for IMS V9 and V8 to coexist. After we completed the updates, all of these utilities and jobs ran successfully.

Updating IPCS and ISPF panels: We updated our IPCS and ISPF dialog panels to point to the IMS V9 libraries.

Upgrading the IMS V9 utilities: We needed to upgrade the following IMS V9 utilities:

- IMS Library Integrity Utilities for z/OS, V1.1 5655-I42 replaces our current IMS Library|Management Utilities 5655-E04.
- IMS Database Control Suite for z/OS, V3.1 5655-L08 replaces our current V2.2 level.
- IMS High Performance Pointer Checker for z/OS, V2.1 5655-K53 replaces our current V1.1 level.

Applying additional service for IMS V9: We did not need to apply any additional service to migrate to IMS V9.

Exploiting new functions in IMS V9: We have completed our migration to IMS V9 and we are currently evaluating ways to exploit new functions, such as High Availability Large Database (HALDB) Online Reorganization. As we implement new functions, we will report on our experiences with them in upcoming editions of our test report.

After our migration to IMS V9 is completed successfully, we will do the following:

- **Update the RECON MINVERS parm:** Use the `CHANGE.RECON` command to update options in the RECON status record to specify V9R1:

  ```
  change.recon minivers(91)
  ```

  For more information see IMS Database Recovery Control (DBRC) Guide and Reference (SC18-7818-00)
- **Uninstall DFSMRCL0:** Our final V9 migration step will be to uninstall module DFSMRCL0 from SYS1.LPALIB. See 1 on page 97 for more information.

Migrating to the integrated IMS Connect

IMS Connect was always a separately orderable product. IMS Version 9 provides an integrated IMS Connect function that offers a functional replacement for the IMS Connect tool (program number 5655-K52). The integrated IMS Connect is included in the IMS System Services function modification identifier (FMID), HMK9900; the integrated IMS Connector for Java for z/OS is included with the IMS Java FMID, JMK9906; and the Integrated IMS Connector for Java distributed can be downloaded from the IMS Web site:

```
www.ibm.com/software/data/ims/
```

For more information see IMS Version 9: Release Planning Guide, GC17-7831

During the IMS V9 migration we chose to migrate from IMS Connect V2.1 to IMS V9 integrated IMS Connect.

Migrating to IRLM Version 2 Release 2

During the IMS migration we also chose to migrate from IRLM Version 2 Release 1 to Version 2 Release 2. This was a simple migration that included the following steps:

- Updated PET.PROCLIB(IRLM) to remove unused parameters:

  ```
  PC=YES
  MAXCSA=12
  ```

- Updated PET.PROCLIB(IRLM) to add new IRLM 2.2 parameter:

  ```
  MEMLIMIT=2G
  ```

Although we chose to do the following migrations within a very short period of time, no major problems were discovered:

- IMS V8 to V9,
- IMS Connect 2.1 to the integrated IMS Connect
- IRLM 2.1 to 2.2

This was by far, the smoothest IMS migration we have performed.

Chapter 8. Implementing the IMS Common Service Layer and the Single Point of Control

The IMS Common Service Layer (CSL) is a collection of IMS manager address spaces that provide the necessary infrastructure for systems management tasks. The IMS CSL reduces the complexity of managing multiple IMS systems by providing you with a single-image perspective in an IMSplex. That is, you can now manage multiple IMS subsystems in an IMSplex as if they were one system.

An IMS single point of control (SPOC) is a program with which you can manage operations of all IMS systems within an IMSplex.

We used the following documentation to help us implement the CSL and SPOC in our production IMSplex:

- *IMS Version 8: Common Service Layer Guide and Reference*, SC27-1293
- *IMS Version 8: Common Queue Server Guide and Reference*, SC27-1292
- *IMS Version 8: Installation Volume 2: System Definition and Tailoring*, GC27-1298

Setting up the Common Service Layer

The CSL address spaces, or CSL managers, include the operations manager (OM), resource manager (RM), and structured call interface (SCI). The CSL managers perform the following functions:

- **Operations manager (OM):** Helps control the operations of all IMS systems in an IMSplex. The OM receives processing control when an OM request (an IMS command, for example) is received by the OM application programming interface (API). All commands and responses to those commands must come through the OM API.
- **Resource manager (RM):** Helps manage resources that are shared by multiple IMS systems in an IMSplex. The RM provides the infrastructure for managing global resource information and coordinating IMSplex-wide processes.
- **Structured call interface (SCI):** Allows IMSplex members to communicate with one another. Communication between IMSplex members can occur within a single z/OS image or among multiple images. The individual IMSplex members do not need to know where the other members reside or what communication interface to use.

See Figure 29 on page 105 for a depiction of these address spaces.

Steps for setting up the CSL

We performed the following steps to set up the CSL on our z/OS production systems:

1. Added PGMNAME(BPEINI00) to the PPT (SCHED00):

```
PPT PGMNAME(BPEINI00)       /* PROGRAM NAME = BPEINI00              */
                 CANCEL     /* PROGRAM CAN BE CANCELED              */
                 KEY(7)     /* PROTECT KEY ASSIGNED IS 7            */
                 NOSWAP     /* PROGRAM IS NON-SWAPPABLE             */
                 NOPRIV     /* PROGRAM IS NOT PRIVILEGED            */
                 DSI        /* REQUIRES DATA SET INTEGRITY          */
                 PASS       /* CANNOT BYPASS PASSWORD PROTECTION    */
                 SYST       /* PROGRAM IS A SYSTEM TASK             */
                 AFF(NONE)  /* NO CPU AFFINITY                      */
                 NOPREF     /* NO PREFERRED STORAGE FRAMES          */
```

Note: BPEINI00 can also be used to start CQS, so the CQSINIT0 entry is no longer needed. Once we migrated all of our production systems to IMS V8.1, we removed the entry for CQSINIT0.

---

2. Added the IMSPLEX parameter to the CQSIP*xxx* member on each system:

```
CQSGROUP=SQGRP,
IMSPLEX(NAME=PROD),
SSN=CQSA,
STRDEFG=ALL,
STRDEFL=PEA
```

---

3. Added the RSRCSTRUCTURE parameter to the CQSSGALL member:

```
STRUCTURE(
 STRNAME=FFMSGQ_STR,
 OVFLWSTR=FFOVFLO_STR,
 STRMIN=0,
 SRDSDSN1=CQS.FF.SRDS1,
 SRDSDSN2=CQS.FF.SRDS2,
 LOGNAME=CQS.FF.LOGSTRM,
 OBJAVGSZ=1024,
 OVFLWMAX=50
 )
STRUCTURE(
 STRNAME=FPMSGQ_STR,
 OVFLWSTR=FPOVFLO_STR,
 STRMIN=0,
 SRDSDSN1=CQS.FP.SRDS1,
 SRDSDSN2=CQS.FP.SRDS2,
 LOGNAME=CQS.FP.LOGSTRM,
 OBJAVGSZ=512,
 OVFLWMAX=50
)
RSRCSTRUCTURE(STRNAME=CSLRMGR_PROD)
```

Note: A resource structure is not needed if only one RM is used in the IMSplex. For availability reasons, we chose to start two RMs and defined a resource structure.

---

4. Created a DFSCG*xxx* member:

```
CMDSEC=N,
IMSPLEX=PROD,
LEOPT=N,
NORSCCC=(),
OLC=LOCAL
```

---

5. Added the CSLG parameter to the DFSPB*xxx* member on each system:

```
DLINM=DLIGRP81,DBRCNM=DBRCGP81,
AUTO=N,
GRSNAME=IMSPETGR,
SUF=1,
CRC=#,LHTS=512,NHTS=512,UHTS=512,
CMDMCS=Y,
CSLG=PET,
APPC=N,AOIS=S,
FIX=HP,VSPEC=81,PRLD=DB,SPM=02,
RSRMBR=,GRNAME=NATIVE2,
...
```

---

6. Created the initialization proclib members for the CSL manager address spaces:

   Example: The following is the SCI initialization member:

```
*-------------------------------------------------------------------*
* Sample SCI Initialization Proclib Member.                         *
*-------------------------------------------------------------------*

ARMRST=N,                   /* ARM should restart OM on failure  */
SCINAME=SCI1,               /* SCI Name (SCIID = SCI1SCI)        */
IMSPLEX(NAME=PROD)          /* IMSplex Name                      */
```

   Example: The following is the OM initialization member:

```
*-------------------------------------------------------------------*
* Sample OM Initialization Proclib Member.                          *
*-------------------------------------------------------------------*

ARMRST=N,                   /* ARM should restart OM on failure  */
CMDLANG=ENU,                /* Use English for Commmand Desc     */
CMDSEC=N,                   /* No Command Security               */
CMDTEXTDSN=IMS810.SDFSDATA, /*                                   */
OMNAME=OM1,                 /* OM Name (OMID = OM1OM)            */
IMSPLEX(NAME=PROD)          /* IMSplex Name (CSLPLEX1)           */
```

   Example: The following is the RM initialization member:

```
*-------------------------------------------------------------------*
* Sample RM Initialization Proclib Member.                          *
*-------------------------------------------------------------------*

ARMRST=N,                   /* ARM should restart RM on failure  */
CQSSSN=CQSC,
IMSPLEX(NAME=PROD,          /* IMSPLEX NAME                      */
RSRCSTRUCTURE(STRNAME=CSLRMGR_PROD)),
RMNAME=RMC                  /* RM Name (RMID = RM1RM)            */
```

---

7. Created the CSL startup procedures:

   Example: The following is our SCI startup procedure, CSLSCI:

```
//CSLSCI   PROC RGN=3000K,SOUT=A,
//             IMSVAR=&IMSVAR,
//             BPECFG=BPECFG00,
//             SCIINIT=000,
//             PARM1=(SCINAME=&SCINAME)
//*
//SCIPROC  EXEC PGM=BPEINI00,REGION=&RGN,
//  PARM='BPECFG=&BPECFG,BPEINIT=CSLSINI0,SCIINIT=&SCIINIT,&PARM1'
//*
//STEPLIB  DD  DISP=SHR,DSN=IMS810.&IMSVAR..SDFSRESL
//         DD  DSN=SYS1.CSSLIB,DISP=SHR
//PROCLIB  DD  DSN=D10.PETDSW4.PROCLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=&SOUT
//SYSUDUMP DD  SYSOUT=&SOUT
//*
```

   Example: The following is our OM startup procedure, CSLOM:

```
//CSLOM    PROC RGN=3000K,SOUT=A,
//             IMSVAR=&IMSVAR,
//             BPECFG=BPECFG00,
//             OMINIT=000,
//           PARM1=(OMNAME=&OMNAME)
//*
//OMPROC   EXEC PGM=BPEINI00,REGION=&RGN,
//  PARM='BPECFG=&BPECFG,BPEINIT=CSLOINI0,OMINIT=&OMINIT,&PARM1'
```

```
//*
//STEPLIB  DD  DSN=IMS810.&IMSVAR..SDFSRESL,DISP=SHR
//         DD  DSN=SYS1.CSSLIB,DISP=SHR
//PROCLIB  DD  DSN=D10.PETDSW4.PROCLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=&SOUT
//SYSUDUMP DD  SYSOUT=&SOUT
//*
```

Example: The following is our RM startup procedure, CSLRM:

```
//CSLRM    PROC RGN=0M,SOUT=A,
//             IMSVAR=&IMSVAR,
//             BPECFG=BPECFG00,
//             INIT=CSLRINI0,
//             RMINIT=&RMINIT
//*
//RMPROC   EXEC PGM=BPEINI00,REGION=&RGN,
//   PARM='BPEINIT=&INIT,BPECFG=&BPECFG,RMINIT=&RMINIT'
//STEPLIB  DD  DSN=IMS810.&IMSVAR..SDFSRESL,DISP=SHR
//         DD  DSN=SYS1.CSSLIB,DISP=SHR
//PROCLIB  DD  DSN=D10.PETDSW4.PROCLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=&SOUT
//SYSUDUMP DD  SYSOUT=&SOUT
//*
```

---

8. Updated the SCI registration exit routine (DSPSCIX0) and placed it into an authorized library:

```
PTBLEYEC DS    0H                    TABLE EYECATCHER
         DC    C'PLEXTABL'
PLEXTABL DS    0H
         DC    CL(DSNL)'RECON1.PROD'             RECON NAME
         DC    CL(PNL)'PROD '                    IMSplex name
         DC    XL(RCL)'00000000'           RC00 = use the IMSplex name
         DC    CL(DSNL)'RECON2.PROD'             RECON NAME
         DC    CL(PNL)'PROD '                    IMSplex name
         DC    XL(RCL)'00000000'           RC00 = use the IMSplex name
         DC    CL(DSNL)'RECON3.PROD'             RECON NAME
         DC    CL(PNL)'PROD '                    IMSplex name
         DC    XL(RCL)'00000000'           RC00 = use the IMSplex name
```

---

9. Defined the resource manager coupling facility structure in the CFRM policy:

```
STRUCTURE NAME(CSLRMGR_PROD)
          SIZE(32000)
          INITSIZE(20000)
          ALLOWAUTOALT(YES)
          FULLTHRESHOLD(60)
          DUPLEX(ALLOWED)
          PREFLIST(CF2,CF1,CF3)
```

Our CSL and SPOC configuration

Figure 29 on page 105 illustrates our the CSL and SPOC configuration in our IMSplex:

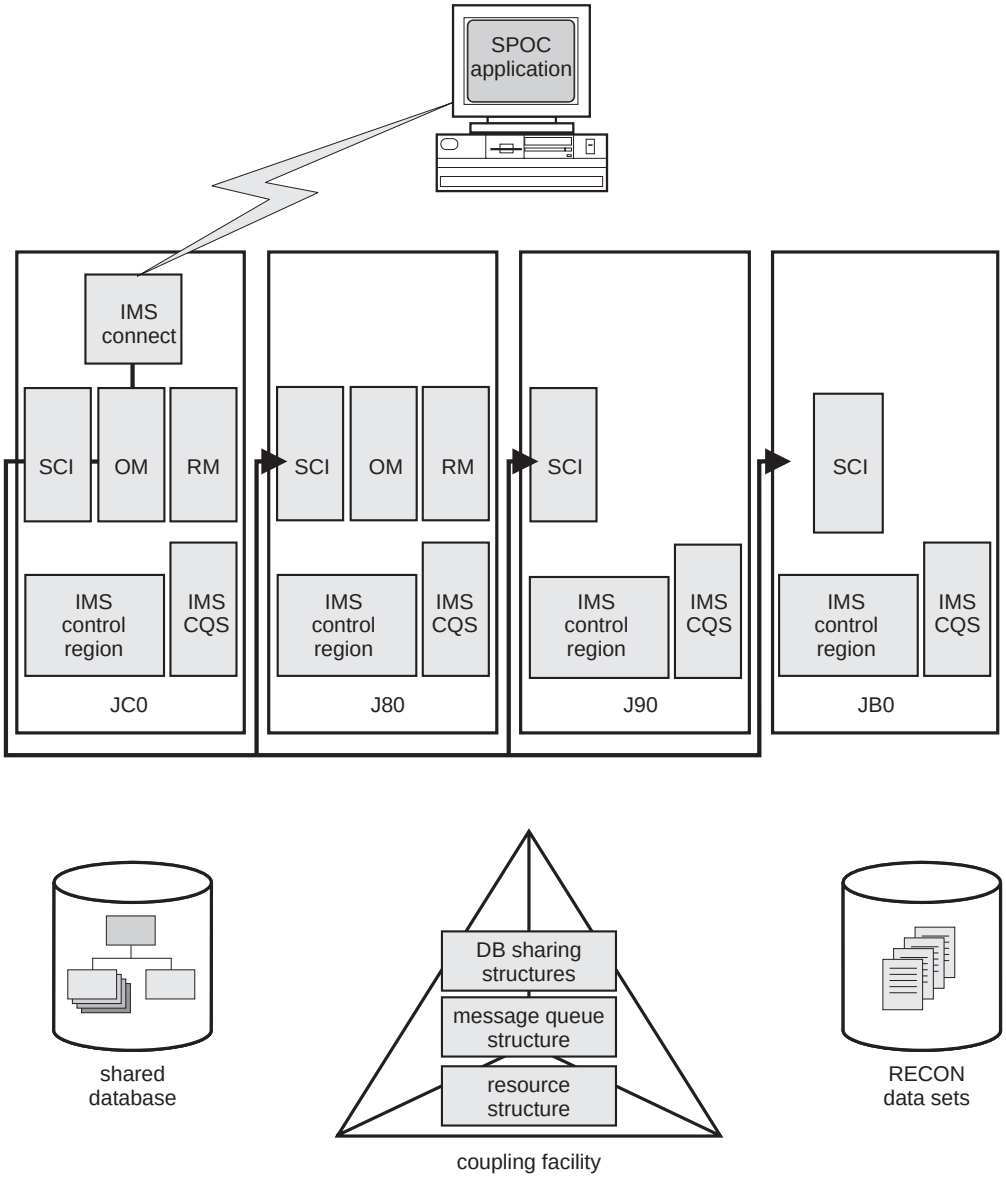


Figure 29. Our IMS CSL and SPOC configuration

Note the following about our configuration:

- We run one SCI address space on each z/OS system where IMS subsystems run.
- We only run OM and RM address spaces on systems J80 and JC0.
- We use automations to start the CSL address spaces following system IPLs. However, we found that if the RM doesn't detect the required CQS address space within 10 minutes, it terminates with a U0010-00000508 user abend. To avoid this, we added the CQS startup to automations, instead of allowing IMS to automatically start the CQS address space. This ensures that the CQS address space is available when the RM expects it.

IMS performance considerations for CSL

For the CLS address spaces, IBM recommends using the SYSSTC service class or a service class with higher importance (that is, a lower-numbered value) than the CNTL and CQS address spaces and all dependent regions. Since WLM provides

five levels of importance, a general guideline would be to group resources into service classes with the following relative importance:

Importance		
Level	**Value**	**Address spaces**
higher	N	IRLM
	N+1	VTAM, APPC, DBRC, SCI, OM, RM
	N+2	CNTL, CQS
	N+3	DLIS
lower	N+4	dependent regions

Thus, when CPU resources are constrained, the following rules would apply:

- All dependent regions should have the lowest dispatching priority among the other IMS address spaces.
- CNTL and CQS, the address spaces with the next largest CPU consumption, should have a lower dispatching priority than the CSL address spaces.

Setting up the single point of control

A SPOC communicates with one OM address space; the OM then communicates with all of the other IMS address spaces in the IMSplex, through the SCI, as required for operations.

Steps for setting up the single point of control

We performed the following steps to set up the single point of control:

1. Verified that IMS service PQ69527 (PTF UQ73719) is installed.

2. Verified that IMS Connect is installed.

   We are currently running IMS Connect V2.1. However, note that if you are running IMS Connect V1.2, you must install PQ62379 (PTF UQ69902) and PQ70216 (PTF UQ74285).

3. Updated the HWS configuration member to add the EXIT and IMSPLEX parameters:

```
HWS
 (ID=HWSC,RACF=N)
TCPIP
 (HOSTNAME=TCPIP,RACFID=RACFID,PORTID=(9999,9998,9997,9996,9995,9994,
 9993,9992,9991,9990),
 EXIT=(HWSCSL00,HWSCSL01),
 MAXSOC=51,TIMEOUT=9999)
DATASTORE
 (ID=IMSC,GROUP=NATIVE2,MEMBER=HWSC,TMEMBER=IMSPETJC)
IMSPLEX
 (MEMBER=IMSPLEXC,TMEMBER=PROD)
```

4. Installed DB2 UDB V8.1 (DB2 Control Center) on the workstation.

   We are currently running at the FixPak 4 service level. You can get the latest FixPaks from the DB2 Technical Support Web site at www.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/index.d2w/report.

Steps for setting up DB2 Control Center for the IMS SPOC

We performed the following steps to set up DB2 Control Center for the IMS SPOC:

1. Started the Control Center, then clicked **Selected → Add** from the menu bar.

---

2. In the **Add System** dialog box:
   a. Selected the **IMS** button
   b. In the **System name** box, typed the name of our IMSplex: PROD
   c. In the **Host name** box, typed the IP address of the system where the IMS Connect subsystem is located
   d. In the **Port number** box, typed the IMS Connect port number we wanted to use: 9995
   e. Clicked **OK**

   Example: Figure 30 is an example of the **Add System** dialog box:

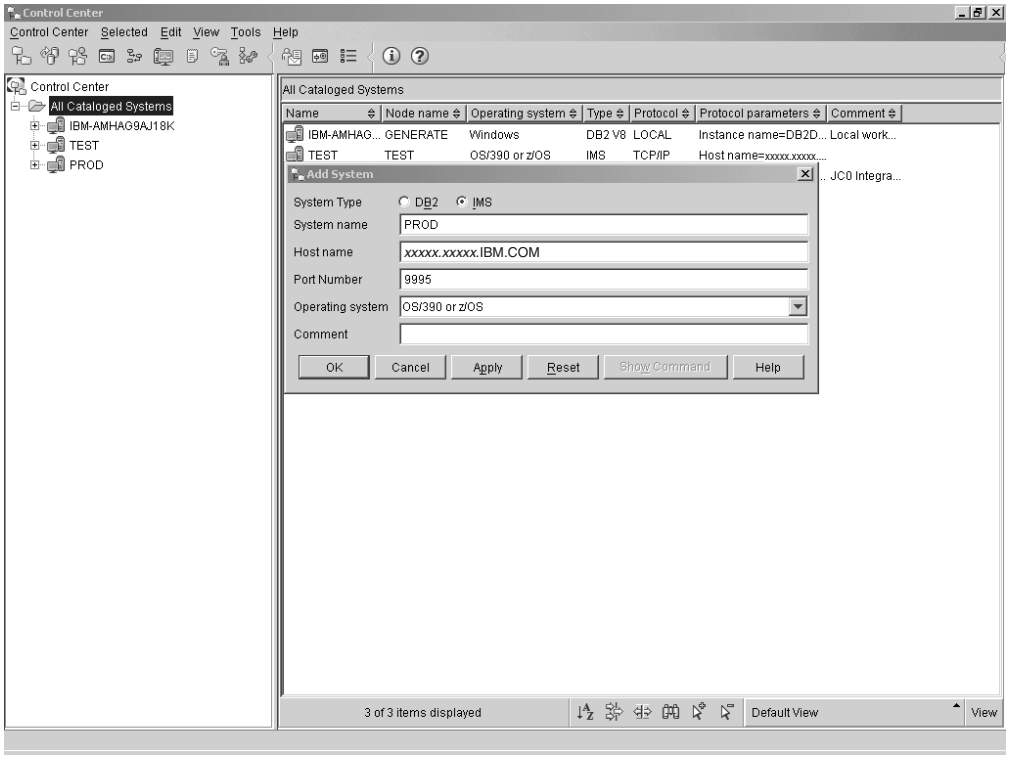


Figure 30. Example of the Control Center Add System dialog

---

3. Started the Command Center by clicking **Tools → Command Center** from the menu bar.

---

4. In the Command Center:
   a. In the **Command type** field, selected **IMS commands** from the pull-down menu
   b. In the **IMS sysplex** field, selected PROD (the name of our IMSplex) from the **Select Connection** dialog
   c. When prompted, entered the user ID and password that we had set during the installation of DB2 Control Center.

Example: Figure 31 is an example of the initial setup of the Command Center: **Add System** dialog box:

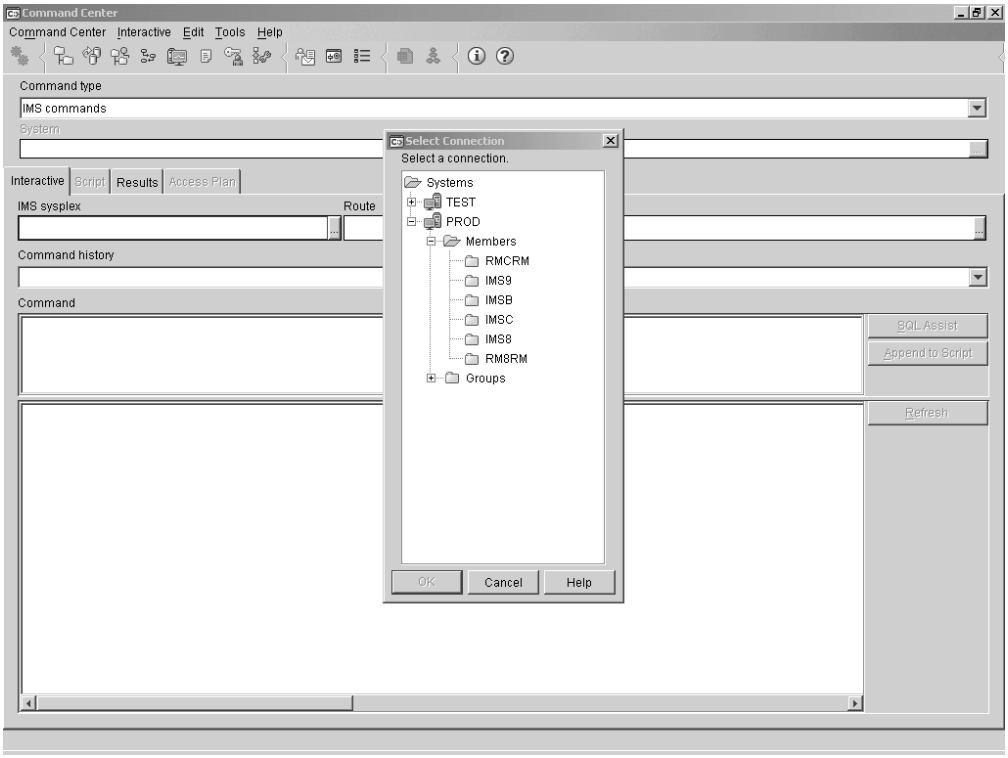


Figure 31. Example of the Command Center initial setup

---

5. To issue an IMS command:
   a. In the **Route** field, selected the IMSplex member to which the command is to be issued
   b. In the **Command** field, entered the IMS command to be issued
   c. Clicked the **Execute** icon at the far left side of the icon bar, in the upper left corner

   Example: Figure 32 on page 109 is an example of using the Command Center to issue the DIS QCNT LTERM MSGAGE 0 command to member IMSC in our PROD IMSplex:

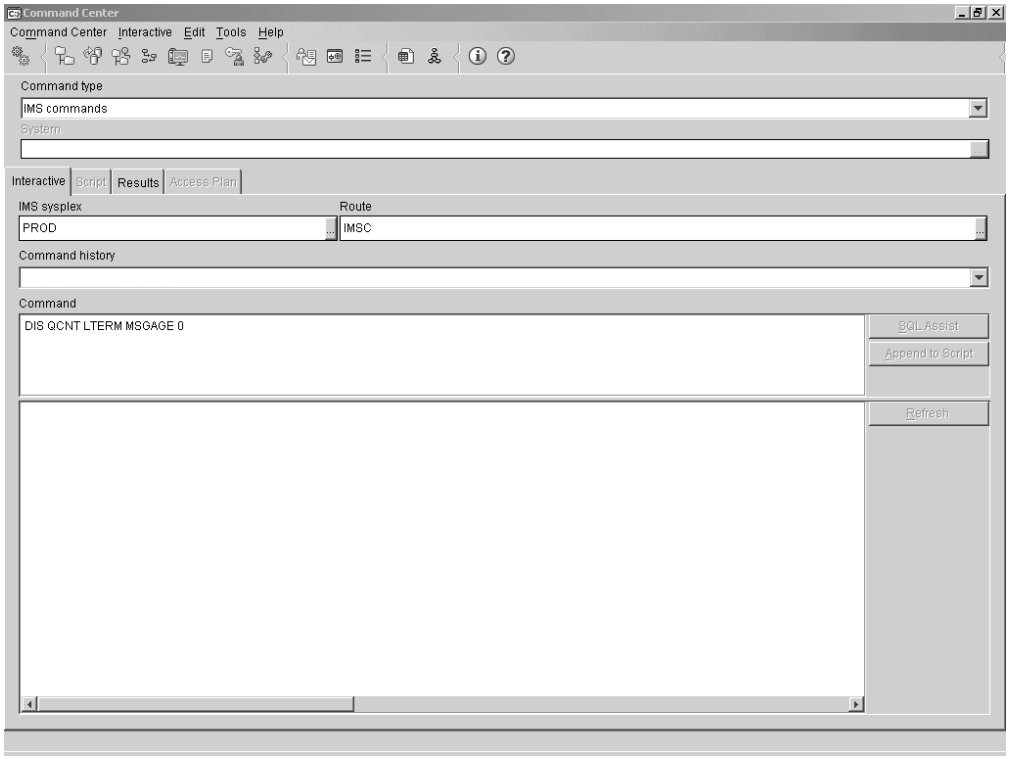


Figure 32. Example of issuing an IMS command to IMSplex member IMSC

Result: Figure 33 on page 110 is an example of the response to the DIS QCNT LTERM MSGAGE 0 command that was issued to member IMSC. Note that the response appears on a separate tab, **Results**, in the Control Center display.

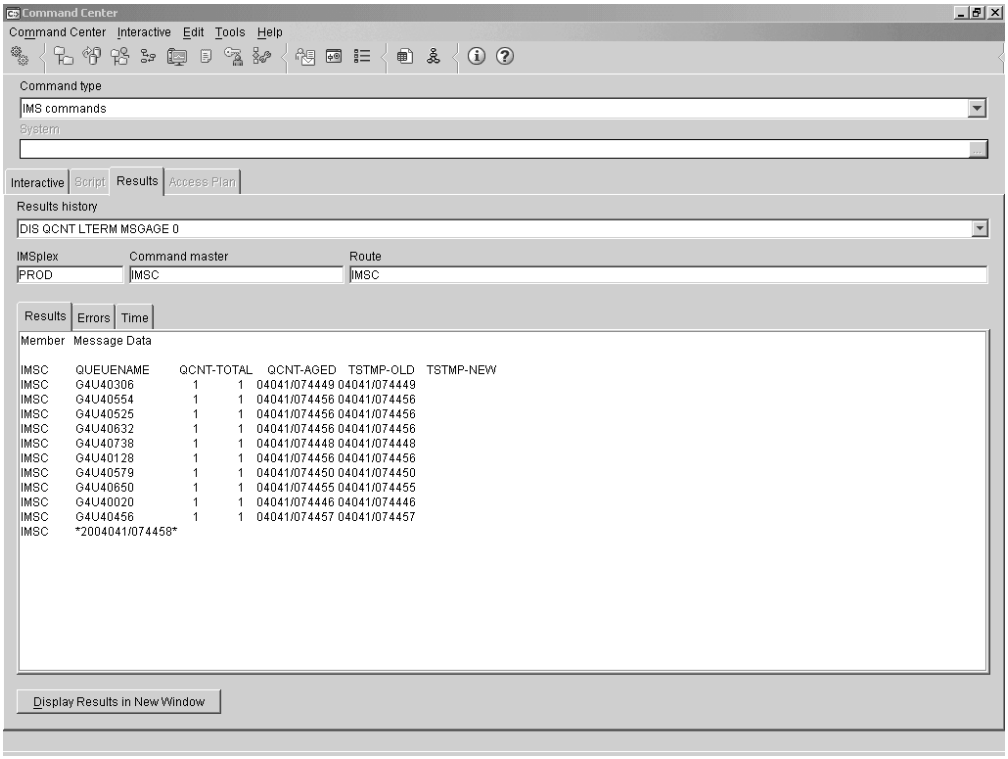


Figure 33. Example of the response to an IMS command that was issued to IMSplex member IMSC

Example: Figure 34 on page 111 is an example of issuing the DIS LINE 1 command to all members of the IMSplex by selecting `All_Members` in the **Route** field:

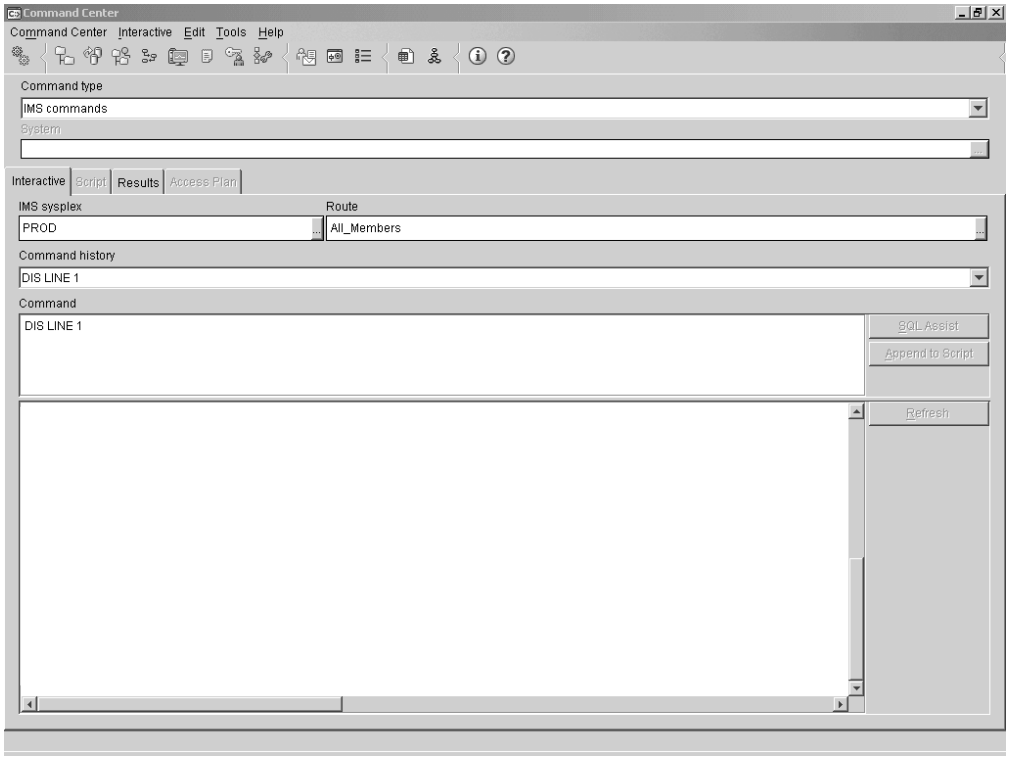


Figure 34. Example of issuing an IMS command to all members of the IMSplex

Result: Figure 35 on page 112 is an example of the response to a command that was issued to all members of the IMSplex:

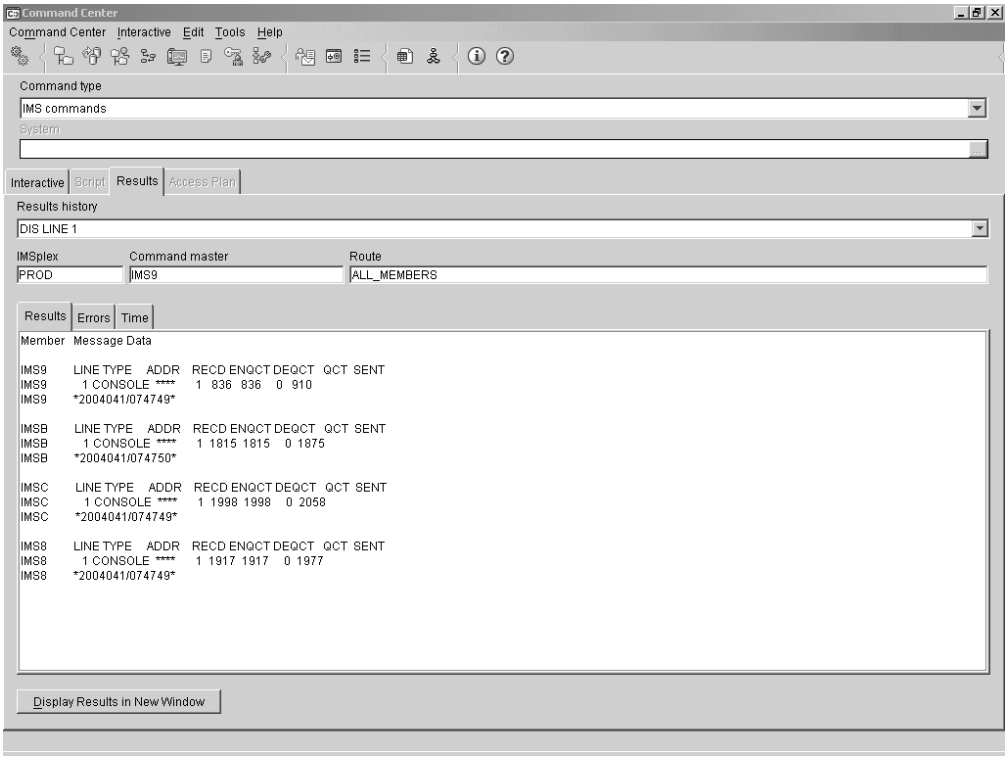


Figure 35. Example of the response to an IMS command that was issued to all members of the IMSplex

Chapter 9. Testing SPE Console Restructure (APAR OA09229)

We installed and tested with the V1R6 level of APAR OA09229 (concerning SPE Console Restructure). Please review the APAR description to see if you are experiencing the reported problem.

Chapter 10. Using IBM Health Checker for z/OS

Because so many system problems are caused by incorrect configuration settings, IBM set out to make it easier to make sure installations have the right configuration settings. They started with a prototype tool, the IBM Health Checker for z/OS and Sysplex prototype, a batch job that analyzes a configuration, looks for exceptions to suggested settings, and returns a report that includes a description of the exception, how to fix it, and where to look for more information. IBM used feedback from users of the popular prototype to transform it into a z/OS product. The new IBM Health Checker for z/OS is an integrated part of z/OS V1R7 as well as being available as a Web deliverable for z/OS V1R4, R5, and R6 of z/OS and z/OS.e. (Note that the Web deliverable is functionally identical to the integrated version and should not be confused with the prototype.)

In this section, we'll report our experiences with both the prototype and the IBM Health Checker for z/OS product:

- “Using the prototype”
- “Using the product”

Using the prototype

The IBM Health Checker for z/OS and Sysplex is a tool that checks the current, active z/OS and sysplex settings and definitions for an image and compares their values to those either suggested by IBM or defined by the installation as the criteria. The objective of the Health Checker is to identify potential problems before they impact system availability or, in the worst cases, cause outages.

We are using Version 3 of the IBM Health Checker for z/OS and Sysplex prototype, which we downloaded from the z/OS downloads page at www.ibm.com/servers/eserver/zseries/zos/downloads/. The documentation, *z/OS and Sysplex Health Checker User's Guide*, SA22-7931, is also available on this Web page.

Using the default USERPARM member supplied with the Health Checker, we created and customized several new members to perform various types of checking. For instance, we use one member with one set of parameters to perform XCF checks, another member with different parameters to perform APF and LINKLST checks, and so on. This also makes the reports easier to look at and use. It also allows us to isolate the checks that examine values that have a sysplex scope so that we can run them on only one z/OS image, rather than on every image in the sysplex—thereby eliminating redundant information from the reports for each image.

Using the product

The current IBM Health Checker for z/OS product is an integrated part of z/OS V1R7, as well as being available as a Web deliverable, called IBM Health Checker, for V1R4, R5, and R6 of z/OS and z/OS.e. The Web deliverable is functionally identical to the integrated version - don't confuse it with the prototype!

IBM Health Checker for z/OS consists of:

- The framework, which provides the services for the checks and the externals for operators and system programmers. It is also called the backbone, and runs on the system as a started task.

- Checks, which look for component, element, or product specific z/OS settings and definitions, checking for potential problems. The specific component or element owns, delivers, and supports the checks. For example, RACF has supplied checks in the RACF element. You can also create your own checks - at this point we are using just the IBM component checks.

A check issues its output as messages, which you can view using SDSF, the HZSPRINT utility, or a log stream that collects a history of check output. If a check finds a potential problem, it issues a WTO exception message text. The check exception messages are also issued to the message buffer in a version including both text and explanation of the potential problem found, including the severity, as well as information on what to do to fix the potential problem.

To get the best results from IBM Health Checker for z/OS, you should let it run continuously on your system so that you will know when your system has changed. When you get an exception, you should resolve it using the information in the check exception message or overriding check values, so that you do not receive the same exceptions over and over.

You can use either the SDSF CK interface, the HZSPRMxx parmlib member, or the IBM Health Checker for z/OS MODIFY (F hzsproc) command to manage checks. We use all of these interfaces to manage IBM Health Checker for z/OS and the component checks. We use SDSF and the MODIFY command to make temporary check changes, and HZSPRMxx to create an IBM Health Checker for z/OS policy that changes checks permanently.

We installed, set up, and are running IBM Health Checker for z/OS on 13 systems with a total of 58 IBM component checks. We used the procedures and examples in *IBM Health Checker for z/OS: User's Guide*, SA22-7994 to do the install and set up. The following are some of the specifics of installation and set up in our environment:

- We copied the following IBM Health Checker for z/OS HZS samples from SYS1.SAMPLIB to a data set we call HCHECKER.PET.JOBS:
  - HZSALLCP
  - HZSMSGNJ
  - HZSPRINT
- We copied the IBM Health Checker for z/OS procedure, HZSPROC, from SYS1.SAMPLIB to our system proclib. We kept the name HZSPROC.
- We copied the HZSALLCP job from SYS1.SAMPLIB and used it to allocate the HZSPDATA data set on each z/OS system where we're running IBM Health Checker for z/OS. The HZSPDATA data set saves check data between restarts. In our environment, we use a high level qualifier of HCHECK.*sysname*.HZSPDATA. We altered HZSPROC to reflect this HZSPDATA data set name.
- We set up an HZSPRMxx parmlib member for each system. You don't have to set up HZSPRMxx parmlib member in order to run IBM Health Checker for z/OS, but we use it to:
  - Make permanent changes to checks, just as deactivating checks that are not appropriate in our environment. Check changes listed in active HZSPRMxx members are applied every time we restart IBM Health Checker for z/OS.
  - Turn on system logger support for IBM Health Checker for z/OS for a system every time you start IBM Health Checker for z/OS.

To identify each member, we use a convention where the *xx* in each HZSPRM*xx* member is the system name. For example, the parmlib member for system J80 would be HZSPRMJ8.

We like to automate everything we can, so we use the system name symbolic to tie the correct parmlib member to the right system when System Automation starts IBM Health Checker for z/OS. For example, System Automation uses the following command to start IBM Health checker for z/OS on system J80:

```
'S HZSPROC,HZSPRM=J8'
```

The following example shows our HZSPRMJ8 parmlib member:

```
/* ------------------------------------------------------------- */
/*                                                               */
/*   LICENSED MATERIALS - PROPERTY OF IBM                        */
/*   5694-A01                                                    */
/*   (C) COPYRIGHT IBM CORP. 2004, 2005                          */
/*                                                               */
/* ------------------------------------------------------------- */
/* The LOGGER command and POLICY statements can be used in HZSPRMxx */
/* to enable logger support, and change the default behavior of  */
/* target checks.   Common syntax of both the LOGGER and POLICY  */
/* statements are documented below.                              */
/* For a complete syntax of the LOGGER command, POLICY statements, */
/* and other commands that can be specified in the HZSPRMxx system */
/* parmlib members, see the                                      */
/* IBM Health Checker for z/OS and Sysplex User's Guide          */
/* ------------------------------------------------------------- */
/*                                                               */
/*                                                               */
   LOGGER=ON,LOGSTREAMNAME=HZS.HEALTH.CHECKER.HISTORY
/*   can be used to enable log stream processing to the specified */
/*   log stream                                                  */
/*                                                               */
/*                                                               */
/* {ADD | ADDREPLACE},POLICY,STATEMENT=statementname,UPDATE,filters,*/
/*        update_options,REASON=(reason text),DATE=yyyymmdd      */
/*                                                               */
/*   can be used to define policy statements that modify the     */
/*   behavior of specified checks.                               */
/*                                                               */
/* Where                                                         */
/*   ADD -     This is a new policy statement that is not active. */
/*             If the named policy statement is already active, the */
/*             new policy statement is rejected.                 */
/*   ADDREPLACE - The specified policy statement may already be  */
/*             active.                                           */
/*             If the policy statement is already active, the    */
/*             existing policy statement is replaced.            */
/*   statementname - 1-16 character policy statement name used to */
/*             identify the policy statement.                    */
/*   UPDATE -  Indicates the policy statement overrides check    */
/*             defaults                                          */
/*   filters - Filters that indicate which check(s) are targeted by */
/*             this policy statement:                            */
/*             CHECK(owner,name) - (Required.) The 1-6 character */
/*                                 check owner, and 1-32 character */
/*                                 check name.  Wild card symbols */
/*                                 '*' and '%' are permitted.    */
/*             CATEGORY(filter_type,Ù category-1,category-2Ù...) */
/*                Allows additional filter capacity based on the */
/*                current assigned categories                    */
/*             EXITRTN=exitrtnÙ - The name of the HZSADDCHECK    */
/*                                 dynamic exit routine that was */
/*                                 used to add the check.        */
/*   Update_options-The options used to override the check       */
```

```
/*            defaults:                                           */
/*            ,ACTIVE|INACTIVEÙ                                   */
/*               Indicates the target check(s) are ACTIVE or      */
/*               INACTIVE.                                        */
/*            ,ADDCAT=(cat1,...,cat16)Ù                           */
/*               Add the target check(s) to the specified         */
/*               categories                                       */
/*            ,DESCCODE=(desccode1,...,desccode#)Ù                */
/*               Additional descriptor code(s) which will be used */
/*               when an exception message is written by the      */
/*               target check(s)                                  */
/*            ,{INTERVAL=ONETIME|INTERVAL=hhh:mm}Ù                */
/*               The interval at which the target check(s) will   */
/*               be run.                                          */
/*            ,PARM=parameterÙ                                    */
/*               The check specific parameter that will be passed */
/*               to the target check.                             */
/*            ,ROUTCODE=(routcode1,..,routcode#Ù)Ù                 */
/*               Additional route code(s) which will be used when */
/*               an exception message is written by the target    */
/*               check(s)                                         */
/*            ,SEVERITY={HIGH|MEDIUM|LOW|NONE}Ù                   */
/*               The severity of target check(s)                  */
/*            ,WTOTYPE={CRITICAL|EVENTUAL|                        */
/*                      INFORMATIONAL|HARDCOPY|NONE}Ù             */
/*               Specifies what message will be used when an      */
/*               exception message is written by the target       */
/*               check(s).  Note: If WTOTYPE is not specified,    */
/*               the message is determined by the check severity. */
/*   REASON -  1-126 character reason that documents why the policy */
/*             statement statement was added to HZSPRMxx.         */
/*   DATE   -  (yyyymmdd) The date when the policy statement was   */
/*             added to HZSPRMxx.                                 */
/*                                                                */
/*----------------------------------------------------------------*/
/*----------------------------------------------------------------*/  00000491
/*                                                                */  00000981
/* MACRO-STMT:                                                    */  00001471
/*                                                                */  00001961
/* DESCRIPTIVE-NAME: Exceptions  HZSPRMxx Member                  */  00002451
/*                                                                */  00002941
/*  LICENSED MATERIALS - PROPERTY OF IBM                          */  00003431
/*  5694-A01                                                      */  00003921
/*  (C) COPYRIGHT IBM CORP. 2005                                  */  00004411
/*                                                                */  00004901
/* STATUS:                                                        */  00005391
/*                                                                */  00005881
/* FUNCTION: This is to override polices to make them inactive    */  00006371
/*           until the exceptions they generate get fixed         */  00006861
/*                                                                */  00007351
/*                                                                */  00007841
/* ============================================================== */  00008331
/*                                                                */  00008821
/* CHANGE-ACTIVITY:                                               */  00009311
/*                                                                */  00010781
/* -------------------------------------------------------------- */  00011271
/* ---------------------------------------------------------------*/
/*  CHECK(IBMCNZ,CNZ_CONSOLE_MSCOPE_AND_ROUTCODE)                 */
/* ---------------------------------------------------------------*/
ADDREPLACE POLICY STMT(CNZ_POLICY_2)
UPDATE CHECK(IBMCNZ,CNZ_CONSOLE_MSCOPE_AND_ROUTCODE)
       DATE(20050728)
       INACTIVE
       SEVERITY(LOW)
       INTERVAL(24:00)   /* RUN ONCE A DAY */
REASON('Make Check inactive to fixed')
/* -------------------------------------------------------------- */
```

```
/*  CHECK(IBMRACF,RACF_SENSITIVE_RESOURCES_DFLT)                    */
/*                                                                  */
/* Check that certain sensitive resources are protected.            */
/*                                                                  */
/* PARAMETERS:                                                      */
/*       1. A user ID may be specified. The                         */
/*         RACF_SENSITIVE_RESOURCES check performs an authorization */
/*         check using this user ID. This parameter is optional.    */
/* ---------------------------------------------------------------- */
ADDREPLACE POLICY STMT(RACF_SENS_DFLT)
UPDATE CHECK(IBMRACF,RACF_SENSITIVE_RESOURCES)
       DATE(20050617)
       INACTIVE
       SEVERITY(HI)
       INTERVAL(08:00)
REASON('Make Check inactive to fixed')
/*-----------------------------------------------------------------*/
/*-----------------------------------------------------------------*/
/* CHECK(IBMRSM,RSM_MEMLIMIT)                                       */
/*                                                                  */
/*                                                                  */
/*   AUDITS THE SETTING OF MEMLIMIT IN SMFPRMxx                     */
/*                                                                  */
/*  PARAMETERS:                                                     */
/*                                                                  */
/*   NONE                                                           */
/*-----------------------------------------------------------------*/
ADDREPLACE POLICY STMT(RSM_MEMLMDEF)
UPDATE CHECK(IBMRSM,RSM_MEMLIMIT)
       DATE(20050802)
       INACTIVE
       SEVERITY(LOW)
       INTERVAL(ONETIME)
 REASON('Make Check inactive to fixed')
/*-----------------------------------------------------------------*/
/*  CHECK(IBMXCF,XCF_CF_STR_PREFLIST)                               */
/*                                                                  */
/* Check that each structure is where it is desired to be based on  */
/* its preference list.                                             */
/*                                                                  */
/* PARAMETERS:                                                      */
/*   1. NONE                                                        */
/* ---------------------------------------------------------------- */
ADDREPLACE POLICY STMT(XCFPOL15)
UPDATE CHECK(IBMXCF,XCF_CF_STR_PREFLIST)
       DATE(20050617)
       INACTIVE
       SEVERITY(MED)
       INTERVAL(08:00)
 REASON('Make Check inactive to fixed')
/*-----------------------------------------------------------------*/
/*  CHECK(IBMGRS,GRS_CONVERT_RESERVES)                              */
/*                                                                  */
/* Check that all RESERVES are being converted if in STAR mode.     */
/*                                                                  */
/* PARAMETERS: None.                                                */
/* ---------------------------------------------------------------- */
ADDREPLACE POLICY STMT(IBMGRS_DEFAULT03)
UPDATE CHECK(IBMGRS,GRS_CONVERT_RESERVES)
       DATE(20050616)
       INACTIVE
       SEVERITY(LOW)
       INTERVAL(ONETIME)
REASON('Make Check inactive to fixed')
```

Our approach to automation with IBM Health Checker for z/OS

There are numerous ways you can automate IBM Health Checker for z/OS and its exception messages, depending on the products installed in your shop and a million other variables. Right now, we've implemented a very simple approach to automation, we may add to that in the future. For more on automation, see More automation ideas in *IBM Health Checker for z/OS: User's Guide*.

Our approach to automation on a test sysplex:

1. **Automate start up:** We set up our systems so that IBM Health Checker for z/OS starts automatically every time a system IPLs. We do this by running the HZSPROC from our System Automations.
2. **Automate HZSPRINT to keep a record of check messages on each system:** We use System Automation running under NetView to automate HZSPRINT. We code the HZSPRINT JCL so that it automatically prints the messages from checks that found an exception. You can code the JCL for HZSPRINT so that it prints the message buffer to a sequential data set or simply to SYSOUT. Our JCL looks prints the message buffer data to a sequential data set for any check that finds an exception, as shown in the following example:

```
//HZSPRINT JOB 'ACCOUNTING INFORMATION','HZSPRINT JOB',
//         CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1)
/*JOBPARM SYSAFF=*
//HZSPRINT EXEC PGM=HZSPRNT,TIME=1440,REGION=0M,
//*   PARM=('CHECK(check_owner,check_name)')
//    PARM=('CHECK(*,*)',
//    'EXCEPTIONS')
//*   PARM=('LOGSTREAM(HZS.HEALTH.CHECKER.HISTORY)')
//*   PARM=('LOGSTREAM(HZS.HEALTH.CHECKER.HISTORY)',
//*   'CHECK(owner,name)')
//*   PARM=('LOGSTREAM(HZS.HEALTH.CHECKER.HISTORY)','EXCEPTIONS',
//*   'CHECK(owner,name)')
//*   PARM=('LOGSTREAM(HZS.HEALTH.CHECKER.HISTORY)','EXCEPTIONS')
//*   PARM=('LOGSTREAM(HZS.HEALTH.CHECKER.HISTORY)','SYSNAME(sysname)')
//*   PARM=('LOGSTREAM(HZS.HEALTH.CHECKER.HISTORY)','SYSNAME(sysname)',
//*   'CHECK(owner,name)')
//*   PARM=('LOGSTREAM(HZS.HEALTH.CHECKER.HISTORY)','EXCEPTIONS',
//*   'SYSNAME(sysname)',
//*   'CHECK(owner,name)')
//*   PARM=('LOGSTREAM(HZS.HEALTH.CHECKER.HISTORY)','EXCEPTIONS',
//*   'SYSNAME(sysname)')
//*YSOUT   DD SYSOUT=A,DCB=(LRECL=256)
//SYSOUT   DD DSN=HCHECKER.PET.CHKEXCPT.SEQ.REPORT,DISP=MOD
```

3. **Automate HZSPRINT on each system to send e-mail messages:** You can add a step to the HZSPRINT JCL for each system that uses the Simple Mail Transfer Protocol (SMTP) FTP command to send e-mail messages. To do this, you must have SMTP set up - see *z/OS Communications Server: IP User's Guide and Commands*. We're using SMTP to send an e-mail alert whenever a check finds an exception. To do this, we key off of the HZS exception messages - see Using HZS exception messages for automation in *IBM Health Checker for z/OS: User's Guide*.

Part 2. Networking and application enablement

The following chapters describe the networking and application enablement aspects of our computing environment.

Chapter 11. About our networking and application enablement environment

In this chapter we describe our networking and application enablement environment, including a high-level view of our configurations and workloads. We discuss networking and application enablement together because the two are greatly intertwined. You need the networking infrastructure in place before you can run many of the application enablement elements and features.

Our networking and application enablement configuration

Figure 36 illustrates at a high level our networking and application enablement configuration. In the figure, the broad arrows indicate general network connectivity of a given type, rather than specific, point-to-point, physical connections.

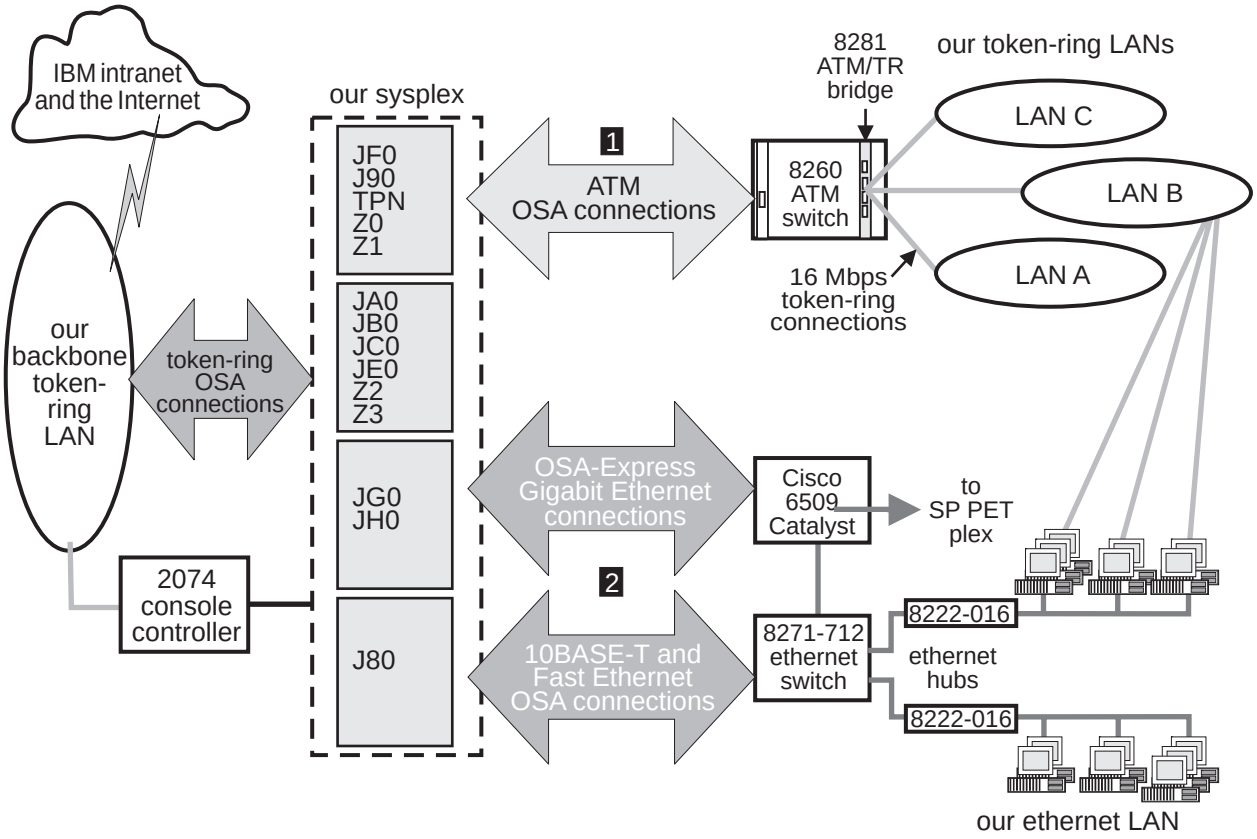


Figure 36. Our networking and application enablement configuration

Note the following about Figure 36:

- We use the following OSA features to connect our systems to our LANs:
  - OSA-2
    ENTR (Ethernet/Token Ring)
  - OSA-Express
    ATM (Asynchronous Transfer Mode)
    FENET (Fast Ethernet)
    Gigabit Ethernet (GbE)

- All ATM connections (1) use ATM LAN emulation; we have no native ATM connections. Although not shown, some of our CPCs that do not have an ATM connection instead use OSA-2 ENTR to directly connect to each of our token-ring LANs.
- Host system Ethernet connections (2) use either OSA-2 ENTR 10BASE-T, OSA-Express FENET, or OSA-Express Gigabit Ethernet features depending on the CPC model and the type of adapter it supports.
- Although not shown, all RS/6000s on LAN A also have a direct connection to the backbone token ring.
- We recently replaced our Gigabit Ethernet switch with a Cisco 6509 Catalyst switch. This new switch handles all of our Gigabit Ethernet and some of our Ethernet connections. Eventually, all of our Ethernet traffic will flow through the Cisco 6509 and we'll remove the 8271-712 Ethernet switch. (For technical details about the Cisco 6500 Catalyst family, go to http://www.cisco.com.)
- We also added Gigabit Ethernet connectivity between our sysplex and a remote AIX cluster owned by the SP PET team. We use this connectivity for the AIX portion of our bookstore application.

For an illustration of our VTAM configuration, see "Our VTAM configuration" on page 17.

If you are familiar with our test reports, then you know that we have always described our networking configuration in exacting detail, as we felt that the complexity of our environment required a great deal of explanation. For example, at one time we were very specific about which of our system images could access which of our network resources. However, as we progress, we are concentrating more and more on TCP/IP and expanding our use of Ethernet. As a result, things are becoming more similar than dissimilar and connectivity between our host systems and network resources is approaching any-to-any.

Accordingly, we have shifted our networking discussion to a somewhat more conceptual level and focus on how our infrastructure enables us to test and exploit new features and functions. We will continue to highlight specific aspects of our configuration as significant changes occur and we introduce new technologies.

Our Ethernet LAN configuration

Our network configuration includes an Ethernet LAN. We primarily use it for FTP testing from Windows 95 and Windows NT clients and for VIPA testing. (For more information about VIPA, see our December '99 edition.) Many of our Ethernet client workstations also contain a token-ring adapter that connects the workstations to our token-ring LAN B as well. We use an OS/2 LAN Server on LAN B to drive the FTP testing on the Ethernet clients. You can read more about this setup in "What's happening in LAN B?" on page 131

Our systems' Ethernet connectivity includes a combination of 10BASE-T, Fast Ethernet, and Gigabit Ethernet connections using OSA-2 ENTR, OSA-Express FENET, and OSA-Express Gigabit Ethernet features, respectively.

Note that the connections between our OSA-Express Gigabit Ethernet features and our Cisco 6509 Catalyst switch operate at 1000 Mbps. The Fast Ethernet connections between our OSA-Express FENET features and our 8271 Ethernet switch, as well as those between our Cisco 6509 and the 8271, operate at 100 Mbps. The 10BASE-T connections from the 8271 to the 8222 Ethernet hubs and client workstations operate at 10 Mbps.

The OSA-Express FENET feature operates at either 10 or 100 Mbps in half- or full-duplex mode and supports auto-negotiation with its attached Ethernet hub, router, or switch. We used the latest edition of *zSeries OSA-Express Customer's Guide and Reference* and the OSA/SF GUI for Windows to install and configure the OSA-Express FENET feature. (See our December 1999 edition for our experiences installing the OSA/SF GUI for Windows.) We also recommend that you check with your IBM support representative to ensure that you have the latest microcode level for this feature.

Our ATM configuration

As we note above, our configuration includes OSA-Express ATM features operating in LAN emulation mode only. Therefore, when you see the term *ATM* in this chapter, understand it to mean *ATM LAN emulation*. (See our December 1998 edition for details on our ATM implementation.)

We use ATM for high-speed, bi-directional, asynchronous connectivity between our z/OS systems and our 8260 ATM switch. The 8260 then connects to the 8281 LAN bridge and provides access to all three of our token-ring LANs. The ATM links operate at 155 Mbps while the token-ring LANs still operate at 16 Mbps. Therefore, the maximum combined token-ring traffic from all three LANs is only 64 Mbps, which *each* ATM link easily accommodates.

Note that because of the wide variety of hardware we employ, not every CPC in our sysplex has an ATM connection. For those CPCs that do not, we use OSA-2 ENTR to provide direct connections to each of our token-ring LANs. Either way, it's all transparent to the end user.

Our ipV6 Environment Configuration

With z/OS V1R6, we now have an ipV6 environment equivalent to our ipV4 environment. V1R6 now supports OSPF V3 for ipV6 and ipV6 support for DVIPA and Sysplex Distributor.

We used the following manuals as guides in setting up ipV6.

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IPv6 Network and Application Design Guide*, SC31-8885

To configure a z/OS image for ipV6 the following changes have to be made:

Note: This is not meant to be an all inclusive guide for ipV6 setup.

1. Add new NETWORK, AF_INET6 to BPXPRMxx statement.
2. Add New INTERFACE to TCPIP profile for ipV6 'device'.
3. Add support for DYNAMIC XCF
4. Create DVIPA for ipV6
5. Add INTERFACE to OMPROUTE profile.
6. Make appropriate additions to Nameserver.

z/OS UNIX System Services changes and additions

The following are the changes and additions we made to z/OS UNIX System Services:

1. Changing BPXPRMxx to add ipV6 support

We made the following changes to BPXPRMxx to add ipV6 support:

```
NETWORK DOMAINNAME(AF_INET6)
DOMAINNUMBER(19)
MAXSOCKETS(60000)
TYPE(INET)
```

Note: INADDRANYPORT and INADDRANYCOUNT values are used for both ipV4 and ipV6 when the BPXPRMxx is configured for ipV4 and ipV6 support. If AF_INET is specified, it is ignored and the values from the NETWORK statement for AF_INET are used if provided. Otherwise, the default values are used.

2. Adding NETWORK statements to have a stack that supports ipV4 and ipV6

   We added the following two NETWORK statements to have a stack that supports ipV4 and ipV6:

```
FILESYSTYPE TYPE(CINET) ENTRYPOINT(BPXTCINT)
NETWORK DOMAINNAME(AF_INET)
DOMAINNUMBER(2)
MAXSOCKETS(2000)
TYPE(CINET)
INADDRANYPORT(20000)
INADDRANYCOUNT(100)
NETWORK DOMAINNAME(AF_INET6)
DOMAINNUMBER(19)
MAXSOCKETS(3000)
TYPE(CINET)
SUBFILESYSTYPE NAME(TCPCS) TYPE(CINET) ENTRYPOINT(EZBPFINI)
SUBFILESYSTYPE NAME(TCPCS2) TYPE(CINET) ENTRYPOINT(EZBPFINI)
SUBFILESYSTYPE NAME(TCPCS3) TYPE(CINET) ENTRYPOINT(EZBPFINI)
```

TCPIP Profile changes

We made the following additions to our IPv6 INTERFACE statements:

```
INTERFACE OSA9E0V6
DEFINE IPAQENET6
PORTNAME GBPRT9E0
IPADDR FEC0:0:0:1:x:xx:xx:xxx  ;(Site-Local Address)
3FFE:0302:0011:2:x:xx:xx:xxx ; (Global Address)
```

Note: In order to configure a single physical device for both IPv4 and IPv6 traffic, you must use DEVICE/LINK/HOME for the IPv4 definition and INTERFACE for the IPv6 definition, so that the PORTNAME value on the INTERFACE statement matches the device_name on the DEVICE statement.

Dynamic XCF addition

We made the following addition for our Dynamic XCF:

```
IPCONFIG6 DYNAMICXCF FEC0:0:0:1:0:168:49:44
```

Dynamic VIPA additions

The following statement was added to our VIAPDYNAMIC section:

Note: V6Z2FTP is the INTERFACE name for this VIPA.

```
VIPADEFINE V6Z2FTP 2003:0DB3:1::2
VIPADISTRIBUTE SYSPLEXPORTS V6Z2FTP PORT 20 21
DESTIP FEC0:0:0:1:0:168:49:37
```

OMPROUTE addition

Setting up OMPROUTE only requires adding the INTERFACE name to the OMPROUTE profile for the basic setup that we used.

```
IPV6_OSPF_INTERFACE
Name = OSA9E0V6;
```

Note: During testing we encountered the following message:

```
EZZ7954I  IPv6 OSPF adjacency failure, neighbor 192.168.25.33, old state 128,
 new state 4, event 10
```

The neighbor id in the message is the ROUTERID from the OMPROUTE profile. It will not show an ipV6 address.

NAMESERVER changes

We created seperate ipV6 names for each LPAR. To keep things simple for the system name, we used the existing LPAR name with IP6 as the suffix. For the ipV6 ip addresses, we used a common prefix and used the ipV4 address as the suffix. This made it easier to identify for diagnosing problems.

Forward file changes

The following change was made to our forward file:

```
J80IP6                IN AAAA 3FFE:302:11:2:9:12:20:150
```

Reverse file entry addition

We added the following for the reverse file entry:

```
$TTL 86400
$ORIGIN 2.0.0.0.1.1.0.0.2.0.3.0.E.F.F.3.IP6.ARPA.
@       IN SOA  Z0EIP.PDL.POK.IBM.COM.  ALEXSA@PK705VMA (
                  012204  ;DATE OF LAST CHANGE TO THIS FILE
                  21600   ;REFRESH VALUE FOR SECONDARY NS (IN SECS)
                  1800    ;RETRY VALUE FOR SECONDARY NS (IN SECS)
                  48384   ;EXPIRE DATA WHEN REFRESH NOT AVAILABLE
                  86400 ) ;MINIMUM TIME TO LIVE VALUE (SECS)
@                  IN NS   Z0EIP.PDL.POK.IBM.COM. ; PRIMARY DNS
0.5.1.0.0.2.0.0.2.1.0.0.9.0.0.0 IN PTR J80IP6.PDL.POK.IBM.COM.
```

Our token ring LAN configuration

As Figure 36 illustrates, we have a total of four token-ring LANs: a backbone ring and three test LANs that use various:

- Communications protocols (TCP/IP, SNA, NetBIOS, and Internet Packet Exchange (IPX))
- Workstation operating systems (AIX, Linux, OS/2, PC DOS, and Microsoft Windows NT, Windows 95, and Windows 2000)
- Workstation types (RS/6000s and various types of PCs)

LANs A, B, and C in Figure 36 use only the token-ring LAN protocol. The three LANs connect to our host systems through the 8281 LAN bridge and 8260 ATM switch as described above. (You can read about our ATM experiences in our December 1998 edition.) For host systems running on CPCs that do not have an ATM connection, we instead use OSA-2 ENTR features to provide direct token-ring connections to each of the three LANs (these connections are not shown in Figure 36).

Note that we also have an OS/2 LAN Server with a CLAW protocol channel adapter that connects to system JE0 for LAN Server. This is not shown in Figure 36; see Figure 39 on page 133 for an illustration of this.

All of the systems in our sysplex can connect to the IBM SNA network using VTAM as long as either system Z0 or system J80 (the network node server) is available. In addition, all systems can get to the IBM TCP/IP network directly through our backbone token ring.

We discuss how we use the backbone and LANs A, B, and C in greater detail in the following sections.

More about our backbone token ring

The token-ring backbone connects our test environment to the IBM corporate network or intranet and, beyond that, to the Internet. Rather than exist as an isolated entity, our ability to connect to the rest of the corporation and to the outside world yields us a much more viable and robust test environment. Some specific advantages include:

- We are able to access our network resources from our offices or while working from home, instead of having to be on the test floor all the time. This convenience and flexibility allows us to be more productive.
- We can perform more complete and realistic test scenarios with products and features, such as:
  - Tivoli Storage Management (TSM, formerly ADSM)
  - Firewall
  - NFS
  - Infoprint® Server
  - Rlogin
  - Telnet
  - Web access
- When we encounter a complex problem, we are able to have product developers from our local site and from other IBM locations work with us in our own test environment to help diagnose and resolve the problem.
- We keep our own documentation, such as test plans and run procedures, on our Web server and can access it from anywhere. As a result, we also implicitly test our networking environment just by performing our day-to-day administrative work.
- We install our configuration tools (for Firewall and OSA/SF, for example) on workstations attached to the backbone so that we can provide central access to the tools and share them across multiple systems.

For many of the same reasons, we also recently switched from running our RS/6000 workloads on LAN A to running them on the backbone, mostly to allow greater access to other resources and provide more realistic testing. See the next section for more about LAN A.

What's happening in LAN A?

Our RS/6000 workstations reside on LAN A but they also directly connect to the backbone. This additional connectivity allowed us to shift a majority of the workloads that we once performed exclusively on LAN A over to the backbone. LAN A itself still exists in our environment, but we don't use it for anything special from a functional standpoint.

Figure 37 on page 131 depicts our z/OS UNIX DCE test configuration in LAN A, including the connections from the RS/6000s to the backbone (which, for clarity, are not shown in Figure 36 above).

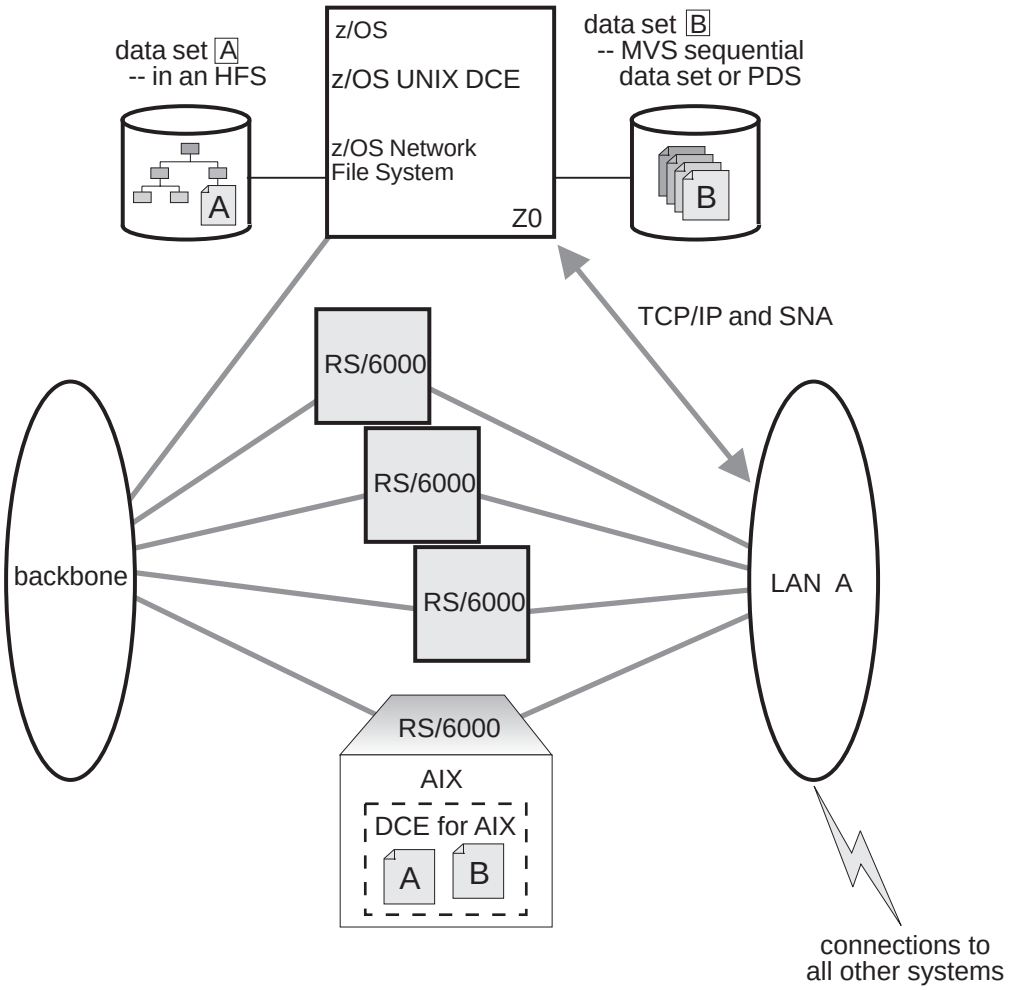


Figure 37. Our token-ring LAN A

The RS/6000 workstations on LAN A all run the AIX operating system. We use them to exercise the z/OS UNIX, DCE, and DFS™ functions. See “Our workloads” on page 18 for a more detailed description of these workloads. For more information about DCE and DFS, see “DCE and DFS Publications” in Appendix E, “Useful Web sites,” on page 345.

What’s happening in LAN B?

You might recall from our December 1996 edition that our LANs B and C started out as two functionally separate LANs. Later on, we combined their functionality and collectively referred to them as logical LAN BC. Well, we’ve now come full circle. For better performance and throughput, we are back to using LANs B and C as two functionally separate LANs.

LAN B has an OS/2 NFS function and an FTP function using TCP/IP. The OS/2 LAN Server on LAN B acts as a control workstation for our NFS and FTP workloads. The control data consists of the commands that start, stop, and otherwise regulate the execution of the workloads. The test data or workload data is the actual data that the workloads manipulate.

The workstations that run the FTP workloads connect to both our token-ring LAN B and to our Ethernet LAN. The FTP control data comes from the OS/2 server to the clients over LAN B. The test data that the workloads manipulate travels over the Ethernet LAN.

The NFS function communicates with z/OS NFS using TCP/IP, and both the control data and the workload data travel over LAN B. (See “Comparing the network file systems” on page 134 for a description of the different types of NFSs we use.)

Figure 38 depicts our NFS and FTP test configuration in LAN B. For more information about NFS, see “Network File System Publications” in Appendix E, “Useful Web sites,” on page 345.

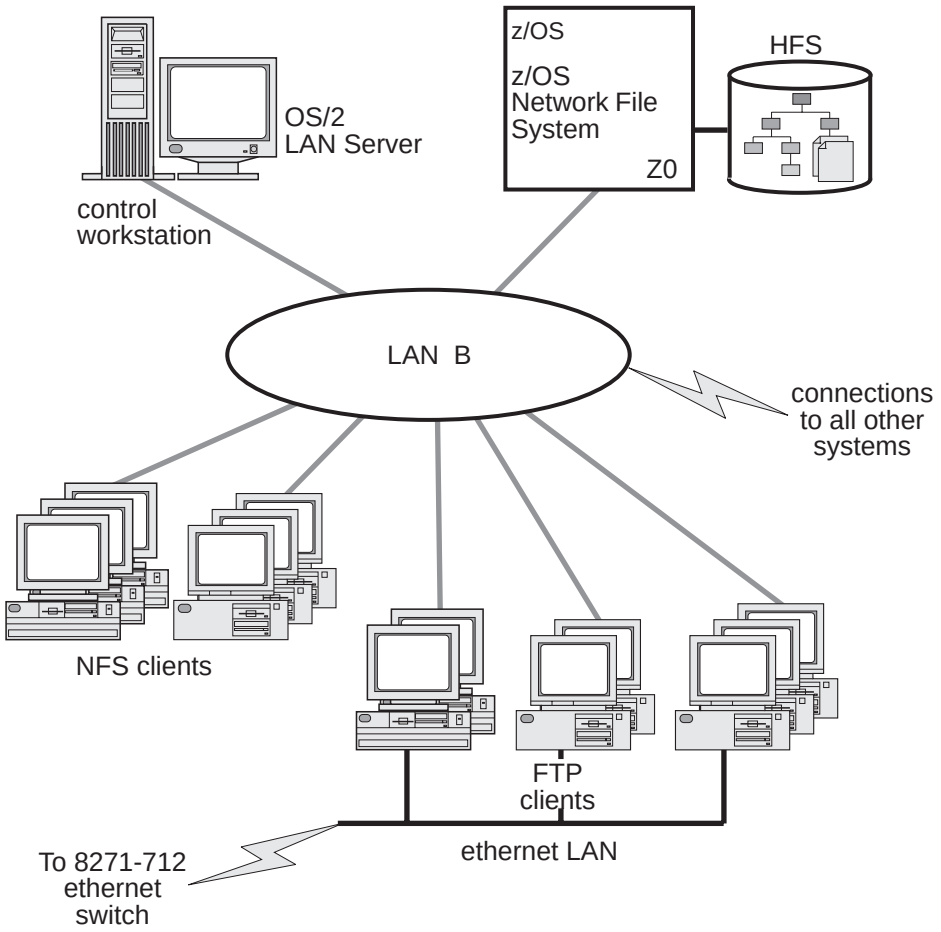


Figure 38. Our token-ring LAN B

What’s happening in LAN C?

LAN C runs two different types of LAN Server scenarios using OS/2 LAN Servers as front-end processors (FEPs) to z/OS LAN Server. z/OS LAN Server expands the file storage capability of the OS/2 LAN Servers by storing workstation-format files in VSAM linear data sets on the z/OS host. These data sets are not readable by MVS users, but appear to the clients as though they are stored on the OS/2 LAN Servers.

First, we have an OS/2 LAN Server acting as a FEP (**A**) with a SNA connection to LAN Server in system Z0. We could conceivably connect the OS/2 LAN Server to

any z/OS system, but we currently happen to be using Z0. We use Communications Manager/2 for the SNA connection and APPC communications.

We also have another OS/2 LAN Server acting as a FEP (B) with a CLAW protocol connection to LAN Server in system JE0.

Typically, a LAN file server contains one or more large-capacity hard disk drives on which it stores files for access by the clients (or requesters). However, in our setup, the OS/2 LAN Servers do not store any workload-related programs or data on their own hard disks for use by the clients. All the workload-related programs and data reside on the z/OS system. This is completely transparent to the requesters, as they are only aware of the OS/2 LAN Servers which, in turn, interact with z/OS LAN Server on the host. The OS/2 servers do keep setup files, automation programs, and workstation configuration files on their own local disk drives.

Figure 39 depicts our LAN Server test configuration in LAN C. For more information about LAN Server, see “LAN Server Publications” in Appendix E, “Useful Web sites,” on page 345.

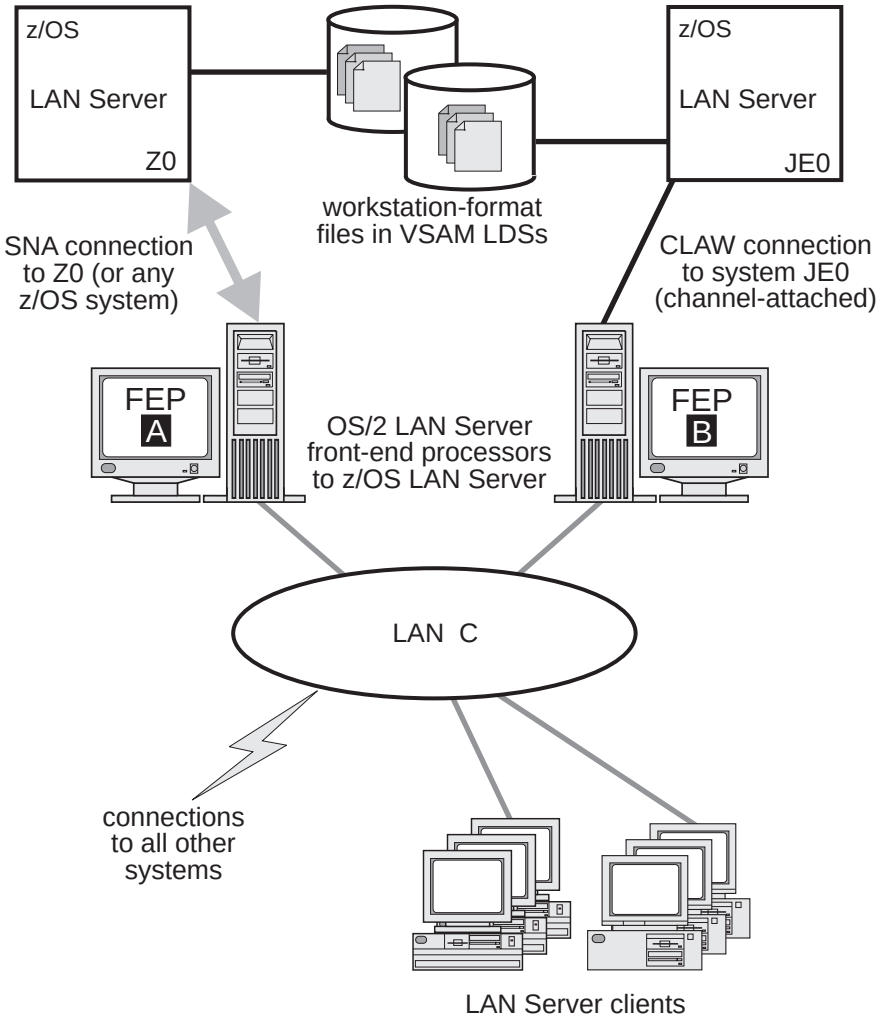


Figure 39. Our token-ring LAN C

Comparing the network file systems

If you are a faithful reader of our test report, you might have noticed that we have changed our Network File System (NFS) approach a number of times, depending on the circumstances at the moment. Currently, we have the z/OS NFS (called DFSMS/MVS® NFS in OS/390 releases prior to R6) on system Z0.

NFS allow files to be transferred between the server and the workstation clients. To the clients, the data appears to reside on a workstation fixed disk, but it actually resides on the z/OS server.

With z/OS NFS, data that resides on the server for use by the workstation clients can be either of the following:

- z/OS UNIX files that are in a hierarchical file system (HFS). The z/OS NFS is the only NFS that can access files in an HFS. You need to have z/OS NFS on the same system as z/OS UNIX and its HFS if you want to use the NFS to access files in the HFS.
- Regular MVS data sets such as PS, VSAM, PDSs, PDSEs, sequential data striping, or direct access.

Migrating to the z/OS NFS: We plan to implement some of the new functions available in z/OS NFS, such as file locking over the z/OS NFS server and file extension mapping support. You can read descriptions of these new functions in *z/OS Network File System Guide and Reference*, SC26-7417. In addition, you can read about WebNFS support in our December 1999 edition, and the use of the LAN Server NFS in our June 2004 edition. We hope to have additional experiences with these new functions to share with you in a future test report.

Note that APAR OW40134 recommends a change to the SHAREOPTIONS specified in the sample JCL for the IDCAMS job used to allocate the mount handle data sets. This sample JCL is both shipped in *hlq*.NFSSAMP(GFSAMHDJ) and illustrated in *z/OS Network File System Guide and Reference*. The sample JCL currently uses SHAREOPTIONS(3 3). However, the APAR instead recommends SHAREOPTIONS(1 3). While the sample code does work as it stands, it allows programs other than NFS to update the files. Using SHAREOPTIONS(1 3) limits the possibility of corruption to the mount handle database.

Networking and application enablement workloads

For information about our networking and application enablement workloads, see “Our workloads” on page 18.

Enabling NFS recovery for system outages

In z/OS V1R6, we improved NFS recoverability and availability by using Automatic Restart Management (ARM) and dynamic virtual IP address (DVIPA) with our NFS server. With these enhancements, the NFS server is automatically moved to another MVS image in the sysplex during a system outage.

Note: We are running a shared HFS environment.

We used the following documentation to help us implement ARM for NFS recovery.

- Automatic Restart Management
  - ARMWRAP as described in the IBM Redpaper *z/OS Automatic Restart Manager* available on the IBM Redbooks Web site.

  - *z/OS MVS Setting Up a Sysplex*, SA22-7625
- Dynamic VIPA(DVIPA)
  - *z/OS Communications Server: IP Configuration Guide*, SC31-8775

Setting up the NFS environment for ARM and DVIPA

Part 1 of Figure 40 on page 136: illustrates how the NFS server on MVS A acquires DVIPA 123.456.11.22. The AIX clients issue a hard mount specifying DVIPA 123.456.11.22. Before the enhancements, the AIX clients specified a static IP address for MVS A. A system outage would result in the mounted file systems being unavailable from the AIX client's perspective until MVS A was restarted.

Part 2 of Figure 40 on page 136 : illustrates that when an outage of MVS A occurs, ARM automatically moves the NFS server to MVS B. The NFS Server on MVS B acquires the DVIPA 123.456.11.22. From the AIX client's perspective the mounted file systems become available once the NFS server has successfully restarted on MVS B. The original hard mount persists.

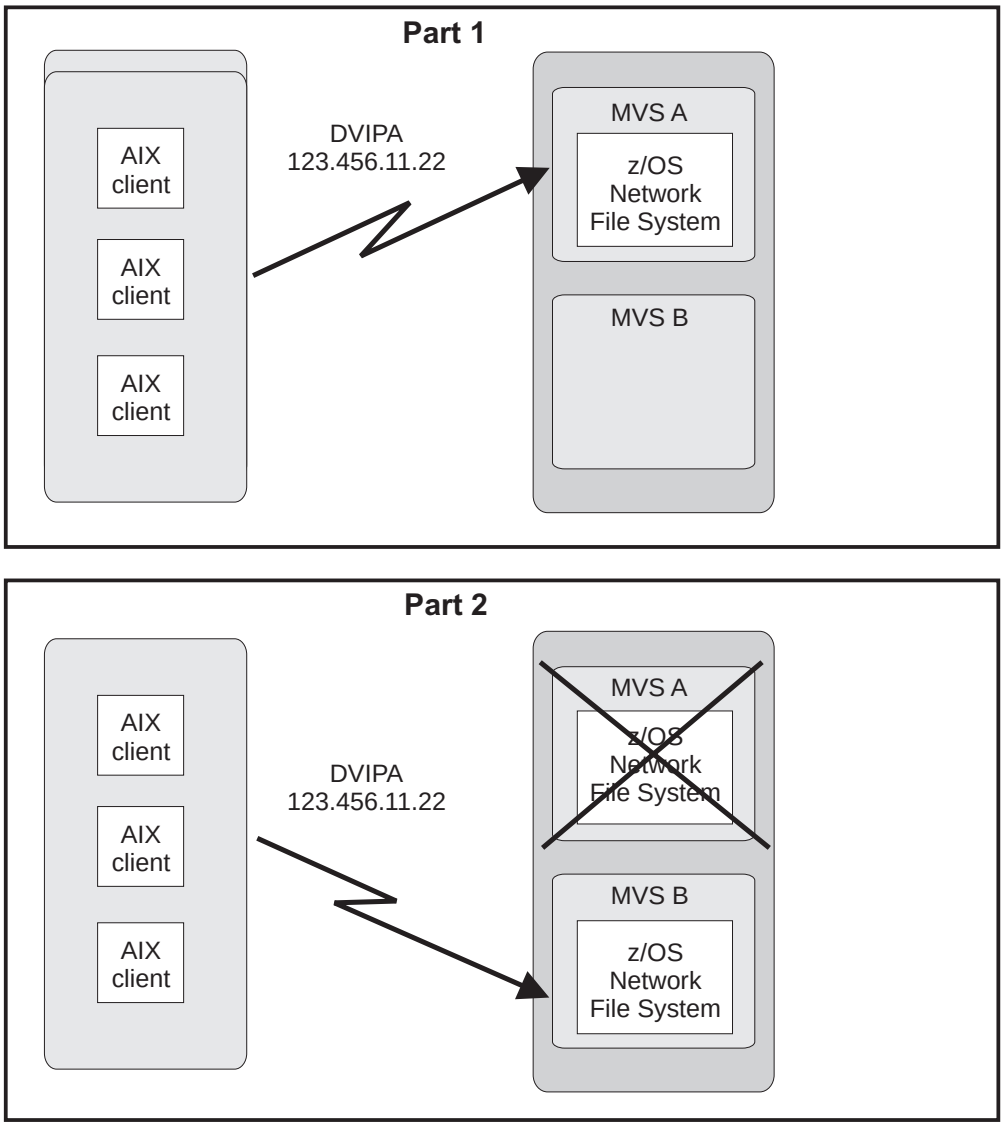


Figure 40. NFS configuration

Note: An ARM enabled NFS will not automatically move back to MVS A after MVS A recovers.

Step for setting up our NFS environment

We performed the following steps to set up our NFS environment for ARM and DVIPA:

1. Acquiring dynamic VIPA:

   We added the following statement in the TCP/IP profiles for MVSA and MVSB to allow NFS to acquire dynamic VIPA:

   ```
   VIPARANGE DEFINE 255.255.255.255 123.456.11.22 ; NFS VIPA
   ```

   We recycled TCPIP on MVSA and MVSB to activate the above changes.

   Note: You could also use the `VARY TCPIP,,OBEYFILE` command with a data set that contains VIPARANGE statement.

   ---

2. Defining the NFS element:

We added the following statement to our ARM policy member (ARMPOLxx) in SYS.PARMLIB member to define the NFS element:

```
RESTART_GROUP(NFSGRP)
  TARGET_SYSTEM(MVSB)
  FREE_CSA(600,600)
    ELEMENT(NFSSELEM)
      RESTART_ATTEMPTS(3,300)
      RESTART_TIMEOUT(900)
      READY_TIMEOUT(900)
```

---

3. Loading the ARM policy:

   We ran the IXCMIAPU utility to load ARMPOLxx and then activated the policy:

   ```
   setxcf start,policy,type=arm,polname=armpolxx
   ```

---

4. Registering NFS using an ARM policy:

   We used ARMWRAP, the ARM JCL Wrapper with the following parameters to register NFS as ARM element:

   ```
   //*******************************************************************
   //*REGISTER ELEMENT 'NFSSELEM' ELEMENT TYPE 'SYSTCPIP' WITH ARM
   //*REQUIRES ACCESS TO SAF FACILITY IXCARM.SYSTCPIP.NFSSELEM
   //ARMREG EXEC PGM=ARMWRAP,
   //       PARM=('REQUEST=REGISTER,READYBYMSG=N,',
   //             'TERMTYPE=ALLTERM,ELEMENT=NFSSELEM,',
   //             'ELEMTYPE=SYSTCPIP')
   //* ---------------------------------------------------------------- *
   //* DELETE VIPA FOR NFS SERVER                                       *
   //* ---------------------------------------------------------------- *
   //DELVIPA  EXEC PGM=EZBXFDVP,
   //            PARM='POSIX(ON) ALL31(ON) /-p TCPIP -d &VIPA'
   //SYSPRINT DD SYSOUT=*
   //* ---------------------------------------------------------------- *
   //* ACQUIRE VIPA FOR NFS SERVER                                      *
   //* ---------------------------------------------------------------- *
   //DEFVIPA  EXEC PGM=EZBXFDVP,
   //            PARM='POSIX(ON) ALL31(ON) /-p TCPIP -c &VIPA'
   //SYSPRINT DD SYSOUT=*
   ```

---

5. Terminating the address space:

   The following example shows what is executed when the address space is terminated:

   ```
   //* ---------------------------------------------------------------- *
   //* DELETE VIPA FOR NFS SERVER                                       *
   //* ---------------------------------------------------------------- *
   //DELVIPA  EXEC PGM=EZBXFDVP,
   //            PARM='POSIX(ON) ALL31(ON) /-p TCPIP -d &VIPA'
   //SYSPRINT DD SYSOUT=*
   //*******************************************************************
   //*FOR NORMAL TERMINATION,DEREGISTER FROM ARM
   //*FOR NORMAL TERMINATION,DEREGISTER FROM ARM
   //*******************************************
   //ARMDREG EXEC PGM=ARMWRAP,
   //        PARM=('REQUEST=DEREGISTER')
   ```

Chapter 12. Using z/OS UNIX System Services

In this chapter, we cover the following z/OS UNIX System Services topics:
- "z/OS UNIX enhancements in z/OS V1R7"
- "Using the hierarchical file system (HFS)" on page 160
- "Automount enhancement for HFS to zSeries file system (zFS) migration" on page 160
- "Using the zSeries file system (zFS)" on page 161

z/OS UNIX enhancements in z/OS V1R7

z/OS UNIX made several enhancements in z/OS V1R7. In this section, we cover the following topics:
- "z/OS UNIX System Services: 64 MB Maximum for OMVS ctrace Buffer"
- "z/OS UNIX System Services: Dynamic Service Activation" on page 140
- "z/OS UNIX System Services: Display Local AF_UNIX Sockets" on page 144
- "z/OS UNIX System Services: /dev/zero, /dev/random, dev/urandom" on page 146
- "z/OS UNIX System Services: Display Information About Move or Mount Failures" on page 147
- "z/OS UNIX System Services: SETOMVS Enhancements" on page 149
- "z/OS UNIX System Services: Display Mount Latch Contention Information" on page 150
- "z/OS UNIX System Services: Enhancements to display file systems" on page 153
- "z/OS UNIX System Services: ISHELL Enhancements" on page 153

z/OS UNIX System Services: 64 MB Maximum for OMVS ctrace Buffer

In z/OS V1R7, to increase the likelihood of capturing valuable trace data, the MAX setting of SYSOMVS CTRACE will be 64 MB. (Prior to z/OS V1R7, the MAX setting for the SYSOMVS component ctrace is 8 MB.) You can set this new value through the SYS1.PARMLIB member CTIBPXxx or through the **TRACE CT** command. The following is an example of setting the SYSOMVS CTRACE to 64 MB through the parmlib CTIBPXxx member:

We used CTRACE parmlib member, CTIBPX64, that defines the 64 MB setting in the parmlib dataset. The following is an excerpt from member CTIBPX64:

```
BUFSIZE(64M)
```

We modified our BPXPRM00 member in the parmlib dataset to use the new CTRACE member. The following is an excerpt from member BPXPRM00:

```
CTRACE(CTIBPX64)
```

We verified the syntax with the following:

```
SETOMVS SYNTAXCHECK=(00)
```

After IPL (and/or OMVS recycle/reinit), we verified the trace buffer setting. For example:

```
D TRACE,COMP=SYSOMVS
IEE843I 10.16.55  TRACE DISPLAY
        SYSTEM STATUS INFORMATION
ST=(ON,0500K,04500K) AS=ON  BR=OFF EX=ON  MT=(ON,140K)
  COMPONENT     MODE BUFFER HEAD SUBS
 ------------------------------------------------------
```

```
  SYSOMVS       ON   0064M
     ASIDS      *NONE*
     JOBNAMES   *NONE*
     OPTIONS    ALL
     WRITER     *NONE
```

The following is an example of setting the SYSOMVS CTRACE to 64 MB with the **TRACE CT** console command:

```
TRACE CT,64M,COMP=SYSOMVS

*0046 ITT006A SPECIFY OPERAND(S) FOR TRACE CT COMMAND.
-46,END

 IEE600I REPLY TO 0046 IS;END
 ITT038I ALL OF THE TRANSACTIONS REQUESTED VIA THE TRACE CT COMMAND WERE SUCCESSFULLY EXECUTED.
 IEE839I ST=(ON,0500K,05000K) AS=ON  BR=OFF EX=ON  MT=(ON,140K)
         ISSUE DISPLAY TRACE CMD FOR SYSTEM AND COMPONENT TRACE STATUS
         ISSUE DISPLAY TRACE,TT CMD FOR TRANSACTION TRACE STATUS

D TRACE,COMP=SYSOMVS
IEE843I 11.43.11  TRACE DISPLAY 286
        SYSTEM STATUS INFORMATION
 ST=(ON,0500K,05000K) AS=ON  BR=OFF EX=ON  MT=(ON,140K)
  COMPONENT     MODE BUFFER HEAD SUBS
 ---------------------------------------------------------------
  SYSOMVS       ON   0064M
     ASIDS      *NONE*
     JOBNAMES   *NONE*
     OPTIONS    ALL
     WRITER     *NONE*
```

z/OS UNIX System Services: Dynamic Service Activation

With z/OS V1R7 or higher systems, some UNIX System Services service items can be activated dynamically without requiring an IPL. Only those APARs, PTFs, or USERMODs marked with ++HOLD REASON(DYNACT) data can use dynamic activation. The instructions must be followed with the ++HOLD documentation. For example, some fixes may require an OMVS (or another component) shutdown or the Dynamic LPA add of BPXINLPA and its ALIASes. The identification of the service libraries can be statements in the BPXPRMxx member, or set with the **SETOMVS** or **SET OMVS** console commands. These datasets must be APF authorized. The dataset names can be retrieved with ″D OMVS,O″ or the get_system_settings() C/C++ API.

Take care identifying the libraries designated for the dynamic activation. The definition requires you to enter the volume where they reside. If the dataset names are moved to other volumes, and the definition is not changed through **SETOMVS**, **SET OMVS**, and/or in member BPXPRMxx, errors can occur. You can set these new statements with **SET OMVS** or **SETOMVS**, for different target service libraries, for any given **F OMVS,ACTIVATE=SERVICE** command invocation. The complete set of dynamic activate fixes within the libraries will be activated. The set will be activated, deactivated, or displayed, depending on the command used.

You should also stay current with UNIX System Services Component maintenance, so the system will be at a high enough level to accept the service.

We recommend that you install these service items into a separate load library from the LPALIB or LINKLIB libraries that are used for your normal install process. This would then be the load library that is regularly used to dynamically activate service on your systems. Selective items can then be copied into them for activation, if all modules are known. However, this is not intended to be used as a way to activate a large set of maintenance for preventive purposes.

BPXPRMxx pertinent statement definitions:

SERV_LPALIB('dsname','volser')
Specifies the target service library where the UNIX System Services modules that are normally built into LPA are located. Value Range:dsname is a 1-to-44 character value representing a valid MVS load library data set name. The alphabetic characters in the load library name must be uppercase. volser is a 1-to-6 character value representing a valid volume serial number for the volume that contains the specified MVS load library. The alphabetic characters in the volume serial number must be uppercase. You can change the value of SERV_LPALIB dynamically using the SETOMVS or SET OMVS command. To make a permanent change, edit the BPXPRMxx member that will be used for future IPLs.

SERV_LINKLIB('dsname', 'volser')
Specifies the target service library where the UNIX System Services modules that are normally loaded from SYS1.LINKLIB into the private area of the OMVS address space are located. Value Range:dsname is a 1-to-44 character value representing a valid MVS load library data set name. The alphabetic characters in the load library name must be uppercase. volser is a 1-to-6 character value representing a valid volume serial number for the volume that contains the specified MVS load library. The alphabetic characters in the volume serial number must be uppercase. You can change the value of SERV_LINKLIB dynamically using the **SETOMVS** or **SET OMVS** command. To make a permanent change, edit the BPXPRMxx member that will be used for future IPLs.

You will encounter a similar message if you supply the wrong volume.:

```
BPXI074I LOAD LIBRARY loadlib IS NOT ON THE SPECIFIED VOLUME voln
```

If a load library is not APF-authorized, the message issued will be one of the following:

```
BPXM059I ACTIVATE=SERVICE REQUEST FAILED,
 LPALIB LIBRARY NOT APF AUTHORIZED
```

or

```
BPXM059I ACTIVATE=SERVICE REQUEST FAILED,
 LINKLIB LIBRARY NOT APF AUTHORIZED
```

The following is an example of setting the libraries using the SETOMVS console command:

```
SETOMVS SERV_LINKLIB=('D10PET.USS.SERV.LINKLIB','SP0004')
 BPXO015I THE SETOMVS COMMAND WAS SUCCESSFUL.

SETOMVS SERV_LPALIB=('D10PET.USS.SERV.LINKLIB','SP0004')
BPXO015I THE SETOMVS COMMAND WAS SUCCESSFUL.
```

The following is an example of setting the libraries using the **SET OMVS** console command. We added the following to our main parm member, SYS1.PARMLIB(BPXPRM00):

```
SERV_LPALIB('D10PET.USS.SERV.LPALIB','SP0004')
SERV_LINKLIB('D10PET.USS.SERV.LINKLIB','SP0004')
```

We verified the syntax with the following:

```
SETOMVS SYNTAXCHECK=(00)
```

The **SET OMVS** command was used to activate the BPXPRM00 new statements. The statements are now in the BPXPRM00 member, which we use upon OMVS initialization. An IPL also validates these settings. However, you should ensure the libraries exist on the volume(s) specified or errors may occur during IPL or SET operations.

```
SET OMVS=(00)
 BPXO032I THE SET OMVS COMMAND WAS SUCCESSFUL.
```

Here is an excerpt from the OMVS Options display:

```
D OMVS,O
...
SERV_LINKLIB    = D10PET.USS.SERV.LINKLIB   SP0004
SERV_LPALIB     = D10PET.USS.SERV.LPALIB   SP0004
...
```

To display and activate the service from the dynamic activate datasets, enter the following. This has to be executed on each system you want the service activated. The amount of ECSA and OMVS address space storage consumed for those service items is also indicated. If a fix capable of dynamic activation is found that cannot be activated due to back-level service found on the active system or missing parts in the target activation libraries, the activation will fail.

```
F OMVS,ACTIVATE=SERVICE
```

This activates service at any time, first displaying the service that can be activated, and prompts users with a WTOR message (reply ″Y″ to continue). Activations should remain in effect across OMVS shutdown. And, these can be performed during a shutdown.

To deactivate the last set of activated service, enter the following. This has to be executed on each system from which you want the service deactivated:

```
F OMVS,DEACTIVATE=SERVICE
```

This deactivates (backs-off) the last set of service items, and prompts users with a WTOR message (reply ″Y″ to continue).

Note: The amount of storage consumed will not decrease when a deactivation is done, the new modules must remain in storage indefinitely.

To display sets of items that have been dynamically activated, enter the following. This has to be executed on each system you want to display these items:

```
D OMVS,ACTIVATE=SERVICE
```

This displays (most recent to oldest) all sets of service items that are currently activated dynamically.

Note: BPXEKDA can be used to retrieve this information

It also reports the library and the volume where each set of fixes were activated from, along with the amount of ECSA and OMVS address space storage that is being consumed.

The following are examples of the ++HOLD REASON(DYNACT) information:

```
++ HOLD(UA20094) SYS FMID(HBB7720) REASON(DYNACT) DATE(05209)
   COMMENT
  (*************************************************************
  * FUNCTION AFFECTED: Unix System Services           (OA11339) *
```

```
*                     Kernel                                     *
*****************************************************************
* DESCRIPTION      : Dynamic PTF Activation without an IPL     *
*****************************************************************
* TIMING           : Post-APPLY                                *
*****************************************************************
* To activate this fix you may IPL your system; or, to         *
* avoid an IPL, you may activate this fix dynamically by       *
* taking the following steps:                                  *
*****************************************************************
*                                                              *
* SET-UP REQUIRED TO ACTIVATE SERVICE:                         *
*  IDENTIFY CURRENT SERVICE LEVEL OF ACTIVE SYSTEM:            *
*   The system has to be at least at the level listed here:    *
*       HBB7720 - BASE                                         *
*                                                              *
*  IDENTIFY TARGET LIBRARIES:                                  *
*   Ensure Unix System Services recognizes the target          *
*   libraries where the PTFs are installed.  The libraries     *
*   are specified on the SERV_LPALIB and SERV_LINKLIB          *
*   parameters defined in the BPXPRMxx parmlib member (set     *
*   at initialization or set/changed via the SETOMVS or        *
*   SET OMVS command).                                         *
*_____________________________________________________________*
*                                                              *
* RESOURCES REQUIRED TO ACTIVATE SERVICE:                      *
*  ECSA and/or OMVS private storage.                           *
*  Required amount of storage will be identified when the      *
*  command to activate service is issued.                      *
*_____________________________________________________________*
*                                                              *
* STEPS TO ACTIVATE SERVICE:                                   *
*  Activate service by issuing the following console command:  *
*    F OMVS,ACTIVATE=SERVICE                                   *
*  A message is displayed with a list of service items to be   *
*  activated and the amount of system resources consumed       *
*  (ECSA and OMVS private storage).                            *
*                                                              *
*  The operator will be prompted with a WTOR asking to         *
*  confirm the activation of service.  The operator must       *
*  respond 'Y' to activate service.                            *
*                                                              *
*  If all service was successfully activated, message BPXM062I *
*  is issued:                                                  *
*  BPXM062I ACTIVATE=SERVICE REQUEST COMPLETED SUCCESSFULLY    *
*                                                              *
*  If service was unable to be activated, either message       *
*  BPXM059I or message BPXM060I is issued.                     *
*_____________________________________________________________*
*                                                              *
* STEPS TO DE-ACTIVATE SERVICE:                                *
*  To deactivate service, issuing the following command:       *
*   F OMVS,DEACTIVATE=SERVICE                                  *
*   No additional resources are used.  However, no resources   *
*   are freed.                                                 *
*****************************************************************)

++ HOLD(UA20084) SYS FMID(HBB7720) REASON(DYNACT) DATE(05208)
   COMMENT
  (*****************************************************************
  * FUNCTION AFFECTED: Unix System Services            (OA12734) *
  *                     Kernel                                    *
  *****************************************************************
  * DESCRIPTION      : Dynamic PTF Activation without an IPL     *
  *****************************************************************
  * TIMING           : Post-APPLY                                *
  *****************************************************************
```

```
* To activate this fix PTF you may IPL your system; or, to     *
* avoid an IPL, you may activate this fix dynamically by       *
* taking the following steps:                                  *
****************************************************************
*                                                              *
* SET-UP REQUIRED TO ACTIVATE SERVICE:                         *
*  IDENTIFY CURRENT SERVICE LEVEL OF ACTIVE SYSTEM:            *
*   The system has to be at least at the level listed here:    *
*       HBB7720 - BASE                                         *
*                                                              *
*  IDENTIFY TARGET LIBRARIES:                                  *
*   Ensure Unix System Services recognizes the target          *
*   libraries where the PTFs are installed.  The libraries     *
*   are specified on the SERV_LPALIB and SERV_LINKLIB          *
*   parameters defined in the BPXPRMxx parmlib member (set     *
*   at initialization or set/changed via the SETOMVS or        *
*   SET OMVS command).                                         *
*--------------------------------------------------------------*
*                                                              *
* RESOURCES REQUIRED TO ACTIVATE SERVICE:                      *
*  ECSA and/or OMVS private storage.                           *
*  Required amount of storage will be identified when the      *
*  command to activate service is issued.                      *
*--------------------------------------------------------------*
*                                                              *
* STEPS TO ACTIVATE SERVICE:                                   *
*  Shutdown OMVS by issuing the following console command:     *
*   F OMVS,SHUTDOWN                                            *
*                                                              *
*  Activate service by issuing the following console command:  *
*   F OMVS,ACTIVATE=SERVICE                                    *
*                                                              *
*   A message is displayed with a list of service items to be  *
*   activated and the amount of system resources consumed      *
*   (ECSA and OMVS private storage).                           *
*                                                              *
*  The operator will be prompted with a WTOR asking to         *
*  confirm the activation of service.  The operator must       *
*  respond 'Y' to activate service.                            *
*                                                              *
*  If all service was successfully activated, message BPXM062I *
*  is issued:                                                  *
*  BPXM062I ACTIVATE=SERVICE REQUEST COMPLETED SUCCESSFULLY    *
*                                                              *
*  Restart OMVS by issuing the following console command:      *
*   F OMVS,RESTART                                             *
*                                                              *
*  If service was unable to be activated, one or more of the   *
*  following messages may be issued:                           *
*  BPXM059I BPXM060I BPXM064I                                  *
*--------------------------------------------------------------*
*                                                              *
* STEPS TO DE-ACTIVATE SERVICE:                                *
*  To deactivate service, after issuing the F OMVS,SHUTDOWN    *
*  command, issue the following commands:                      *
*   F OMVS,DEACTIVATE=SERVICE                                  *
*   F OMVS,RESTART                                             *
*   No additional resources are used.  However, no resources   *
*   are freed.                                                 *
****************************************************************).
```

z/OS UNIX System Services: Display Local AF_UNIX Sockets

In z/OS V1R7 the **DISPLAY OMVS** command was enhanced with a Sockets option that displays information about AF_UNIX Sockets and their sessions. It is used to

display how AF_UNIX socket programs are performing or what sessions have been established. This display for AF_UNIX sockets is similar to the netstat display for AF_INET sockets.

Format:
```
 DISPLAY OMVS,Sockets|So
```

Displays the following information about each AF_UNIX socket:

jobname
The job name of the process that owns the socket.

id The inode number of the socket, in hexadecimal.

peerid The inode number of a connected socket's peer socket.

state The socket state, which is one of the following:

LISTEN
A server TCP stream socket that accepts connections.

DGRAM
A UDP datagram socket.

ACP An accepted stream socket.

CONN A connected stream socket.

STRM An unconnected stream socket.

readbyte
The number of bytes read on this socket, in hexadecimal. For a server socket, this value is the number of connections that have been accepted. After 4 GB, this value wraps.

writebyte
The number of bytes written on this socket, in hexadecimal. After 4GB, this value wraps.

socket name: *socketname*
The name to which this socket was bound, if any.

peer name: *peersocketname*
The name of the socket this socket is connected to, if it is connected and if the peer socket has a name.

The following is an example of the display output:

D OMVS,SO

```
BPXO060I 20.59.18 DISPLAY OMVS 661
OMVS     0010 ACTIVE          OMVS=(00,Z3)
JOBNAME     ID    PEER ID  STATE    READ   WRITTEN
-------- -------- -------- ------ -------- --------
FTPJOB   0000A3DD 00000002 DGRAM  00000000 00000000
   PEER NAME: /dev/log
WT2SR0A  00002575          CONN   00000000 00000000
TCPIP    0000000E 00000002 DGRAM  00000000 000000A6
   PEER NAME: /dev/log
OSNMPD   0000000D 0000000C ACP    00007BFA 000042FB
 SOCKET NAME: /tmp/dpi_socket
TCPIP    0000000C 0000000D CONN   000042FB 00007BFA
   PEER NAME: /tmp/dpi_socket
OSNMPD   0000000B 0000000A ACP    0000005C 0000002E
 SOCKET NAME: /tmp/dpi_socket
OMPROUTE 0000000A 0000000B CONN   0000002E 0000005C
   PEER NAME: /tmp/dpi_socket
```

```
OSNMPD   00000009          LISTEN 00000002 00000000
 SOCKET NAME: /tmp/dpi_socket
OMPROUTE 00000008 00000002 DGRAM  00000000 000090EC
   PEER NAME: /dev/log
FTPD     00000007 00000002 DGRAM  00000000 00000454
   PEER NAME: /dev/log
OSNMPD   00000006 00000002 DGRAM  00000000 00000B09
   PEER NAME: /dev/log
INETD7   00000003 00000002 DGRAM  00000000 00000262
   PEER NAME: /dev/log
SYSLOGD6 00000002          DGRAM  0000A86F 00000000
 SOCKET NAME: /dev/log
```

z/OS UNIX System Services: /dev/zero, /dev/random, dev/urandom

z/OS V1R7 support the /dev/zero, /dev/random, and /dev/urandom character special files. You can continually write to these files and the data will be accepted and discarded. These character special files are created upon IPL or OMVS shutdown/restart, if not present. They are also created, if not present, upon first reference. However, the first reference must be on the specific system, with path name starting with /dev (ie. /dev/zero, /dev/random, and /dev/urandom), or a regular file may be created. Also, referencing these with the system name, in the path name, will not create the character special file, but a regular file (for example, /SYS1/dev/zero, /SYS1/dev/random, and /SYS1/dev/urandom).

Note: Backlevel releases cannot open these devices.

When using the **mknod** command in the shell, the permissions allocated to the files will be affected by the ″umask″ setting. When using the TSO MKNOD command, the files are usually created with the correct permissions. You can use the **chmod** command to correct any permissions.

/dev/zero

/dev/zero is a character special device file that accepts and discards anything written to it and provides binary zeros for any amount read from it. The buffer passed on read() or buffers passed on readv() will be filled with binary zeros for the amount specified to be read. The Return_value for a successful read or write type operation will be equal to the amount of data that was requested to be read or written. There is no ″end of file″ for reads. The supply of zeros is inexhaustible. The /dev/zero file will be created, if necessary, when OMVS starts and will be dynamically created anytime it is referenced by its full name (ie. /dev/zero on the specific system) and it does not exist. The default permissions for /dev/zero will be 666, RW-RW-RW-. These may be changed by the chmod command or function.

The following is an example of using **mknod** in the shell to create the file on the specific system:

```
mknod /dev/zero c 4 1
chmod 666 /dev/zero
```

Note: For device major number 4, the device minor number 0 represents /dev/null and the device minor number 1 represents /dev/zero.

/dev/random and /dev/urandom

Integrated Cryptographic Service Facility (ICSF) is required with either Cryptographic Coprocessor Feature or PCI X Cryptographic Coprocessor depending on the model of the zSeries server. /dev/random and /dev/urandom are character-special device files that generate random numbers. They are opened and read from like any other file. Various applications use this output for creating security keys and other cryptographic purposes. The source of these random

numbers will be the native Cryptographic hardware available on zSeries machines and this will be accessed through the Random Number Generator callable service of ICSF.

The /dev/random and /dev/urandom Character Special devices provide cryptographically secure random output generated from the hardware cryptographic feature available on the zSeries. The foundation of this random number generator is a time variant input with a very low probability of recycling. These device files require ICSF and either Cryptographic Coprocessor Feature or PCI X Cryptographic Coprocessor depending on the model of the zSeries server.

On some Unix systems, /dev/random may block waiting for naturally occurring randomness to occur and /dev/urandom is an alternative, less secure but non blocking, random number generator. On z/OS both of these devices are the same; they rely on the hardware to provide the random numbers and they will not block. The hardware is designed to produce eight-byte random numbers but on a read operation any amount of data requested will be provided and the Return_value for a successful read operation will be equal to the amount of data that was requested.

Reads may fail if ICSF or the hardware is not available or if any addresses passed are invalid. Data written to these devices is ignored without being referenced and the Return_value for a write type operation is equal to the amount of data that was being written. There is no ″end of file″ for reads; the supply of random data is inexhaustible. Reads and writes will not block. These devices are created, if necessary, when OMVS starts and are dynamically created anytime they are referenced by full name and do not exist (that is, /dev/random or /dev/urandom on the specific system). The default permissions are 666, RW-RW-RW-. These may be changed by the **chmod** command or function, or by explicitly defining the devices with **mknod**. Additionally, you must be RACF permitted to the CSFRNG profile in the CSFSERV security class or ICSF must have been started with the CHECKAUTH(NO) option.

The following are examples of using **mknod** in the shell to create the files.

```
mknod /dev/random c 4 2
chmod 666 /dev/random

mknod /dev/urandom c 4 2
chmod 666 /dev/urandom
```

Note: For device major number 4, the device minor number 0 represents /dev/null, the device minor number 1 represents /dev/zero, and the device minor number 2 is for /dev/random and /dev/urandom.

z/OS UNIX System Services: Display Information About Move or Mount Failures

The **D OMVS,MF** command enables you to display failures from prior mount or move file system commands (such as, TSO, Ishell, OMVS, and BPXPRMxx mounts). When executed, this command returns message BPXO058I, which lists previous errors. Here are three ways to use the **D OMVS,MF** command:

```
D OMVS,MF - Returns BPXO058I message, listing up to 10 prior errors
D OMVS,MF=ALL | A - Returns BPXO058I message, listing up to 50 prior errors
D OMVS,MF=PURGE | P  - Purges the saved failure information listed in BPXO058I message
```

The following are some examples of the **D OMVS,MF** command:

- No prior failures:

```
D OMVS,MF
BPXO058I 12.33.17 DISPLAY OMVS 773
OMVS     0010 ACTIVE          OMVS=(00,TP)
NO MOUNT OR MOVE FAILURES TO DISPLAY
```

- The short list of failures. This sample displays two failures; the list can hold up to 10 failures:

```
D OMVS,MF
BPXO058I 12.33.30 DISPLAY OMVS 542
OMVS     0010 ACTIVE          OMVS=(00,JC)
SHORT LIST OF FAILURES:
TIME=08.52.51  DATE=2005/09/27        MOUNT RC=0079  RSN=055B005B
NAME=fileSystemName
  TYPE=fileSystemType
  PATH=mountpoint
TIME=11.37.43  DATE=2005/09/27        MOUNT RC=0079  RSN=055B005C
NAME=fileSystemName
  TYPE=fileSystemType
  PATH=mountpoint
```

- The entire list of failures. This sample displays two failures; the list can hold up to 50 failures:

```
D OMVS,MF=ALL
BPXO058I 12.34.03 DISPLAY OMVS 361
OMVS     0010 ACTIVE          OMVS=(00,J8)
ENTIRE LIST OF FAILURES:
TIME=08.52.51  DATE=2005/09/27        MOUNT RC=0079  RSN=055B005B
  NAME=fileSystemName
  TYPE=fileSystemType
  PATH=mountpoint
TIME=11.37.43  DATE=2005/09/27        MOUNT RC=0079  RSN=055B005C
NAME=fileSystemName
TYPE=fileSystemType
  PATH=mountpoint
```

- Purging the entire list of failures:

```
D OMVS,MF=PURGE
BPXO058I 12.34.19 DISPLAY OMVS 403
OMVS     0010 ACTIVE          OMVS=(00,J8)
PURGE COMPLETE: TIME=12.34.19  DATE=2005/09/27
```

- Running the display command after purging the saved failures. Notice that there is a new line of output, LAST PURGE:, that was not present when running **D OMVS,MF** before running the **D OMVS,MF=PURGE** command for the first time.

```
BPXO058I 12.34.29 DISPLAY OMVS 192
OMVS     0010 ACTIVE          OMVS=(00,JA)
LAST PURGE: TIME=12.34.19  DATE=2005/09/27
NO MOUNT OR MOVE FAILURES TO DISPLAY
```

Note that during BPXPRMxx parmlib member processing, any mount statement that duplicates a mounted file system in both FILESYSTEM name and MOUNTPOINT is silently ignored and is not considered an error. Therefore, it will not show up in BPXO058I. On the other hand, if you try to mount a mounted file system at the mountpoint on which it is already mounted, you will receive the following error:

```
RETURN CODE 00000079, REASON CODE 055B005C.  THE MOUNT FAILED FOR FILE SYSTEM filesystemName
```

If you run **D OMVS,MF** after this error, you will receive information similar to the following:

```
D OMVS,MF
BPXO058I 13.03.59 DISPLAY OMVS 007
OMVS     0010 ACTIVE          OMVS=(00,TP)
LAST PURGE: TIME=12.34.19  DATE=2005/09/27
SHORT LIST OF FAILURES:
```

```
TIME=13.03.40  DATE=2005/09/27        MOUNT RC=0079  RSN=055B005C
  NAME=fileSystemName
  TYPE=fileSystemType
  PATH=mountPoint
```

z/OS UNIX System Services: SETOMVS Enhancements

Enhancements introduced to the **SET OMVS** MVS System Command for z/OS V1R7 enable you to change your mount configuration from the console. Now the **SET OMVS** command executes the **ROOT**, **MOUNT**, **FILESYSTYPE**, **SUBFILESYSTYPE** and **NETWORK** commands that are contained in the specified parmlib member.

Processing of the MOUNT statements:

- Each successful MOUNT operation generates a console message.
- Any MOUNT operation that tries mounting a file system that is already mounted at the mount point on which it is already mounted, is silently ignored.
- Any other MOUNT failure causes an error message to be written to the console.

Processing of the FILESYSTYPE, SUBFILESYSTYPE and NETWORK statements is done the same way in which **SETOMVS RESET** does it.

Note that these enhancements are not an issue for migrating from an older level of z/OS. Prior releases of z/OS ″SET OMVS″ simply ignore these statements. Now, z/OS executes them by ignoring those that duplicate what is already configured. For example; if the UNIX System Services parmlib member BPXPRM00 includes the following mount statements:

1. Mount statement 1:

   ```
   MOUNT FILESYSTEM('OMVSSPN.PET1.ZFS') TYPE(ZFS) MODE(RDWR)
   MOUNTPOINT("/pet2") AUTOMOVE(Y)
   ```

2. Mount statement 2:

   ```
   MOUNT FILESYSTEM('OMVSSPN.PET2.ZFS') TYPE(ZFS) MODE(RDWR)
   MOUNTPOINT("/pet2") AUTOMOVE(Y)
   ```

3. Mount statement 3:

   ```
   MOUNT FILESYSTEM('OMVSSPN.PET2.ZFS') TYPE(ZFS) MODE(RDWR)
   MOUNTPOINT("/pet10") AUTOMOVE(Y)
   ```

We run ″SET OMVS=(00)″ on SYS1 and receive:

```
SYS1       2005200 14:25:46.53 USER1      00000200  SET OMVS=(AA)
IEE252I MEMBER BPXPRMAA FOUND IN SYS1.PARMLIB
BPXO032I THE SET OMVS COMMAND WAS SUCCESSFUL.
```

Based on the above, the following occurs:

1. The mount from Mount Statement 1 is successful:

   ```
   BPXF013I FILE SYSTEM OMVSSPN.PET1.ZFS 398
   WAS SUCCESSFULLY MOUNTED.
   ```

2. The mount point specified in Mount Statement 2 has another file system mounted on it (OMVSSPN.PET1.ZFS) due to Mount Statement 1:

   ```
   BPXF236I FILE SYSTEM OMVSSPN.PET2.ZFS 661 WAS NOT MOUNTED.
   THE MOUNT POINT SPECIFIED IN BPXPRM00 ALREADY HAS
   FILE SYSTEM OMVSSPN.PET1.ZFS MOUNTED ON IT.
   ```

3. The mount point specified in Mount Statement 3 does not exist:

   ```
   BPXF008I FILE SYSTEM OMVSSPN.PET2.ZFS 401 WAS NOT MOUNTED.
   THE MOUNT POINT SPECIFIED IN BPXPRM00 DOES NOT EXIST
   ```

z/OS UNIX System Services: Display Mount Latch Contention Information

A new addition to the set of MVS System Commands that display the z/OS UNIX System Services status is **D OMVS,WAITERS | W**. This command is created for better identifying the reason for which a mount latch is being held and similar information for outstanding cross system messages.

Using this command, you can display the following. The display output is for the specific system on which the command was entered:

- The task that is holding the LFS Mount Latch
- The reason why the task started holding the LFS Mount Latch
- What that task is doing
- The tasks waiting for that Mount Latch and why they want it
- The tasks that are currently waiting for messages from other systems in a sysplex.

On the sender systems what the senders are waiting for and the systems they are waiting from are displayed. On the receiver systems, the messages that have arrived and have not yet been responded to are shown.

The output of this command is separated into two different sections: ″Mount Latch Activity″ and ″Outstanding Cross System Messages.″

If there is some ″Mount Latch Activity″ in the system, the related section displays the following:

- Who is holding the Mount Latch
- Who is waiting for the Mount Latch
- What the holder is doing at the moment
- How long the Mount Latch has been held
- How long each waiter has been waiting for the Mount Latch.

If there are some ″Outstanding Cross System Messages″ on the system, the related section displays the following:

- Who is waiting for a reply
- What type of message was sent
- The systems to which the message was sent
- How long the reply has been outstanding.

The following are sample **D OMVS,W** outputs:

Sample 1: If there are no Mount Latch activity or Outstanding Cross System Messages at the time, you will receive something similar to the following:

```
BPXO063I 09.54.31 DISPLAY OMVS 712
OMVS     0010 ACTIVE          OMVS=(00,JE)
MOUNT LATCH ACTIVITY: NONE
OUTSTANDING CROSS SYSTEM MESSAGES: NONE
```

Sample 2: If the system on which you run **D OMVS,W** is waiting on some replies from other systems for messages that it had sent, you will receive something similar to the following:

```
SYSTEM JB0 RESPONSE TO D OMVS,W
BPXO063I 09.54.31 DISPLAY OMVS 255
OMVS     0010 ACTIVE          OMVS=(00,JB)
MOUNT LATCH ACTIVITY: NONE
OUTSTANDING CROSS SYSTEM MESSAGES:
 SENT SYSPLEX MESSAGES:
    USER   ASID   TCB    FCODE  MEMBER   REQID     MSG TYPE     AGE
  U082001  0336 007F8080  0003   Z2     045E5B89   RDWRCall  00.00.00
```

where:

USER User id of the address space that is involved

ASID AsID of the address space that is involved

TCB Task that is involved

FCODE
Function code being sent cross system

MEMBER
The system(s) to which the message is sent

REQID
Unique request ID of this message

MSG TYPE
Function that the messages is performing

AGE How long the task has been waiting

Sample 3: If the system on which you run **D OMVS,W** has received some messages but has not responded to them, then you will see something similar to the following:

```
SYSTEM Z2 RESPONSE TO D OMVS,W
BPXO063I 09.54.31 DISPLAY OMVS 809
OMVS     0010 ACTIVE          OMVS=(00,Z2)
MOUNT LATCH ACTIVITY: NONE
OUTSTANDING CROSS SYSTEM MESSAGES:
 RECEIVED SYSPLEX MESSAGES:
                     FROM  FROM                  FROM
   ON TCB   ASID     TCB     FCODE  MEMBER   REQID     MSG TYPE     AGE
  007D2CF0 0336    007F8080  0003     JB0   045E5B89   RDWRCall   00.00.00
     IS DOING: HFS RDWRCall / Running
    FILE SYSTEM: OMVSSPN.U2.U059048.FS
```

where:

ON TCB
TCB of the Worker Task that is processing this message.

FROM ASID
AsID of the message sender.

FROM TCB
The TCB of the message sender.

FCODE
The function code to be processed.

FROM MEMBER
The sysplex member that sent this message.

REQID
Unique request ID of this message.

MSG TYPE
Function that the message is performing.

AGE How long this Working Task has been processing the message.

IS DOING
What the worker task is actually doing; that is, what is holding the worker task from responding to the message. (Shown only for Worker Tasks that appear to be hung or that are running in a different component than OMVS.)

FILE SYSTEM
The file system involved (if any).

Sample 4: You receive something similar to the following, if you had Mount Latch activity at the time of the display:

```
BPXO063I 03.33.02 DISPLAY OMVS 782
OMVS     0010 ACTIVE          OMVS=(00,JE)
MOUNT LATCH ACTIVITY:
    USER   ASID     TCB           REASON                        AGE
 HOLDER:
  OMVS     0010  008EA400     Inact Cycle                     00.06.22
     IS DOING: XPFS VfsInactCall / XSYS Message To: Z1
     FILE SYSTEM: OMVSSPN.U2.FS
 WAITER(S):
  OMVS     0010  008EA840     FileSys Sync                    00.06.17
OUTSTANDING CROSS SYSTEM MESSAGES: NONE
```

The following are displayed for both the HOLDER and the WAITERs:

USER User id of the address space that is involved

ASID AsID of the address space that is involved

TCB Task that is involved

REASON
What the user is trying to do

AGE How long the task has been waiting for the Mount Latch.

The following are displayed for the HOLDER:

IS DOING
What the holder task is doing

FILE SYSTEM
The name of the file system involved (If any).

Sample 5: Note that Sample 2 is marked as ″System JB0 Response to D OMVS,W″ and Sample 3 is marked as ″System Z2 Response to D OMVS,W″. Then notice that the FROM MEMBER field in Sample 2 shows JB0 and the MEMBER field in Sample 1 shows Z2. Now look at the REQID fields for both samples and notice that they are the same. Sample 2 shows the message that JB0 sent to Z2 and is currently expecting a reply for it. On the other hand, Sample 3 shows that Z2 has received a message from JB0 and has not yet responded to it.

In summary, by running **D OMVS,W** across the sysplex, you can track Outstanding Sysplex Messages as well as Mount Latch Activity, just like Sample 5.

Note: This display is very useful in diagnosing mount latch contention and hangs. See ″Procedure: Diagnosing and resolving mount latch contention″ in *z/OS MVS Diagnosis: Reference* for more information.

z/OS UNIX System Services: Enhancements to display file systems

Using **D OMVS,FILE** | **F**, you can display the list of file systems that z/OS UNIX System Services is currently using along with their status. This is not a new function but it is important to note that beginning with z/OS V1R7 this command displays mounts that are in an earlier stage of the mount sequence than it did before, as well as new status items.

D OMVS,F is a key display command that collects data during z/OS UNIX System Services problems, such as Latch Hangs. The enhancements to this command help with problem determination.

For z/OS V1R7, **D OMVS,F** displays the following status items:

- The status of each file system
- The date and time that the file system was mounted
- The latch number for the file system
- The quiesce latch number for the file system, or 0 if UNIX System Services has never quiesced the file system.

The following are two sample outputs from **D OMVS,F**. In both examples, the following is true:

MOUNTED
Specifies the date and time this file system was mounted

LATCHES
| **L** specifies the latch number for this file system and **Q** specifies the quiesce
| Latch number for this file system

Example 1:

```
TYPENAME  DEVICE ---------STATUS-----------MODE  MOUNTED    LATCHES
ZFS        178            ACTIVE           RDWR  09/26/2005 L=187
   NAME=fileSystemName                             18.56.18   Q=0
   PATH=mountPoint
   OWNER=ownerSystemName      AUTOMOVE=Y  CLIENT=N
```

Example 2:

```
TYPENAME  DEVICE ---------STATUS-----------MODE  MOUNTED    LATCHES
NFS       4511           FORCE UNMOUNT    RDWR  07/21/2005 L=184
  NAME=fileSystemName                              12.24.13   Q=1053
  PATH=mountPoint
  MOUNT PARM=mountParm
  OWNER=ownerSystemName       AUTOMOVE=Y CLIENT=N
```

z/OS UNIX System Services: ISHELL Enhancements

This section lists the ISHELL enhancements for z/OS V1R7.

New “Do not normalize the selected path to the real path” option

Use the new option “Do not normalize the selected path to the real path” to specify the use of Real (normalized) or Logical paths on the file list when using ISHELL. That is, as you navigate through directories using ISHELL, if this option is NOT

selected (default), the displays expand the symbolic links in the pathnames. If you select this option, the displays show the pathnames as you enter them or select them in ISHELL.

You can select this option from the main ISHELL panel. Follow the "Options->Directory List" pull-down menu, and this option is at the very bottom, selected by default.

Here is an example to clarify what this new option does. In this example, we have:
/dir0

and it is a symbolic link to:
/dir1/dir2/dir3/dir4/dir5

In ISHELL, if you navigate to */dir0/dir6/dir7* with "Do not normalize the selected path to the real path" option NOT selected (default), your display shows the following (assuming */dir0/dir6/dir7* is empty):

```
EUID=2406   /dir1/dir2/dir3/dir4/dir5/dir6/dir7/
  Type  Perm  Filename
_ Dir    750  .
_ Dir    750  ..
```

On the other hand, if you navigate to */dir0/dir6/dir7* with "Do not normalize the selected path to the real path" option selected, your display shows the following (assuming */dir0/dir6/dir7* is empty):

```
EUID=2406   /dir0/dir6/dir7/
  Type  Filename
_ Dir   .
_ Dir   ..
```

The New "View and set attributes" Option

You can set this option to Y or /, when creating a new file or a directory. Setting that option Y or / takes you to a dialog box where you can set or change any modifiable attributes for the file you just created.

For example, to create a new file /u/user1/test, enter it at the main panel, as shown in Figure 41 on page 155:

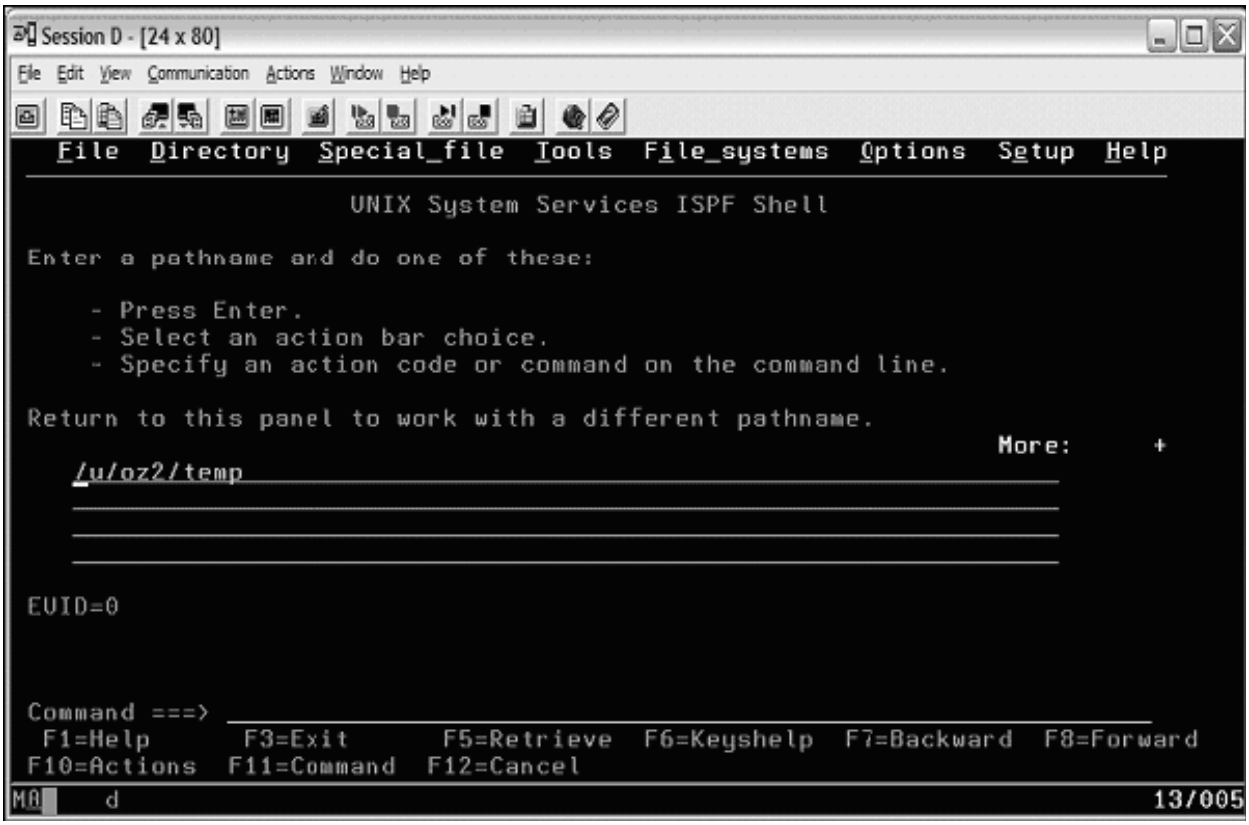


Figure 41. Entering /u/user1/test on the z/OS UNIX System Services main panel

The dialog box shown in Figure 42 on page 156 is displayed:

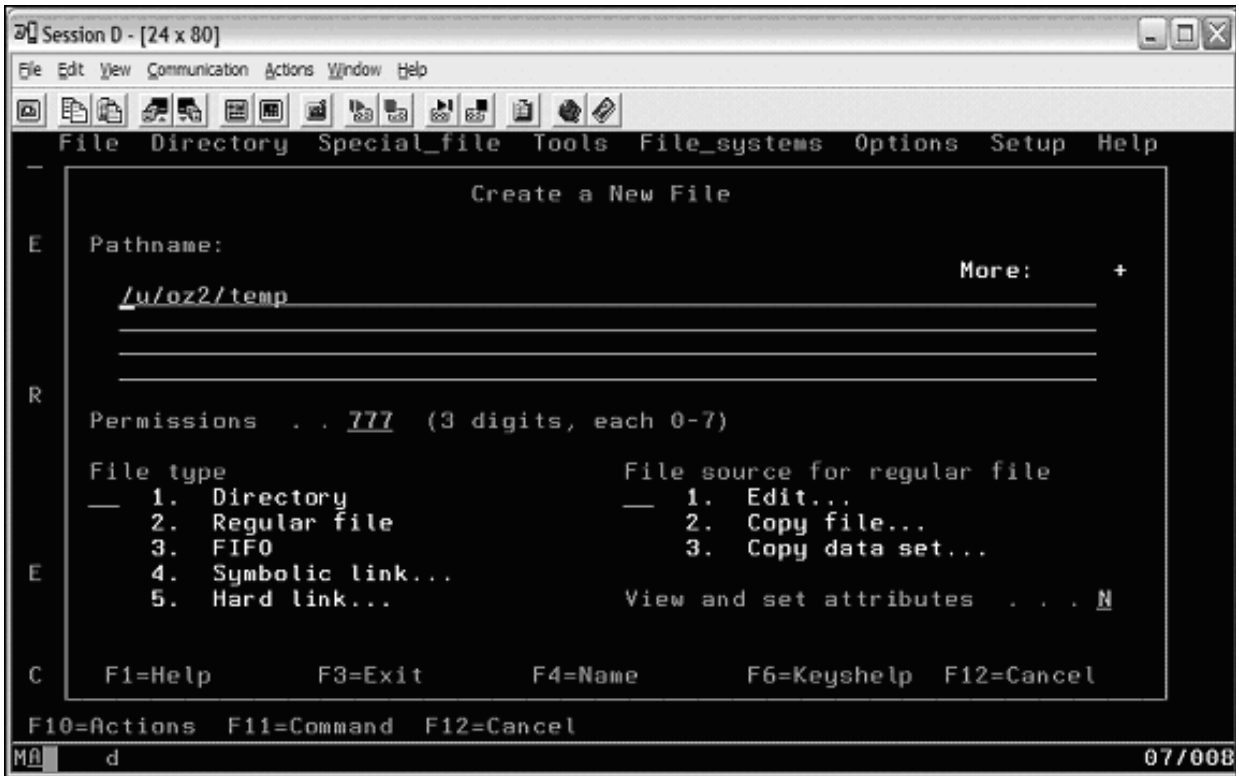


Figure 42. Dialog box for /u/user1/test

In the dialog box, the “View and set attributes” option is set to “N” by default. If you leave that option as is, pick a 2 as “File Type” and press Enter, you are presented with the panel shown in Figure 41 on page 155. On the other hand, if you set that option to “Y” or “/”, pick 2 as the “File Type” and press Enter, the Display File Attributes panel (Figure 43 on page 157) is displayed.

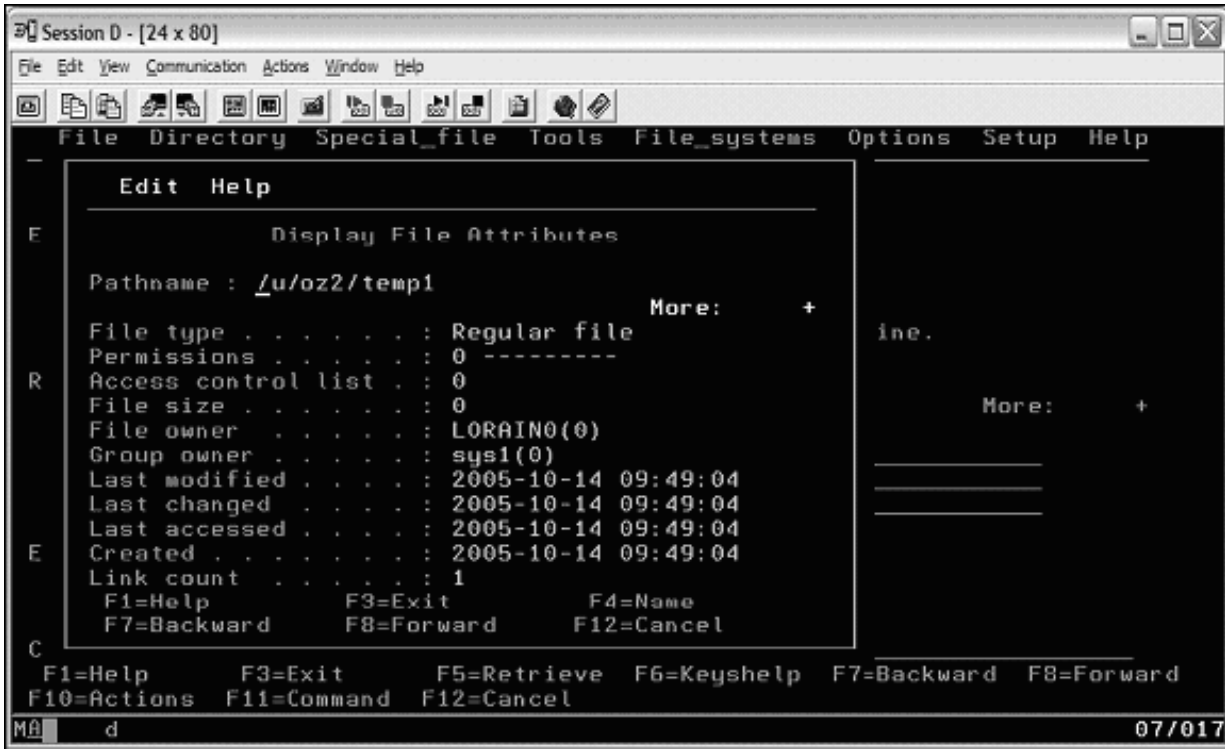


Figure 43. File attributes for /u/user1/test

In this panel we can navigate through all the modifiable options and change or set them for the file we just created.

The New REFRESH Command

When listing directories you can enter the new “refresh | refr” command to cause the display to be refreshed. You may visit the ISHELL Help panels for more details.

The New “GROUP LIST” Choice

A new choice, GROUP LIST, is added to the SETUP pull-down on the main panel. When selected it displays a list of all the groups and their GIDs in a table, as shown in Figure 44 on page 158.

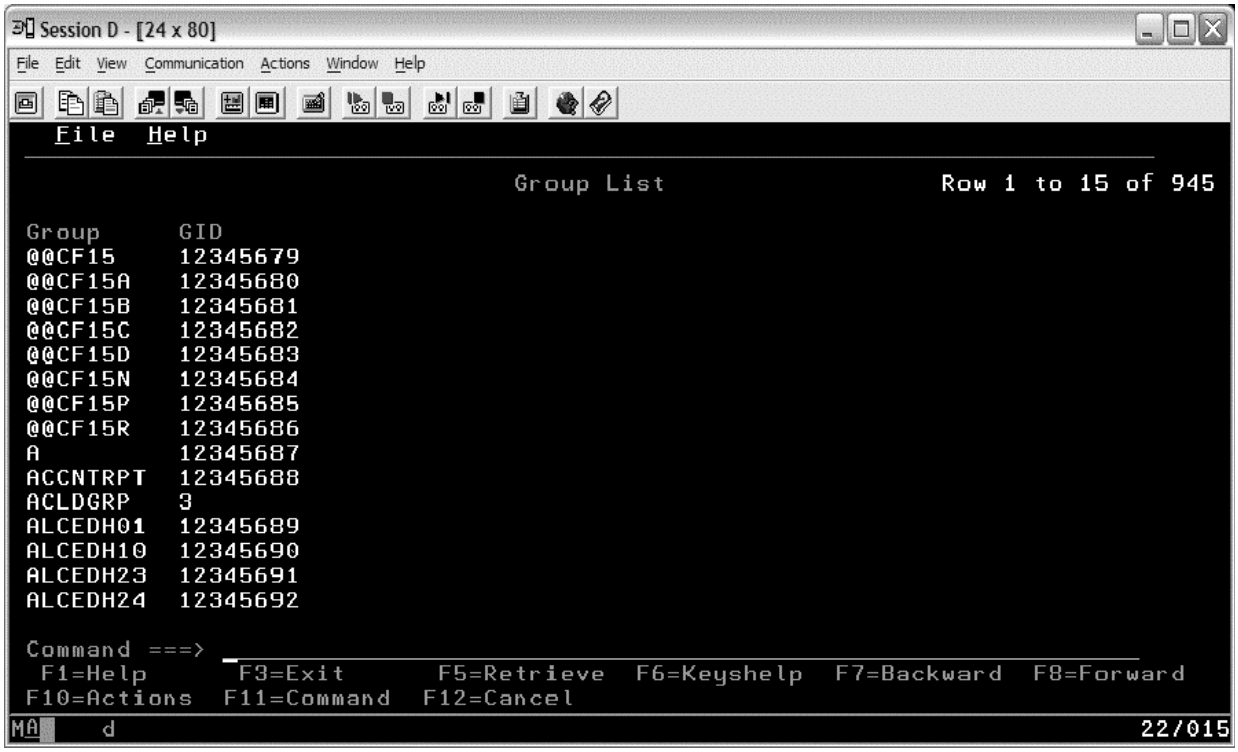


Figure 44. Groups and GIDs

By default the table is sorted by group name but you can sort it by GID. Follow the “File” pull-down and pick “Sort GID.” Figure 45 on page 159 shows a Group List sorted by GID.

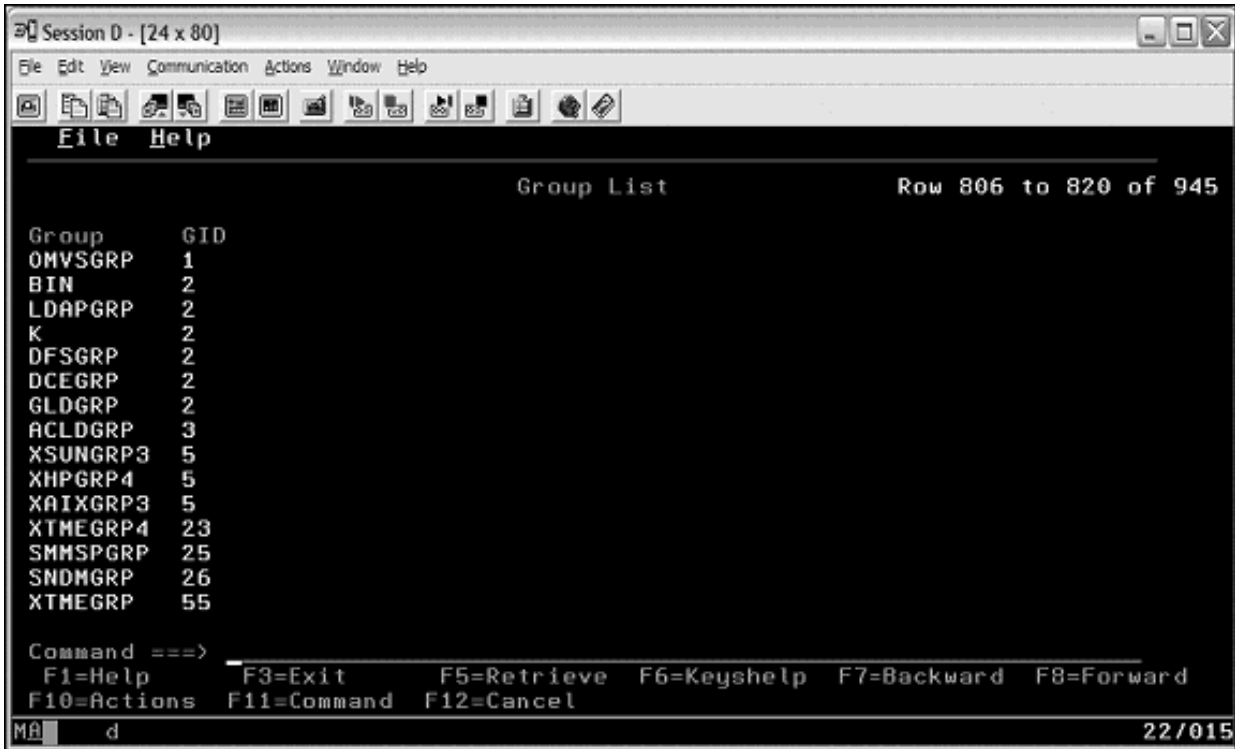


Figure 45. Sorting by GID

See the ISHELL Help panels for more details.

Keeping a List of Recently-Viewed Directories

Another ISHELL enhancement for z/OS V1R7 has the ability to keep a list of recently viewed directories, a history file.

Enabling the directory reference list can be done in either of two ways (if you try to view the reference list without first enabling it, you will receive an error message):

- From the command line, enter **REF ON**
- Using ISHELL menus: Select Options –> Advanced –> Select “Enable directory reference list”

Viewing the directory reference list can be done in either of two ways:

- From the command line, enter **REF**
- Using ISHELL menus: Select Tools –> Reference List (REF)

Disabling the directory reference list can be done in either of two ways:

- From the command line, enter **REF OFF**
- Using ISHELL menus: Select Options –> Advanced –> Deselect “Enable directory reference list”

Refreshing/Clearing the reference list:

- From the command line, enter **REF CLEAR**

Saving the reference list manually (Will be saved automatically when you leave ISHELL):

- From the command line, enter **REF SAVE**

The reference list file for user1 is saved as the following. You should not alter the contents of this file:

```
/u/user1/.ishell-reflist-USER1
```

See the ISHELL Help panels for more details.

Using the hierarchical file system (HFS)

We provided extensive coverage of our strategy for managing the z/OS UNIX hierarchical file system (HFS), including shared HFS, in our December 2001 edition. Refer to that edition for more information.

Automount enhancement for HFS to zSeries file system (zFS) migration

We tested a new automount enhancement that eases the migration from HFS to zFS file systems. Prior to the new function, you could not use a generic automount policy to automount both HFS and zFS file systems - all the file systems had to be the same type for a given automount managed mountpoint. The enhanced HFS to zFS automount migration function allows a single automount policy to mount both HFS and zFS file systems. This will help if you want to migrate your file systems over time rather than all at once, and so have a mixture of HFS and zFS file systems in your installation.

It works like this: the automount function has changed so that when you specify either HFS or ZFS as the file system type in an automount policy, the system re-checks the data set at mount time to determine what type of data set it really is, and then directs the mount to the appropriate file system type. However, to use this function, the naming conventions of the file systems for both HFS and zFS must be the same.

The example below shows a zFS policy that we implemented to mount both HFS and zFS file systems. This policy will mount both pre-existing HFS or zFS type file systems, but only allocates new file systems as zFS file systems. This is the recommended policy for easing the migration to zFS file systems.

```
name *
type ZFS
filesystem OMVSSPN.<uc_name>.FS
lowercase no
allocuser  space(3,2)   cyl    storclas(SMSOE)
mode rdwr
duration 30
delay 10
```

The next automount policy example will mount both pre-existing HFS or zFS file types as well, but will only allocate new file systems as HFS file systems:

```
name *
type HFS
filesystem OMVSSPN.<uc_name>.FS
lowercase no
mode rdwr
allocuser space(3,1) cyl storclas(SMSOE)
duration 30
delay 10
```

Using the zSeries file system (zFS)

We provided extensive coverage of our strategy for setting up and managing a z/OS DFS zSeries file system (zFS) in our December 2003 edition. Refer to that edition for more information.

zFS enhancements in z/OS V1R6

The following topics describe some of the new zFS functions in z/OS V1R6 which we implemented and tested.

- "zFS parmlib search"
- "zFS performance monitoring with zfsadm (query and reset counters)"
- "HANGBREAK, zFS modify console command" on page 164

zFS parmlib search

zFS implemented a new logical parmlib search capability. We tested the following options:

Using IOEPRM00: If an IOEZPRM DD statement for specifying zFS configuration parameters is not in the started proc, the zFS will look in SYS1.PARMLIB for the existence of an IOEPRM00 member. If that member is not found, then zFS uses default settings. We created member IOEPRM00 and populated it with the settings that we use in our sysplex:

```
user_cache_size=256m
debug_setting_dsn=sys1.&SYSNAME..zfs.debug(file1)
trace_dsn=sys1.&SYSNAME..zfs.trace
trace_table_size=128m
```

Specifying IOEPRMxx: Another option is to specify one or more IOEPRMxx members of parmlib to use. The members are identified in the zFS FILESYSTYPE statement of the BPXPRMxx parmlib member. The following example shows how to specify that we want to use members IOEPRM01 and IOEPRM02 for our zFS configuration settings.

```
FILEYSTYPE TYPE(ZFS) ENTRYPOINT(IOEFSCM) ASNAME(ZFS,'SUB=MSTR') PARM('PRM=(01,02)')
```

Using the SYSCLONE symbolic: Another option allows us to have a unique IOEPRMxx for each image by using the SYSCLONE symbolic. The following example illustrates how parmlib members would be selected if we were to start zFS on system Z0. Members IOEPRM97, IOEPRM98, IOEPRM99, and IOEPRMZ0 would be used. If a parmlib member is not found, the search for the configuration option will continue with the next parmlib member.

The maximum number of suffixes for IOEPRMxx that can be specified on the FILESYSTYPE statement is 32.

zFS performance monitoring with zfsadm (query and reset counters)

The **zfsadm query** command displays and resets zFS internal performance statistics counters and timers.

Format:

```
zfsadm query [-locking] [-reset] [-storage] [-usercache] [-iocounts]
             [-iobyaggregate] [-iobydasd] [-level] [-help]
```

Options:

-locking Specifies that the locking statistics report should be displayed.

-reset Specifies the report counters should be reset to zero. Should be specified with a report type.

-storage Specifies that the storage report should be displayed.

-usercache Specifies that the user cache report should be displayed.

-iocounts Specifies that the I/O count report should be displayed.

-iobyaggregate
Specifies that the I/O count by aggregate report should be displayed.

-iobydasd Specifies that the I/O count by Direct Access Storage Device (DASD) report should be displayed.

-level Prints the level of the zfsadm command. This is useful when you are diagnosing a problem. All other valid options specified with this option are ignored.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Note that the **-reset** option will reset the counters AFTER, not before, the display that is displayed when the option is used. For example, **zfsadm query -locking -reset** would display the locking statistics report, then reset the counters. Subsequent locking display will show statistics from counter reset.

Example: `zfsadm query -locking -reset`

Result:

```
Locking Statistics


Untimed sleeps:      13575   Timed Sleeps:          0   Wakeups:      13574

Total waits for locks:              22319606
Average lock wait time:             1.906 (msecs)

Total monitored sleeps:             13522
Average monitored sleep time:       5.692 (msecs)

                Top 15 Most Highly Contended Locks

  Thread     Async       Spin
   Wait      Disp.       Resol.    Pct.      Description
----------  -------   ----------  -----     -------------
 23683110       0      983063    99.583%    Log system map lock
        3   37549           6     0.151%    Volser I/O queue lock
    11852       0       20564     0.130%    Async global device lock
    10321       0         192     0.42%     Vnode-cache access lock
     2113       0        3705     0.23%     Transaction-cache complete list lock

     1134    1116        3425     0.22%     Transaction-cache main lock
     2446       0         187     0.10%     Anode bitmap allocation handle lock

     1405       0         269     0.6%      Anode fileset quota lock
     1437       0           4     0.5%      Async IO device lock
      202     425         531     0.4%      User file cache main segment lock
      829       0          81     0.3%      Anode fileset handle lock
      609       0          29     0.2%      Metadata-cache buffer lock
      352       0         169     0.2%      Anode file zero lock
      420       0          39     0.1%      Anode file notify lock
```

```
      280       0          21     0.1%      Transaction-cache active list lock
Total lock contention of all kinds:   24769331

               Top 5 Most Common Thread Sleeps

   Thread Wait             Pct.             Description
   -----------            -----            -----------
      13521              99.992%           Transaction allocation wait

          1                 0.7%           OSI cache item cleanup wait

          0                 0.0%           Directory Cache Buffer Wait

          0                 0.0%           User file cache Page Wait

          0                 0.0%           User file cache File Wait
```

Example: `zfsadm query -locking`

Result:

```
Locking Statistics
Untimed sleeps:         10   Timed Sleeps:          0   Wakeups:         10

Total waits for locks:              15898
Average lock wait time:             2.174 (msecs)

Total monitored sleeps:                10
Average monitored sleep time:       5.622 (msecs)

               Top 15 Most Highly Contended Locks

  Thread      Async    Spin
   Wait       Disp.    Resol.     Pct.    Description
----------   -------   -------   -----    --------------
     17009       0      679     99.718%  Log system map lock
         0      19        0      0.107%  Volser I/O queue lock
         7       0        7      0.78%   Async global device lock
         6       0        0      0.33%   Vnode-cache access lock
         6       0        0      0.33%   Anode bitmap allocation handle lock

         3       0        0      0.16%   Transaction-cache complete list lock
         1       0        0      0.5%    Vnode lock
         1       0        0      0.5%    Async IO device lock
         0       0        0      0.0%    Async IO set free list lock
         0       0        0      0.0%    Async IO event free list lock
         0       0        0      0.0%    LVM global lock
         0       0        0      0.0%    OSI Global process lock
         0       0        0      0.0%    Main volume syscall lock
         0       0        0      0.0%    User file cache all file lock
         0       0        0      0.0%    User file cache main segment lock

Total lock contention of all kinds:      17738

               Top 5 Most Common Thread Sleeps

Thread Wait     Pct.       Description
-----------   --------     -------------
       10      100.0%      Transaction allocation wait

        0        0.0%      OSI cache item cleanup wait

        0        0.0%      Directory Cache Buffer Wait

        0        0.0%      User file cache Page Wait

        0        0.0%     User file cache File Wait
```

Corresponding pfsctl Application Programming Interface (APIs) are also provided to retrieve these performance statistics.

- **Statistics iobyaggr Information** – The statistics iobyaggr information subcommand call contains information about the number of reads and writes and the number of bytes transferred for each aggregate.
- **Statistics iobydasd Information** – The statistics iobydasd information subcommand call contains information about the number of reads and writes and the number of bytes transferred for each DASD volume.
- **Statistics iocounts Information** – The statistics iocounts information subcommand call contains information about how often zFS performs I/O for various circumstances and how often it waits on that I/O.
- **Statistics Locking Information** – The statistics locking information subcommand call is a performance statistics operation that returns locking information.
- **Statistics Storage Information** – The statistics storage information subcommand call is a performance statistics operation that returns storage information.
- **Statistics User Cache Information** — The statistics user cache information subcommand call is a performance statistics operation that returns user cache information.

For more information on these APIs, see *z/OS Distributed File Service zSeries File System Administration*.

HANGBREAK, zFS modify console command

The following new modify console command for zFS attempts recovery for specific hang conditions: **modify procname,hangbreak**.

The **hangbreak** command causes zFS to post a failure to any requests in zFS that are waiting. This can allow the hang condition to be broken and resolved. This should only be used if you suspect that there is a hang involving zFS. The modify **zfs,query,threads** operator command is used to determine if one or more requestor threads remain in the same wait over several queries. If this command does not successfully break the hang, you need to stop or cancel zFS. If you suspect that zFS is in an infinite loop, you need to cancel zFS.

Example:

```
F ZFS,HANGBREAK
IOEZ00025I zFS kernel: MODIFY command - HANGBREAK completed successfully.
```

zFS: Migrating the Sysplex Root File System from HFS to zFS

We converted our Sysplex-Root file system from an HFS to a zFS. Note that the Sysplex-Root file system is the top-of-the-tree in the UNIX System Services file system hierarchy for sysplex. Therefore, to replace the Sysplex-ROOT file system, we had to unmount all file systems.

We took the following approach to convert our sysplex root from an HFS to a zFS:

1. Make a zFS copy of the sysplex root

   We had the following:

   ```
   HFS sysplex root: OMVSSPN.SYSPLEX.ROOT.FS size: 6 cylinders
   ```

   We created the following:

   ```
   zFS sysplex root: OMVSSPN.SYSPLEX.ROOT.ZFS size: 10 cylinders
   ```

   We created the zFS version of the sysplex root to be larger than the HFS sysplex root. The main reason is that zFS file systems are formatted differently

than HFS file systems and because we were running with ″dynamic grow″ defaulted to OFF to ensure that there was sufficient space for growth. We then mounted the zFS file system (RDWR) at /tempMountPoint.

Note: The sample ″SYS1.SAMPLIB(BPXISYZR)″ can be modified and used to perform the zFS file system creation. It also executes ″BPXISYS1″ which creates needed files. If you do not use this sample, you will have to evaluate if BPXISYS1 needs to be run.

We copied the HFS file system to the zFS file system using copytree:

```
/samples/copytree / /tempMountPoint
```

You should perform the copy when it is known that there is no activity against the file system, and it will not change, after the copy. Because we copied from a known, good sysplex-root file sytem that contains the required links and files, we did not have to run BPXISYS1.

Once copytree completed, we double checked to make sure the copy was successful and immediately unmounted the zFS from /tempMountPoint.

Note: There are other ways of copying an HFS to a zFS. This is the way we did it. Please reference ″Migrating from HFS to zSeries File Systems (zFS) in /OS V1R7" for some other ways. Also see the section, Migrating data from HFS to zFS, in z/OS Distributed File Service zFS Administration.

2. Edit the ROOT statement in the BPXPRMxx member with the new sysplex root name, and change the type to ZFS. We used member BPXPRM00 in SYS1.PARMLIB:

```
ROOT FILESYSTEM('OMVSSPN.SYSPLEX.ROOT.FS') TYPE(HFS)
  MODE(RDWR)
```

with

```
ROOT FILESYSTEM('OMVSSPN.SYSPLEX.ROOT.ZFS') TYPE(ZFS)
  MODE(RDWR)
```

Note: The HFS and zFS file system types in mount statements and command operands are now generic file system types that can mean either HFS or zFS.

3. Bring down all systems in the sysplex but one. You can do this before the copy, also.
4. Unmount all the file systems, after taking down all subsystems and such that were using file systems.

   We ran the **f bpxoinit,filesys=unmountall** modify command to unmount all file systems.

   We ran the **d omvs,f** display command. It should display something similar to this:

```
RESPONSE=SYS1
 BPXO045I 17.44.31 DISPLAY OMVS 592
 OMVS     0010 ACTIVE          OMVS=(00,SYS1)
 TYPENAME   DEVICE ----------STATUS----------- MODE  MOUNTED    LATCHES
 BPXFTCLN 73741826 ACTIVE                          RDWR  07/21/2005      L=11
   NAME=SYSROOT                                  17.43.20    Q=0
   PATH=/
   OWNER= AUTOMOVE=Y CLIENT=N
```

5. Mount the file systems specified in the BPXPRMxx member (this includes the ROOT file system). We used member BPXPRM00 in SYS1.PARMLIB for ROOT identification.

```
SET OMVS=(00)
BPXO032I THE SET OMVS COMMAND WAS SUCCESSFUL.
```

6. Test the system. We did the following:
   - Started an OMVS session and did such tasks as creating files and navigating the session
   - Checked the system log for error messages
   - Ran **d omvs,f** to see if all mounts specified in BPXPRMxx were mounted successfully.
7. Once you have confirmed that all looks good, IPL the rest of the systems in the sysplex, checking each system as it enters the sysplex.

zFS: Improved Mount Performance (Fast-Mount)

Starting in z/OS V1R7, zFS has improved the performance of mounting zFS type file systems. This required a change in the structure and on-disk format of the zFS aggregate. The new structure is referred to as version 1.4. The prior structure was referred to as version 1.3.

New zFS aggregates created in z/OS V1R7 are in version 1.4 format. Existing aggregates are converted to the version 1.4 format automatically (from the version 1.3 format) upon the first read-write (R/W) mount in z/OS V1R7. Since this occurs in addition to the normal mount processing, the first R/W mount of existing aggregates takes as long as they did with previous formats. Subsequent mounts benefit from the new format and mount more quickly.

During the conversion, you may get messages similar to the following:

```
IOEZ00500I Converting aggr_name for fast mount processing
IOEZ00518I Converting filesystem filesystem_name to allow for fast mount
```

Migration/Coexistence Notes

Toleration APAR OA11573 must be installed on prior releases before IPLing z/OS V1R7. Prior releases will then be able to correctly access zFS aggregates with the new version 1.4 structure. Conversion will not take place on prior releases.

If you choose to have zFS aggregates in the version 1.3 format in a mixed z/OS level sysplex, you should mount the version 1.3 aggregates NOAUTOMOVE. Or, you could choose to AUTOMOVE(EXCLUDE) all z/OS V1R7 systems. This prevents movement to a z/OS V1R7 system, where the aggregate is automatically converted to a version 1.4 aggregate upon first R/W mount.

In cases where a version 1.4 aggregate needs to be used on a system that does not have the toleration APAR OA11573 applied, the aggregate must be converted back to version 1.3.

The zFS IOEAGSLV (salvager) utility for z/OS V1R7 has been modified to accept a new option (-converttov3) that can be used to convert a version 1.4 zFS aggregate back to a version 1.3 zFS aggregate. The new syntax is the following:

```
ioeagslv -aggregate name [-recoveronly]
[{-converttov3 | -verifyonly | -salvageonly}] [-verbose] [-level] [-help]
```

The following is a sample job:

```
//USERIDA  JOB ,'Salvage',
// CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//STEPLIB  DD DSN=hlq.MIGLIB,DISP=OLD
//SALVAGE  EXEC PGM=IOEAGSLV,REGION=0M,
// PARM=('-aggregate aggr.name -converttov3')
//SYSPRINT DD SYSOUT=H
//STDOUT   DD SYSOUT=H
```

```
//STDERR   DD SYSOUT=H
//SYSUDUMP DD SYSOUT=H
//CEEDUMP  DD SYSOUT=H
//*
```

Notes:

1. When -convertov3 is specified, the aggregate is recovered (that is, the log is replayed) whether or not -recoveronly is specified.
2. If a conversion is interrupted, it must be run again.

zFS: Migrating from HFS to zFS in z/OS V1R7

This section provides notes on using the BPXWH2Z tool and the z/OS V1R7 level of the pax command.

Using the BPXWH2Z Tool

BPXWH2Z is an ISPF based tool that migrates HFS file systems to zFS file systems. It lets you alter space allocation, placement, SMS classes and data set names. You can invoke the BPXWH2Z tool from the ISPF COMMAND panel.

In summary, you can do the following:

- Migrate HFS file systems (both mounted and unmounted) to zFS file systems. If the HFS being migrated is mounted, the tool automatically unmounts it and then mounts the new zFS file system on its current mount point.
- Define zFS aggregates by default to be approximately the same size as the HFS. The new allocation size can also be increased or decreased.
- Have the migration run in TSO foreground or UNIX background.

See the ″Migrate from HFS file systems to zFS file systems″ section in *z/OS Migration* for more information.

A walkthrough sample on how to use the BPXWH2Z migration tool: Here we will walk through an HFS to zFS migration. The file systems used are listed in Table 14. Obviously you wouldn't want to migrate a zFS to a zFS but we included one to show that the tool will ignore the non-HFS type file systems that are submitted for migration.

Table 14. Migrating HFS and zFS file systems with the BPXWH2Z migration tool

File system being migrated	File system type	File system status
OMVSSPN.OZTEST.HFS	HFS	MOUNTED
OMVSSPN.OZTEST1.HFS	HFS	NOT MOUNTED
OMVSSPN.OZTEST2.ZFS	ZFS	NOT MOUNTED

The following are the steps we took when creating the new zFS file systems:

1. Start BPXWH2Z. We started BPXWH2Z from an ISPF Command Shell (ISPF Option 6)
2. Enter the new HFS file system name(s) that you want to migrate to zFS. Knowing that the only file systems with the OMVSSPN.OZTEST* qualifier are the ones listed above, we entered OMVSSPN.OZTEST* in Figure 46 on page 168 as the name of the file system that we wanted to migrate from HFS to zFS and pressed enter.

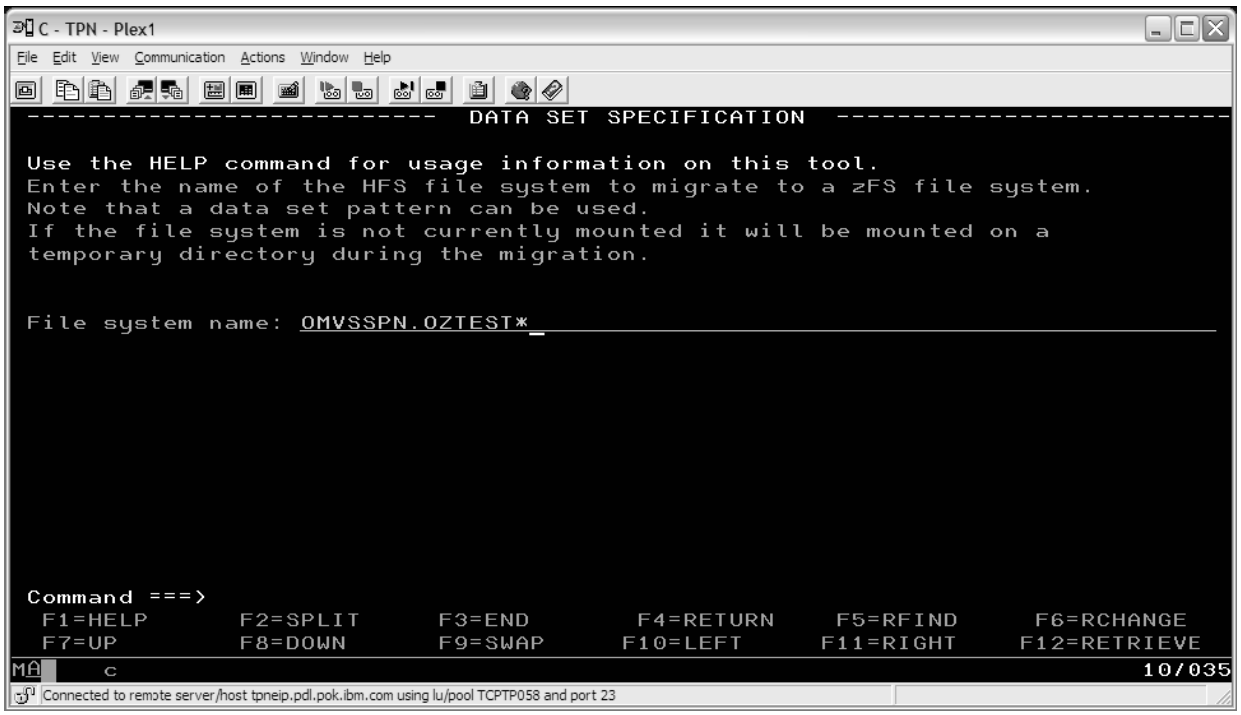


Figure 46. OMVSSPN.OZTEST qualifier*

3. Enter class and volume defaults. BPXWH2Z wanted to know about the class and volume defaults to use when creating the new zFS file systems. In Figure 47 we left these fields blank so that the tool uses the same names as the current HFS allocation of the file system being migrated. We pressed enter.

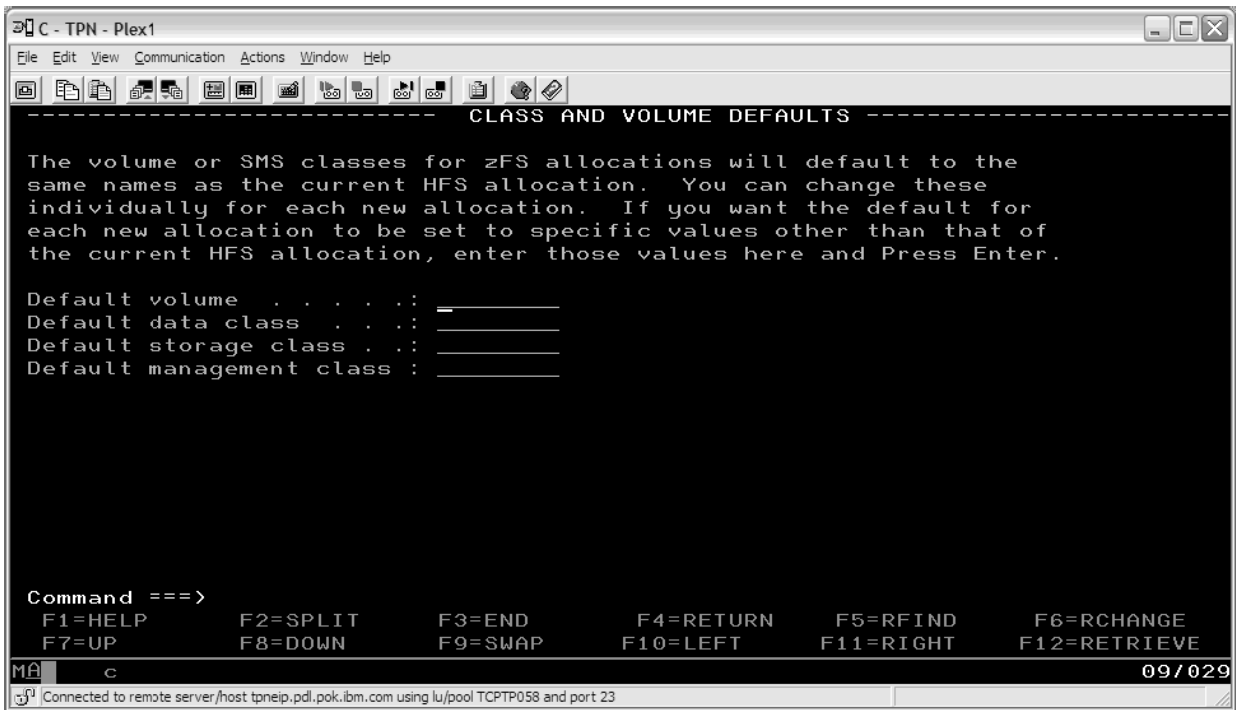


Figure 47. Entering Class and Volume Defaults

4. BPXWH2Z notification of non HFS file systems. In Figure 48 BPXWH2Z told us that one of the file systems we submitted for migration is not an HFS and that it is skipping it. We pressed enter.

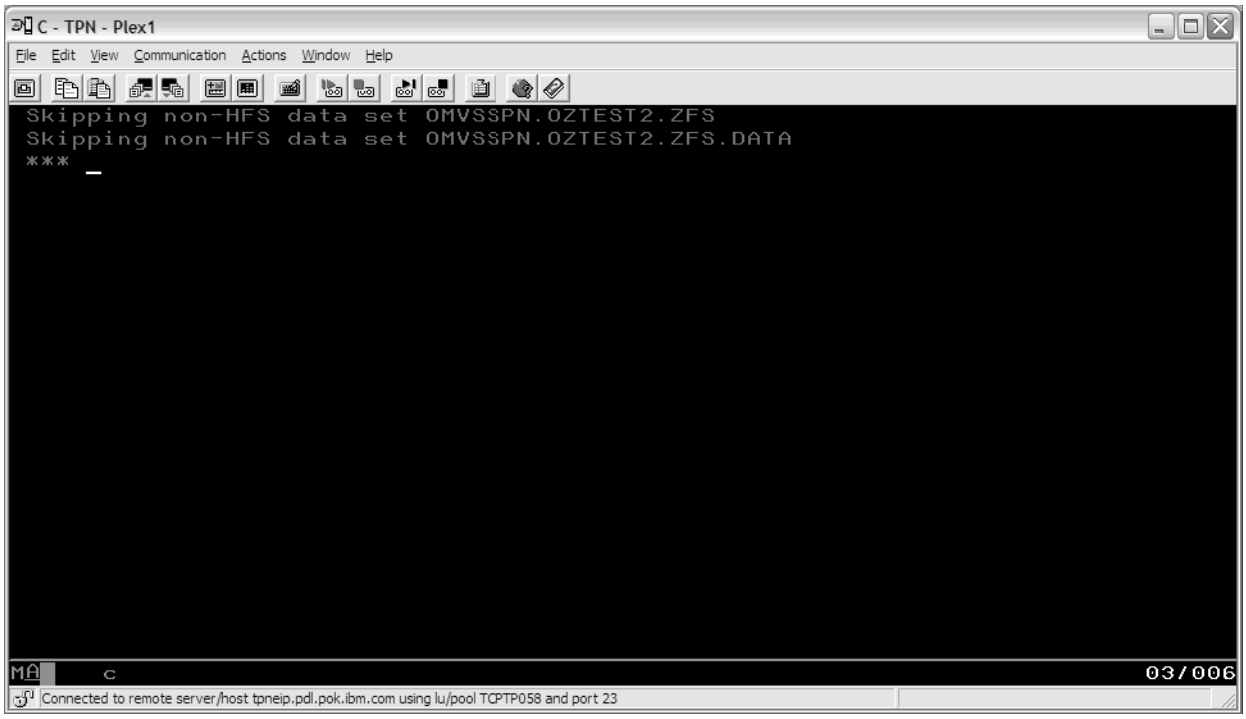


Figure 48. BPXWH2Z notification of non HFS file systems

5. Figure 49 on page 170 lists all the file systems we submitted for migration along with some of their attributes, all of which can be edited by the user.

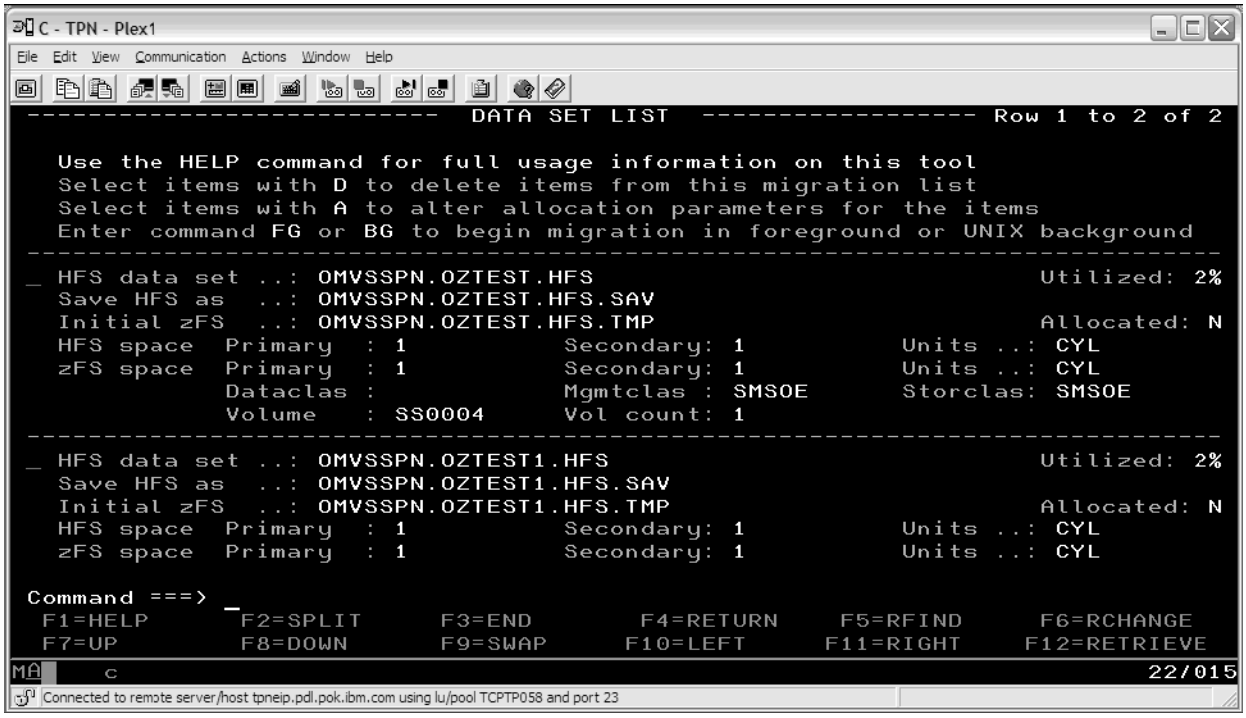


Figure 49. File systems we submitted for migration

6. In Figure 50 we edited some of the attributes. We put an A by the “HFS data set ..:” field. The location is indicated by an underscore and then we pressed enter.

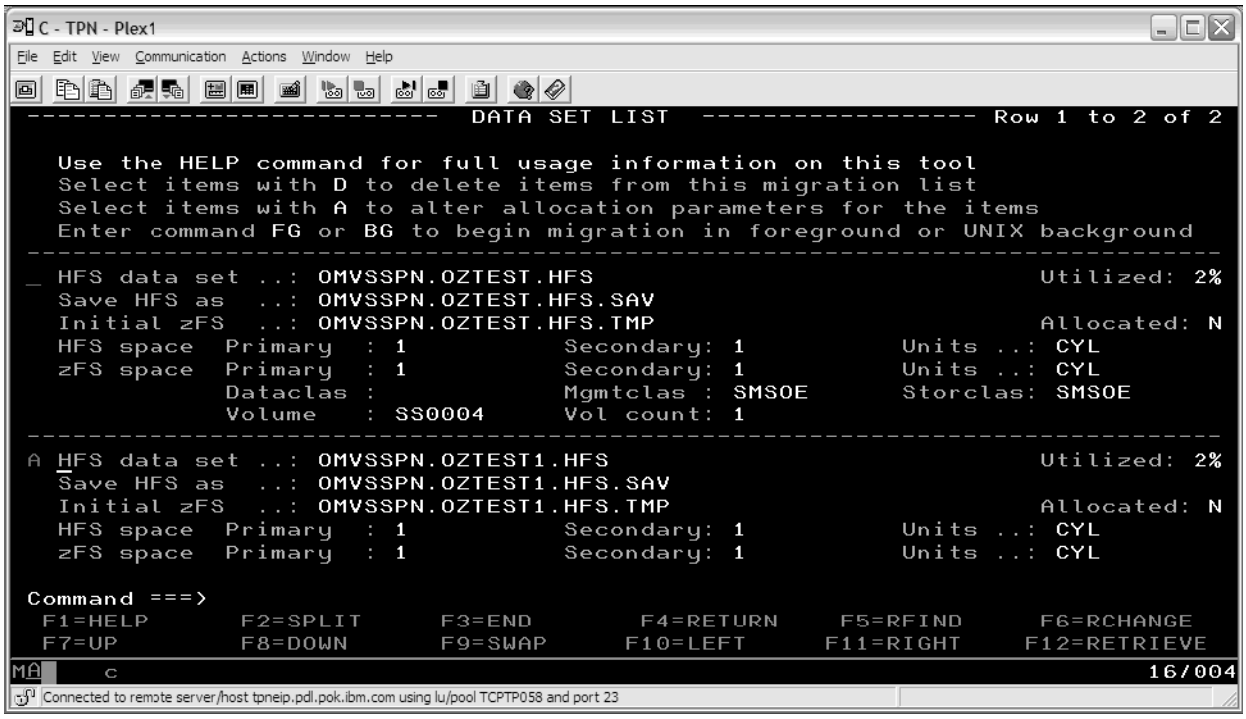


Figure 50. File systems we submitted for migration (cont.)

7. We were taken to Figure 51 where we had the option to edit these file system attributes. We decided not to and pressed enter.

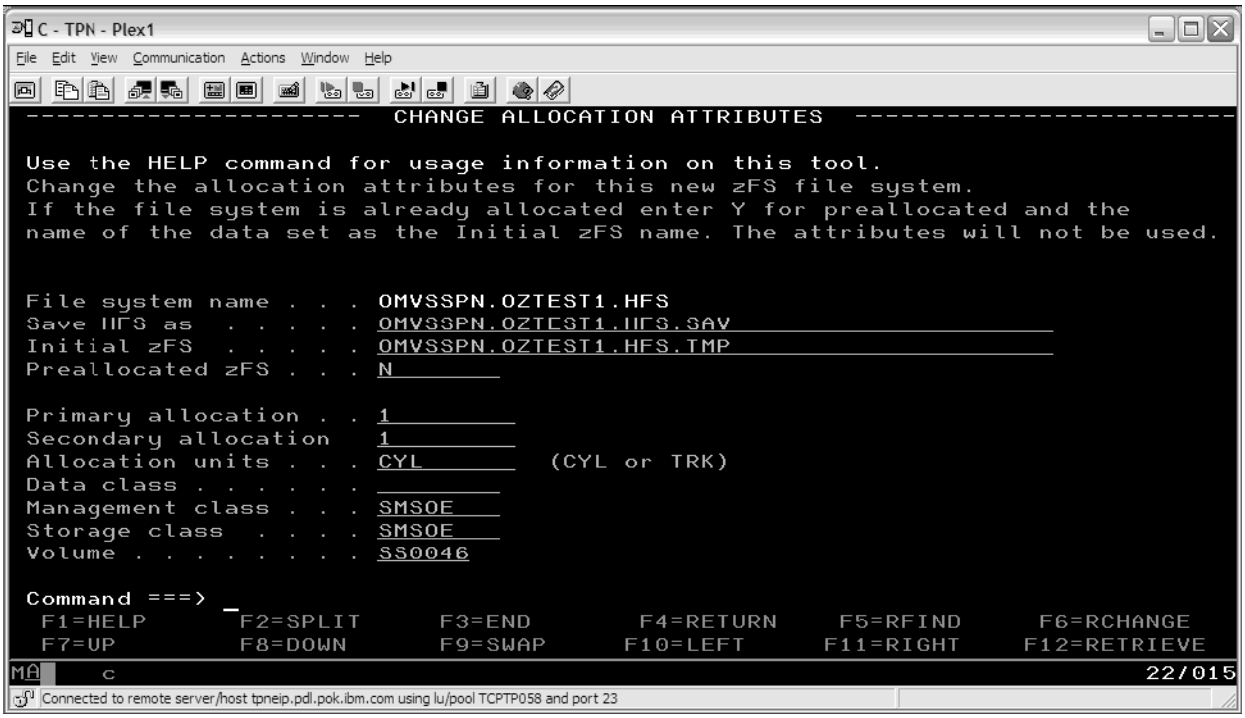


Figure 51. Change Allocation Attributes screen

8. Next we started the migration by entering “FG” on the command line and pressing the enter button. We could have started the migration in the back ground by entering “BG” but we wanted to show you the output that the tool creates shown in Figure 52 on page 172 as it goes through a migration.

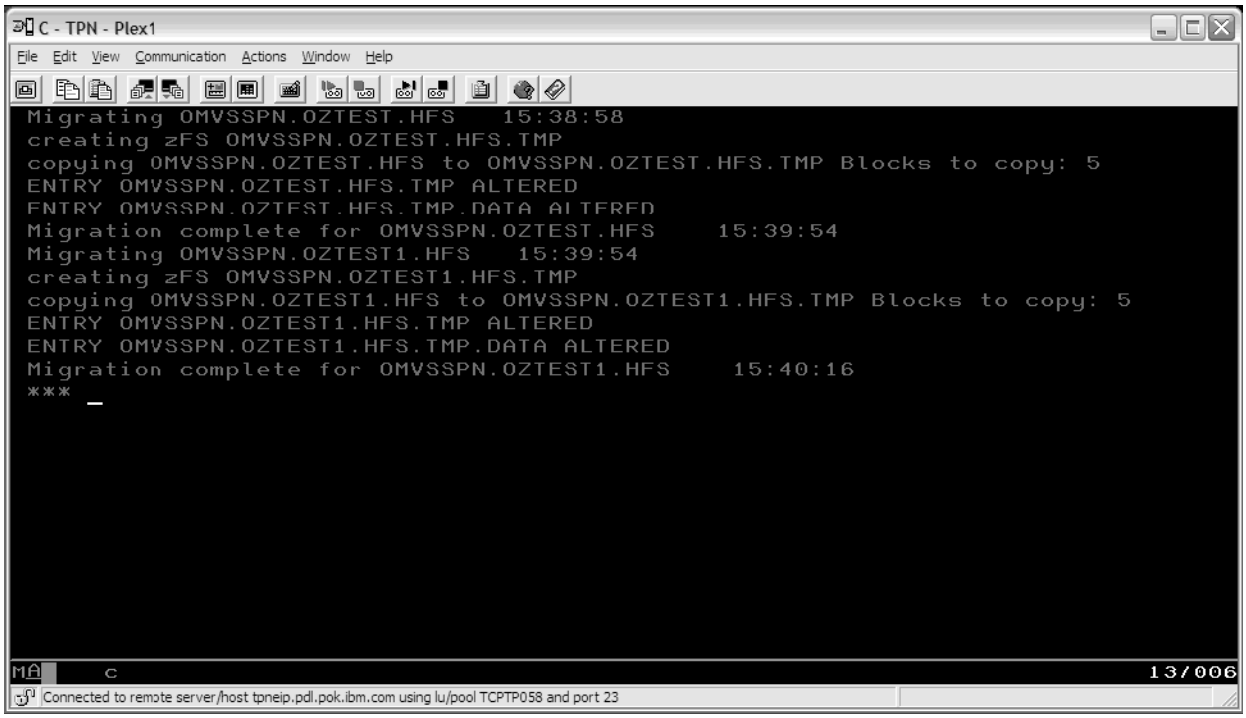


Figure 52. Output received after migrating with the FG command

9. Once the migration was complete we pressed enter and were sent back to the ISPF Command Shell (ISPF Option 6)

There are a couple scenarios that we would like to draw your attention to:

1. *The file system you want to migrate is mounted*. You start the migration.

   During the migration the source file system will be *unmounted* and the new copy will be *mounted* on the source file system's mount point. Since the file system you are migrating will be *unmounted* and replaced with the new copy, there should not be any activity going against that file system during the migration, even if it is a read only file system.

   Migrating a file system that is active at the time of the migration can be disruptive to the system or the users' applications that are using that file system. For example, consider an OMVS user's home directory file system; an HFS named: OMVSSPN.USER.OZ.FS. If this user has an OMVS session running then this file system is mounted and active. If you migrate OMVSSPN.USER.OZ.FS then it will be force *unmounted* and will cause problems with the user's OMVS session. The user will have to get out of OMVS and get back in it for it to pick up the new zFS copy as the home directory file system.
2. *The file system you want to migrate is not mounted* . You start the migration.

   During the migration the source file system and the new one will be *mounted* in order to perform the copy. At the end of the migration however both file systems will be *unmounted*.

Using the z/OS V1R7 Level of the pax Command

The z/OS V1R7 version of the **pax** command can also be used for migrating a file system from HFS to zFS, as follows:

- Manually create a zFS
- Mount the zFS and the HFS

- Use **pax** to copy the HFS to a zFS.

Example: If you want to copy the HFS to a zFS and you have OMVSSPN.MYHFS.HFS and OMVSSPN.MYZFS.ZFS, you can mount them both at /HFSmountPoint and /ZFSmountPoint, respectively. Then you can run **pax** to copy the HFS into the zFS, as follows:

```
pax -rw -X -E /HFSmountPoint /ZFSmountPoint
```

See *z/OS UNIX System Services Command Reference*, SA22-7802 to learn more about the pax command and its options.

Using the zSeries File System (zFS) version root file system

As of z/OS V1R7, zSeries File System (zFS) is the strategic file system type for the z/OS UNIX System Services file system hierarchy. zFS has higher performance characteristics than the HFS file system type.

The version file system is the IBM-supplied root file system. It contains the binary files and text files delivered by IBM, which is called the version root (in a sysplex environment) or the root file system (in a non-sysplex environment). To avoid confusion with the sysplex root file system, the version root file system known as the ″root file system″ can be referred to as the ″version file system.″

IBM supplies the version file system in ServerPac. CBPDO users obtain the version file system by following directions in the Program Directory. There is one version file system for each set of systems participating in a shared file system which are at the same release level.

Prior to z/OS V1R7, Integration Test has been receiving and configuring an HFS file system for our version file system in our sysplex environment. Since z/OS V1R7, we are now receiving and configuring a zFS file system. We found that there is little or no difference in the way we configured a zFS version file system compared to an HFS version file system. The following was performed, which we call STAGE3, to implement the zFS version file system. Note that the obtaining, restoring, and customization of the zFS version file system may be different for your installation. Following are the steps we used:

1. Request and obtain, with each new version or service build, a zFS dumped version file system.
2. Restore the dumped zFS version file system.
3. Mount it on a temporary mount point.
4. From an operating z/OS UNIX System Services system, we ran our customization jobs against the version file system mounted on the temporary mount point, which execute:
   a. SYS1.SAMPLIB(BPXISETS) REXX exec to convert the */etc* and */var* directories to symlinks. Note that if you require the */etc* or */var* symbolic links to be removed and the */etc* or */var* directories recreated, use the BPXISETD REXX exec from SAMPLIB.
   b. Specialized customization for our location, symlinks to product file systems mounted off the sysplex root file system, element and feature specific actions, sysplex and system specific file systems modifications and service, and other needed modifications.
5. Unmount the version file system from the temporary mount point.
6. Ensure that this zFS file system is identified as the version file system in the BPXPRMxx member you use to initialize z/OS UNIX System Services. We have

the mount identified for the version file system as READ-ONLY, and designated as AUTOMOVE. If needed, an administrator and superuser can use the REMOUNT option on the UNMOUNT command to change to READ-WRITE, then back to READ-ONLY. Note that the mount point for the version file system is dynamically created if the VERSION statement is used in BPXPRMxx.

7. IPL with the associated SYSRES volume(s), using this new zFS version file system.

zFS: Unquiesce Console Modify Command

In cases where a zFS aggregate must be quiesced (for example, during backup), it may happen that the job that quiesced the aggregate may fail. If this occurs, the aggregate is unavailable to any application, since it remains quiesced. z/OS V1R7 provides an operator command to allow the operator to unquiesce the aggregate, thus allowing applications to access the aggregate.

The modify zFS unquiesce command must be issued from the owning system of the file system.

Format:
MODIFY ZFS,UNQUIESCE,*aggregate_name*

Note: The **zfsadm unquiesce** command (**zfsadm unquiesce -aggrname** *name*) can be used from any system in the shared file system sysplex.

The following is an example of the use of the unquiesce console command:

From the ″D OMVS,F″ console display, or the ″zfsadm lsaggr″ command, we see that this file system is owned by Z0. Note that the indication of the zFS file system quiesce is from the zfsadm commands.

```
D OMVS,F
...
ZFS            17 ACTIVE                      RDWR  10/12/2005  L=24
  NAME=OMVSSPN.PET3.ZFS.FS                          20.33.29    Q=938
  PATH=/pet3
  AGGREGATE NAME=OMVSSPN.PET3.ZFS.FS
  OWNER=Z0       AUTOMOVE=Y CLIENT=Y
...

zfsadm lsaggr
...
OMVSSPN.PET3.ZFS.FS                              Z0        R/W QUIESCE
...

zfsadm aggrinfo -aggregate omvsspn.pet3.zfs.fs
MVSSPN.PET3.ZFS.FS (R/W COMP QUIESCED): 561 K free out of total 720
```

From system TPN, which is not the owning system, if we issue the following, we get Return code 129 and Reason code EF176775. This indicated that the unquiesce console command has to be entered from the owning system.

```
MODIFY ZFS,UNQUIESCE,OMVSSPN.PET3.ZFS.FS

IOEZ00425E UNQUIESCE FAILURE: rc = 129 rsn = EF176775
IOEZ00024E zFS kernel: MODIFY command - UNQUIESCE,OMVSSPN.PET3.ZFS.FS failed.
```

From Z0, which is the owning system, the unquiesce is successful.

```
R0,Z0,MODIFY ZFS,UNQUIESCE,OMVSSPN.PET3.ZFS.FS

IOEZ00025I zFS kernel: MODIFY command - UNQUIESCE,OMVSSPN.PET3.ZFS.FS completed successfully.
```

Displaying z/OS UNIX and zFS diagnostic information through message automation

Following the migration to z/OS V1R7 some actions were taken to collect better documentation in cases of z/OS UNIX System Services Mount Latch hangs in the Integration Test parallel sysplex.

The process to properly diagnose and resolve z/OS UNIX Mount Latch contention is documented in *z/OS MVS Diagnosis: Reference*, GA22-7588 under the section "Understanding UNIX System Services latch contention".

This is an extra step to collect some documentation automatically in case the console messages displayed are missed by the operators or the system programmers.

A message automations tool is used to trap some zFS and z/OS UNIX System Services messages and to run some commands when these traps hit.

The message automations tool used in the Integration Test environment is part of the "IBM Tivoli NetView for z/OS". For more information on this product visit:

`http://www.ibm.com/software/tivoli/products/netview-zos/index.html`

The messages trapped and the commands executed through NetView for z/OS are listed in Table 15. Once a trap hits the respective command(s) are executed on the system that the message was displayed.

Table 15. Messages trapped and commands executed through NetView for z/OS

Message trapped	Command(s) executed
IOEZ00524I zFS has a potentially hanging thread.	D OMVS,W F ZFS,QUERY,THREADS
BPXM056E UNIX SYSTEM SERVICES LATCH CONTENTION DETECTED	D OMVS,W
BPXM057E UNIX SYSTEM SERVICES LATCH CONTENTION NOT RESOLVING	D OMVS,W
IOEZ00547I zFS has a potentially hanging XCF request.	D OMVS,W F ZFS,QUERY,THREADS

In addition to the trap/execute method followed, the automations tool was also used to execute the two commands found in Table 15 every three hours, on each system in the sysplex, even if none of those messages appeared.

The reason for such an implementation is that there may be multiple systems involved in a z/OS UNIX Mount Latch hang in a sysplex. The culprit might not always be the one that the above messages are displayed on. Remember that these display commands provide information about only the systems that they are executed on and not the others.

One solution to the issue of only getting responses back from systems the commands were issued on could have been to route and execute the display commands on every single system in the sysplex each time one system hit one of the traps. However, multiple systems in the sysplex could receive the above messages all around the same time. For example, if 4 systems out of 10 received

the above messages then there would be 4 x 10 = 40 executions of the commands from around the same time frame, 4 per system, which can lead to excessive amounts of wasted log space.

The decision to execute these commands every three hours and not four or five was due to the amount of log space these executed commands took under normal conditions in the Integration Test environment. Of course other shop limitations and procedures played a role in this decision also.

Here are some details about these trapped messages and the commands that are executed:

IOEZ00524I
is self explanatory; it is displayed when the zFS Hang Detector notices that a thread might be hung.

IOEZ00547I
indicates that the zFS Hang Detector identified that a thread sent a message to another member of the sysplex and that zFS may be experiencing a hang condition.

BPXM056E
indicates that the system detected a UNIX System Services latch contention situation that has existed for an excessive amount of time. As a result this task is not progressing as expected nor are the tasks waiting on the held resources.

BPXM057E
indicates that the "F BPXOINIT,RECOVER=LATCHES" command did not resolve the UNIX System Services latch contention. You may execute the recover command as part of the process to resolve z/OS UNIX Mount Latch hangs. The process to diagnose and resolve such hangs is documented in *z/OS MVS Diagnosis: Reference*, GA22-7588.

The command *D OMVS,W* displays the following, where the display output is for the specific system on which the command was entered:

- The task that is holding the LFS Mount Latch
- The reason why the task started holding the LFS Mount Latch
- What that task is doing
- The tasks waiting for that Mount Latch and why they want it
- The tasks that are currently waiting for messages from other systems in a sysplex.

For more details on the *D OMVS,W* command see *z/OS MVS Diagnosis: Reference*, GA22-7588 and "Displaying z/OS UNIX System Services Status" section in *z/OS MVS System Commands*, SA22-7627.

The modify zFS query command is used to display zFS counters or values. The syntax is as follows:

```
modify procname,query,{all | level | settings | storage | threads}
```

The zFS procname used in the Integration Test environment is called "ZFS", so the command executed is:

```
F ZFS,QUERY,THREADS
```

The above command displays the threads running in the zFS address space. For more information on the modify command with respect to the zFS processes see the “modify zfs process” section of the *z/OS Distributed File Service zSeries File System Administration*, SC24-5989.

It must be noted that displaying z/OS UNIX and zFS diagnostic information through message automation is something that was implemented in the Integration Test environment so that automations can catch the messages that are missed by the operators and the system programmers.

The messages listed above can be displayed even if the contention is temporary. In which case the automations will still execute the display and modify commands. The output from these commands can be rather large, where the size really depends on the environment and the activity on the systems.

Removing additional diagnostic data collection from OMVS CTRACE LOCK processing

We noticed increased CPU utilization and performance degradation running z/OS V1R6 with OMVS CTRACE options set at ALL or any set of options that include LOCK. This was caused by calls to query latch ownership activity when a dubbed z/OS UNIX System Services task terminates to collect additional diagnostic information. This utilization was especially noticeable when running with OMVS Heavy-weight threads, which are prevalent, for example, in Java workloads.

Due to the effect this has on the system when using the LOCK CTRACE option, the additional diagnostic data collection calls were removed from z/OS V1R6 by APAR OA10735 (PTF UA16780).

Additional collection of diagnostic latch information may be considered in future releases by other means.

Chapter 13. Using LDAP Server

LDAP Server is a component of z/OS Security Server which uses the Lightweight Directory Access Protocol (LDAP) standard, an open industry protocol for accessing information in a directory.

This chapter contains the following sections:

- "Overview of our LDAP configuration"

Overview of our LDAP configuration

We have a multiplatform LDAP configuration and we use both replication and referral. Figure 53 shows a high-level view of our LDAP multiplatform configuration:

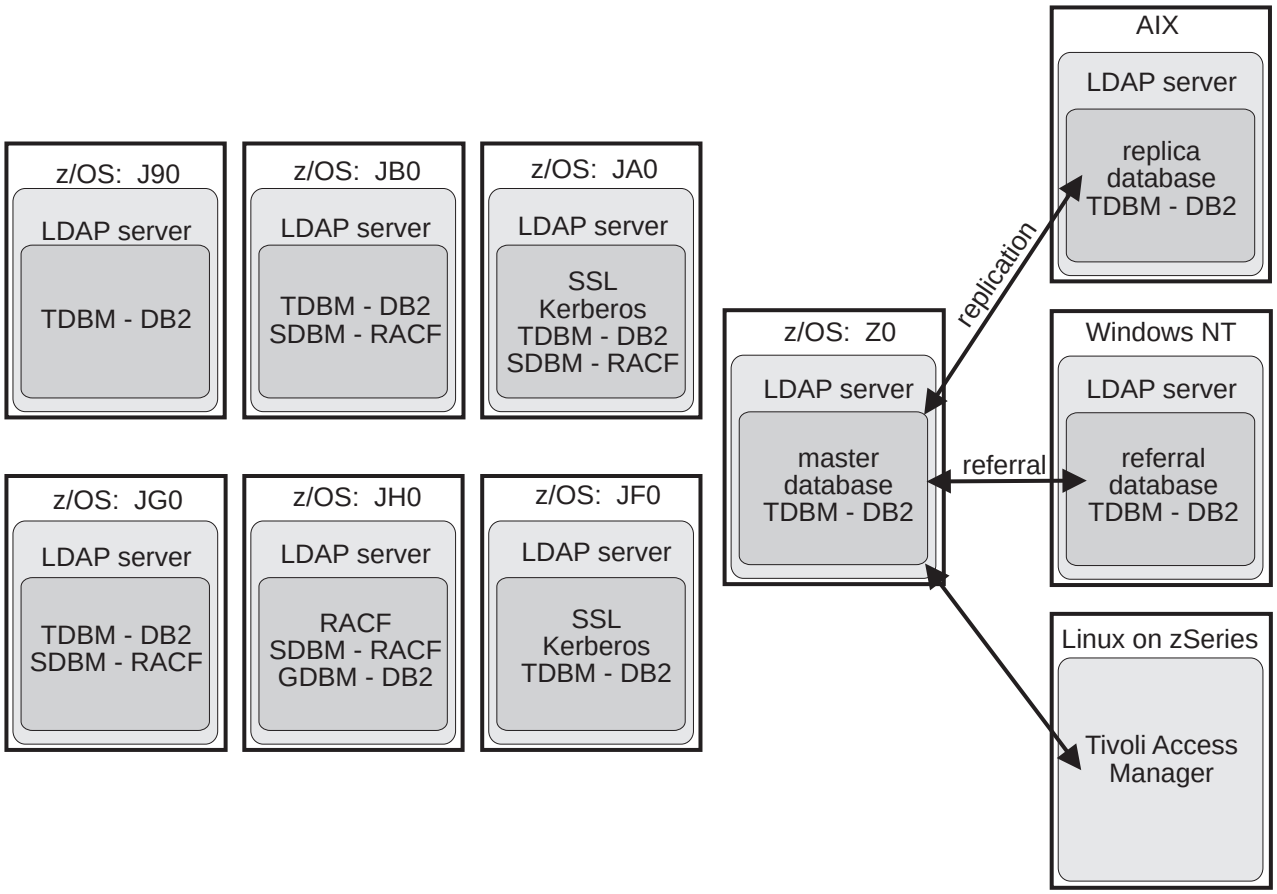


Figure 53. Overview of our LDAP configuration

Our LDAP environment includes the following features:

- **Master LDAP server:** Our master server on z/OS system Z0 has a DB2 backend database, TDBM.
- **Secure LDAP servers:** We have two LDAP servers in our sysplex (on systems JA0 and JF0) that are set up for SSL secure transactions and Kerberos authentication. The servers have a TDBM backend and listen on port 636.
- **LDAP server for our RACF backend:** We have an LDAP server with an SDBM backend that connects to the RACF directory for our sysplex.

- **LDAP referral from Windows NT to z/OS:** We manage an LDAP server on Windows NT using a Web-based management tool at:

  `http://`*`NT_server_IP_address`*`/ldap/index.html`

  This LDAP server has a TDBM backend and has a general referral in its configuration file that points to our master LDAP server on z/OS. This allows a user to issue an **ldapsearch** command from the Windows NT LDAP server for an entry that is not found in that directory but that might be found in the directory on the master server on z/OS. The **ldapsearch** command returns all matching entries from both directories.
- **LDAP referral from z/OS to Windows NT:** We maintain an LDAP server referral database on Windows NT using the Directory Management Tool for Windows. This server is set up to have referral processing enabled between the master LDAP server on z/OS and the LDAP server on Windows NT.
- **LDAP replica server on AIX:** We manage an LDAP replica server on AIX using a Web-based management tool at:

  `http://`*`AIX_server_IP_address`*`/ldap/index.html`

  This LDAP server has a TDBM backend and was configured through the Web-based management tool to be a replica of the master LDAP server on z/OS.
- **Tivoli Access Manager running on Linux on zSeries:** We set up Tivoli Access Manager on a SUSE Linux image running on zSeries to enable cross-platform testing between Linux and z/OS. Tivoli Access Manager uses the z/OS LDAP Server as a backend to store user ID information that is used to authorize access for users of Tivoli Access Manager. We perform our testing by running shell scripts on Linux that create a workload to stress the master LDAP server on z/OS.

Chapter 14. Using the IBM WebSphere Business Integration family of products

The IBM WebSphere MQ (formerly MQSeries) family of products forms part of the newly re-branded WebSphere Business Integration portfolio of products. These products are designed to help an enterprise accelerate the transformation into an on demand business.

This chapter discusses the following topics:

Using WebSphere MQ shared queues and coupling facility structures

Using Websphere MQ, programs can talk to each other across a network of unlike components, including processors, operating systems, subsystems, and communication protocols, using a simple and consistent application programming interface.

We recently migrated our WebSphere for z/OS queue managers from V5.3.1 to V6.0 (we will explain our migration experience later in this chapter). For this reason much of our discussion here focuses on our experience with the usage and behavior of the coupling facility structures that support shared queues as well as using shared channels in a distributed environment with queue managers running V5.3.1. The next release of this test report will dicuss our experiences with queue managers at V6.0.

We used information from the following sources to set up and test our shared queues:

- *WebSphere MQ for z/OS System Administration Guide*, SC34-6053 and *WebSphere MQ for z/OS System Setup Guide, SC34-6583*, for information about recovery from DB2, RRS, and CF failures. This document is available from the WebSphere Business Integration library at www.ibm.com/software/integration/websphere/library/.
- *WebSphere MQ in a z/OS Parallel Sysplex Environment*, SG24-6864, available from IBM Redbooks at www.ibm.com/redbooks/
- *WebSphere MQ Queue Sharing Group in a Parallel Sysplex Environment*, REDP-3636, available from IBM Redbooks at www.ibm.com/redbooks/

Our queue sharing group configuration

We currently have two queue sharing groups: one with three members and another with four members. The smaller queue sharing group is for testing new applications or configurations before migrating them to our production systems. The queue sharing groups each connect to different DB2 data sharing groups. This discussion will focus on the four-member production queue sharing group. All of the queue managers in the group run WebSphere MQ for z/OS Version 6.0.

Managing your z/OS queue managers using WebSphere MQ V6 Explorer

Websphere MQ V6 now offers an extensible Eclipse-based graphical configuration tool which replaces the Windows-based MQ Explorer. The Websphere MQ V6 Explorer is supported on both Windows and Linux operating systems.

This tool, in conjunction with SupportPac MO71, has provided us with the ability to monitor as well as perform remote administration and configuration of our entire MQ network. The queue manager being managed does not have to be running WebSphere MQ V6 except when the queue manager is running on z/OS. If you wish to manage your z/OS queue managers using WebSphere MQ V6 Explorer and security is enabled on these queue managers you be will required to install refresh pack 6.0.1.1. This is because userids on z/OS are validated by RACF security and should be in uppercase. Without refresh pack 6.0.1.1, WebSphere MQ V6 Explorer transmits the userid to the queue manager in lowercase and subsequently the connections are rejected by RACF.

Our coupling facility structure configuration

We defined our MQ coupling facility structures to use two coupling facilities (CF2 and CF3) as defined in the preflist in the structure definitions. (See “Coupling facility details” on page 10 for details about our coupling facilities.)

The following is the structure definition for our CSQ_ADMIN structure:

```
STRUCTURE NAME(MQGPCSQ_ADMIN)
          INITSIZE(18668)
          MINSIZE(15000)
          DUPLEX(ENABLED)
          SIZE(18668)
          ALLOWAUTOALT(YES)
          PREFLIST(CF3,CF2)
          REBUILDPERCENT(1)
          FULLTHRESHOLD(85)
```

We also have the following four message structures defined to support different workloads:

- MSGQ1 — for the batch stress workload
- CICS — for the CICS bridge application
- EDSW — for the IMS bridge application
- WMQI — for the WebSphere Message Broker retail application

The following is the structure definition for the message structure that supports the MQ-CICS bridge workload:

```
STRUCTURE NAME(MQGPCICS)
          INITSIZE(10240)
          DUPLEX(ENABLED)
          SIZE(20480)
          ALLOWAUTOALT(YES)
          PREFLIST(CF2,CF3)
          REBUILDPERCENT(1)
          FULLTHRESHOLD(85)
```

The other three message structures are defined similarly, except for the sizes. All of the structures are enabled for duplexing.

We chose to create multiple message structures in order to separate them by application. That way, if there is a problem with a structure, it will not impact the

other applications. However, this is not necessarily the recommended approach from a performance perspective. See the Redbook Paper *WebSphere MQ Queue Sharing Group in a Parallel Sysplex Environment* for more information.

The CICS, EDSW, WMQI, and MSGQ1 structures are recoverable and are backed up daily.

Recovery behavior with queue managers at V5.3.1 using coupling facility structures

We conducted the following types of test scenarios during our z/OS release testing:

- CF structure errors
- CF structure duplexing and moving structures between coupling facilities
- CF-to-CF link failures
- MQ CF structure recovery

During these tests, we monitored the behavior of the MQ queue managers as well as the behavior of applications that use shared queues.

Queue manager behavior during testing

We observed the following behavior during our test scenarios:

CF structure errors: With the MQ CICS bridge workload running, we used a local tool to inject errors into the coupling facility structures. When we injected an error into the MQ administrative structure, the structure moved to the alternate coupling facility, based on the preflist, as expected. Throughout the test, the CICS bridge workload continued to run without any errors.

CF structure rebuild on the alternate coupling facility: With system-managed CF structure duplexing active and a shared queue workload running, we issued the SETXCF STOP,REBUILD command to cause XCF to move the MQ structures to the alternate coupling facility. The queue manager produced no errors and the application continued without any interruption.

We also tested recovering into an empty structure. We first issued the SETXCF FORCE command to clear the structure, followed by the RECOVER CFSTRUCT(CICS) TYPE(PURGE) command. Again, the structure recovered with no errors.

Additional experiences and observations

MQ abends during coupling facility failures: Although coupling facility failures are extremely rare under normal operations, we induce many failures in our environment in the course of our testing. When coupling facility failures occur which have an impact on WebSphere MQ, such problems generally manifest themselves as MQ dumps with abend reason codes that start with 00C51*nnn*. Many of these are actually coupling facility problems or conditions that result in MQ having a problem and are not necessarily MQ problems in their own right. When such abends occur, we suggest that you analyze the system log for any IXC or IXL messages that might indicate a problem with a coupling facility.

Intra-group queuing: We have all members of the queue sharing group set up for intra-group queuing. This was done by altering the queue manager to enable intra-group queuing. SDSF makes use of the SYSTEM.QSG.TRANSMIT shared queue for transmitting data between SDSF servers instead of the cluster queues. It continues to use the cluster queues and channels for members not in the queue

sharing group. Currently all systems in our sysplex have the SDSF MQ function enabled so job output for one system can be viewed from any other system in the sysplex.

Effects of DB2 and RRS failures on MQ: We also tested how MQ reacts when DB2 or RRS become unavailable. The following are some of our observations:

- APAR PQ77558 fixes a problem with MQ V5.3.1 when RRS is cancelled while the queue manager is running.
- When DB2 or RRS become unavailable, the queue manager issues an error message to report its loss of connectivity with DB2 and which subsystem is down. An example of such a message is:

  ```
  CSQ5003A !MQJA0 CSQ5CONN Connection to DB2 using DB1G pending, no active DB2
  ```

  When DB2 becomes available again, MQ issues a message to report that it is again connected to DB2. For example:

  ```
  CSQ5001I !MQJA0 CSQ5CONN Connected to DB2 DBD1
  ```
- MQ abend reason codes that indicate a DB2 failure start with 00F5*nnnn*.

Notes about MQ coupling facility structure sizes:

- All of our MQ coupling facility structures are defined to allow automatic alter (by specifying ALLOWAUTOALT(YES) in the structure definitions in the CFRM policy), whereby XCF can dynamically change the size of a structure, as necessary. This is beneficial because it allows XCF to automatically increase the size of a message structure as needed to hold more messages.
- When we first defined the CSQ_ADMIN structure, we made it 10000K bytes in size. Our original sizing was based on the guidelines in *WebSphere MQ for z/OS Concepts and Planning Guide*, GC34-6051. However, we have since migrated to a higher CFCC level and increased the number of queue managers in the queue sharing group, which increases the size requirement for the CSQ_ADMIN structure. As a result, the queue manager recently failed to start because the CSQ_ADMIN structure was too small and issued the following message:

  ```
  CSQE022E !MQJA0 Structure CSQ_ADMIN unusable, size is too small
  ```

  We used the SETXCF START,ALTER command to increase the size of the structure. The following is an example of the command we issued:

  ```
  SETXCF START,ALTER,STRNAME=MQGPCSQ_ADMIN,SIZE=16000
  ```

  Accordingly, we also increased the value of INITSIZE() and MINSIZE() for CSQ_ADMIN in the CFRM policy from 10000 to 15000 to accommodate the increase in usage.

Running WebSphere MQ V5.3.1 implemented shared channels in a distributed-queuing management environment

We implemented shared channels within the larger of our two queue sharing groups to bolster our distributed-queuing management (DQM) environment. Previously, we have had a DQM workload that exercised distributed messaging using MQ channels that provided an environment to test channel functionality such as SSL, as well as more general testing such as load stress. For z/OS V1R5, we modified the underlying DQM environment to utilize both shared inbound and shared outbound channels without having to change the workload application. We are now able to handle higher amounts of inbound messages from remote MQ clients and, at the same time. provide transparent failover redundancy for those inbound messages.

Our MQ ″clients″ are in fact full MQ servers on distributed platforms such as Linux and Windows 2000.

Our shared channel configuration

The following sections describe the configuration of our shared inbound and outbound channels. We used information in *WebSphere MQ Intercommunication*, SC34-6059, to plan our configuration.

Shared inbound channels

We decided to implement the shared channel environment on our sysplex using TCP/IP services because our distributed DQM clients are mainly TCP/IP clients. All queue managers in the queue sharing group were configured to start group listeners on the same TCP port (1415), as described in the MQ intercommunication guide.

Example: The following is an example of the command to start group listeners on TCP port 1415:

```
START LISTENER INDISP(GROUP) PORT(1415)
```

The MQ intercommunication guide describes how the group listener port maps to a generic interface that allows the queue sharing group to be seen as a single network entity. For our DQM environment, we configure the Sysplex Distributor service of z/OS Communications Server to serve as the TCP/IP generic interface. This is a slight departure from the intercommunication guide, which utilizes DNS/WLM to provide the TCP/IP generic interface. VTAM generic resources is another available service that can provide the generic interface for channels defined using LU6.2 connections.

Example: The following is an example of our Sysplex Distributor definition for TCP port 1415:

```
VIPADYNAMIC
VIPADEFINE MOVEABLE IMMED 255.255.255.0  192.168.32.30
VIPADISTRIBUTE DEFINE 192.168.32.30 PORT 1415
DESTIP 192.168.49.31 192.168.49.32 192.168.49.33 192.168.49.34 92.168.49.36 192.168.49.38
ENDVIPADYNAMIC
```

We added this definition to the TCP/IP profile of one of our queue sharing groups (in this case 192.168.49.32), but it can be added to any TCP/IP host within the sysplex in which the queue sharing group resides. The IP addresses listed for DESTIP are the XCF addresses of the queue managers in our queue sharing group. The remote client can then specify 192.168.32.30 (or, correspondingly, the host name MQGP, which maps to that IP address in our DNS server for our 192.168.*xx.xx* LAN) on its sender channel, which then causes the receiver channel start to be load-balanced using the WLM mechanisms of Sysplex Distributor.

Example: The following is an example of our definitions for the remote sender channel and the local receiver channel:

```
DEFINE  CHANNEL(DQMLNXP.TO.DQMMQGP) +
  REPLACE +
  CHLTYPE(SDR) +
  XMITQ(DQMMQGP.XMIT.QUEUE) +
  TRPTYPE(TCP) +
  DISCINT(15) +
  CONNAME('MQGP(1415)')

DEFINE  CHANNEL(DQMLNXP.TO.DQMMQGP) +
```

```
  REPLACE +
  CHLTYPE(RCVR) +
  QSGDISP(GROUP) +
  TRPTYPE(TCP)
```

Note that QSGDISP(GROUP) specifies that a copy of this channel is defined on each queue manager in the queue sharing group. This allows the inbound channel start request to be serviced by any queue manager in the queue sharing group. At this point, messages can be placed on application queues that are either shared or local to the queue manager (as long as they are defined on each queue manager in the queue sharing group, specifying QSGDISP(GROUP) in the definitions).

Shared outbound channels

The MQ intercommunication guide states that an outbound channel is a shared channel if it moves messages from a shared transmission queue. Thus, we defined a shared transmission queue for our outbound channels, along with an outbound sender channel with a QSGDISP of GROUP. This enables the queue managers in the queue sharing group to perform load-balanced start requests for this channel.

Example: The following is our definition for the shared transmission queue:

```
DEFINE  QLOCAL(DQMLNXP.XMIT.QUEUE) +
  REPLACE +
  QSGDISP(SHARED) +
  CFSTRUCT(MSGQ1)
  TRIGGER +
  TRIGDATA(DQMMQGP.TO.DQMLNXP) +
  INITQ(SYSTEM.CHANNEL.INITQ) +
  USAGE(XMITQ)  +
  STGCLASS(DQMSTG)
```

Example: The following are our definitions for the local sender channel and the remote receiver channel:

```
DEFINE CHANNEL(DQMMQGP.TO.DQMLNXP) +
  REPLACE +
  CHLTYPE(SDR) +
  XMITQ(DQMLNXP.XMIT.QUEUE) +
  QSGDISP(GROUP) +
  TRPTYPE(TCP) +
  DISCINT(15) +
  CONNAME(remote_client_host_name)

DEFINE CHANNEL(DQMMQGP.TO.DQMLNXP) +
  REPLACE +
  CHLTYPE(RCVR) +
  TRPTYPE(TCP)
```

Migrating from WebSphere MQ V5.3.1 to V6

We recently migrated all our I/T systems to the latest WebSphere MQ release V6.0. Our I/T environment consists of 13 systems in a parallel sysplex. Within the MQ environment we had a Queue Sharing Group (QSG) consisting of 7 systems and another QSG consisting of 3 systems. We have since reduced our QSG of 7 systems to a group which only contains 4 systems. The rest of the systems do not participate in a QSG.

Our migration consisted of the following steps:

- “Installing migration PTFs on both systems” on page 187
- “Copying the MQ V6 libraries” on page 187
- “Ensuring APF authorization is in place” on page 187

- "Customizing and running the migration jobs"

Installing migration PTFs on both systems

The first task we needed to perform was to ensure both levels of WebSphere MQ had the migration PTFs installed. According to the migration text in the Websphere MQ Setup guide manual, this maintenance has to be installed on both levels or you will not be able to run in a mixed level QSG environment (using both V5.3.1 and V6) or be able to fall back to the V5.3.1 level. We used the following MQ website link for migration PTF info:

```
http://www-1.ibm.com/support/docview.wss?rs=171 &context=SSFKSJ &q1=migration&uid=swg27006519&loc=en_US&cs=utf-8&lang=en
```

We installed PTFs UK05386, UK05223, UK05224 for MQ V6R0M0 and PTF UK04544 for MQ V5R3M1. The other PTFs referred to in the doclink above had been installed during prior monthly service updates.

Copying the MQ V6 libraries

The next step was to copy over all the MQ V6 libraries from the SMP/E target libraries where we install and apply PTF maintenance. We utilize a flip/flop method for the dataset naming convention so when we go to pick up service we use the set of libraries that are not actively in use. We added a suffix of PETCPYA or PETCPYB to the dataset names.

Ensuring APF authorization is in place

After the MQ libraries were copied to our volumes the next step was to ensure the APF authorization was in place. The volumes we use are on shared volumes and not SMS managed and as such require us to code the VOLSER in the APF define. Following are our APF authorizations:

```
APF FORMAT(DYNAMIC)
APF ADD
    DSNAME(MQS.V6R0M0.SCSQAUTH.PETCPYA)
    VOLUME(SHR009)
APF ADD
    DSNAME(MQS.V6R0M0.SCSQLINK.PETCPYA)
    VOLUME(SHR003)
APF ADD
    DSNAME(MQS.V6R0M0.SCSQANLE.PETCPYA)
    VOLUME(SHR009)
APF ADD
    DSNAME(MQS.V6R0M0.SCSQSNLE.PETCPYA)
    VOLUME(SHR013)
APF ADD
    DSNAME(MQS.V6R0M0.SCSQMVR1.PETCPYA)
    VOLUME(SHR005)
APF ADD
    DSNAME(MQS.V6R0M0.SCSQLOAD.PETCPYA)
    VOLUME(SHR016)
```

Customizing and running the migration jobs

We then had to customize and run the Migration jobs as outlined in the *WebSphere MQ for z/OS System Setup Guide (SC34-6583-00)* found at:

```
http://publibfp.boulder.ibm.com/epubs/pdf/csqsav04.pdf
```

Copying and running the CSQ45ATB sample job

We copied the CSQ45ATB sample from MQS.V6R0M0.SCSQPROC.PETCPYA to local dataset MQS.LOCAL.SCSQPROC.V6R0M0. CSQ45ATB is the multi-step job that migrates the DB2 table definitions from WebSphere MQ Version 5.3 and/or 5.3.1 (with the migration PTFs installed on ALL queue managers in the QSG) to

WebSphere MQ Version 6. The steps in the job create new tables and tablespaces needed for Version 6 and also modify the existing tablespaces so that both levels of WebSphere MQ can co-exist within the same QSG. The job steps perform the following tasks:

1. PREPBIND: Prepares the DB2 tables for binding the CSQ5PQSG plan for WebSphere MQ version 6.0
2. BINDPQSG: Binds the new DB2 plan for CSQ5PQSG for WebSphere MQ version 6.0
3. GRANT: Grants execute authority for the CSQ5PQSG plan
4. MIGRDSG: Migration check
5. TBLCREA: Creates new tables and table spaces
6. TBLADD: Adds information to existing tables
7. BINDPLAN: Re-binds DB2 plans for WebSphere MQ versions 5.3 and/or 5.3.1.

For the customization of the job we used the following values for migrating MQGT QSG which is a 3 way queue sharing group. When we migrated the 7 way QSG MQGP, the only values that needed to be changed from the first migration job were the DB2 data sharing group name, DB2 storage group name, DB2 subsystem name, and the DB2 database name for the WebSphere tables. The settings shown are specific for our environment and in some cases we used the recommended minimum values. You may need to adjust these accordingly to fit your specific environment.

Replace &&DB2QUAL&&
with the high level qualifier of the
DB2 target library data sets.
We substituted value : DB2.DB2810

Replace &&THLQUAL&&
with the high level qualifier of the
WebSphere MQ V6.0 target library data sets.
We substituted value : MQS.V6R0M0.SCSQDEFS.PETCPYA

Replace &&V53QUAL&&
with the high level qualifier of the
WebSphere MQ V5.3 or V5.3.1
target library data sets.
We substituted value : MQS.V5R3M1.SCSQDEFS.PETCPYB

Replace &&LANGLETTER&&
with the letter for the language that
you want messages shown in.
We substituted value : E

Replace &&DSGNAME&&
with the name of the DB2 data-sharing group
used by the WebSphere MQ queue-sharing group.

We substituted value : DSNDB2G

Replace &&DB2SSID&&
with the DB2 subsystem ID or
batch group attach name through which access
is gained to the DB2 data-sharing group.
We substituted value : DB2G

Replace &&DB2STGGRP&&
with the name of the DB2 storage group
associated with the database.
We substituted value : MQSTG

Replace &&DB2TBLSPCBLOBMSGBASE4K&&
with the name of the DB2 table space to be used
for the CSQ.ADMIN_B_MESSAGES table which
contains the BASE table for the BLOBs
to be used to store large SQ messages.
We substituted value : TSBLOBAD

NOTE: This table space will be a PARTITIONed
table space with 4 (FOUR) partitions.

Replace &&DB2BPQMGR4K&&
wth the name of a DB2 4K buffer pool
associated with the table space
to be used for the CSQ.ADMIN_B_MESSAGES table
which contains the BASE table (as above)
We substituted value : BP5

Replace &&DB2TBLSPCLOB1-32K&&
with the name of the DB2 LOB table space
which will contain the BLOBs associated
with the FIRST partition. Large message
data will be stored here.
We substituted value : TSLOBP1

Replace &&DB2TBLSPCLOB2-32K&&
with the name of the DB2 LOB table space
which will contain the BLOBs associated
with the SECOND partition. Large message

data will be stored here.
We substituted value : TSLOBP2

Replace &&DB2TBLSPCLOB3-32K&&
with the name of the DB2 LOB table space
which will contain the BLOBs associated
with the THIRD partition. Large message
data will be stored here.
We substituted value : TSLOBP3

Replace &&DB2TBLSPCLOB4-32K&&
with the name of the DB2 LOB table space
which will contain the BLOBs associated
with the FOURTH partition. Large message
data will be stored here.
We substituted value : TSLOBP4

Replace &&DB2BPBLOB32K&&
with the name of a DB2 32K buffer pool
associated with the LOB table spaces.
We substituted value : BP32K1

IMPORTANT NOTE #### IMPORTANT NOTE #### IMPORTANT NOTE

The bufferpool defined for the LOB table spaces
should be for the EXCLUSIVE use of these table spaces
and not used by any other DB2 resource.
In addition this bufferpool should have the following
parameters set (DWQT=0 and VDWQT=0)

Replace &&DB2VER&&
with the version number of DB2 you
are currently using, for example
1 = DB2 for OS/390 V7.1
We substituted value : 81

Replace &&DB2DBNAME&&
with the name of the DB2 database used for
the WebSphere MQ tables.
We substituted value : MQTSTDB

Replace &&DB2STGGRP&&
with the name of the DB2 storage group
associated with the database.

We substituted value : MQSTG

Replace &&IMS1PRIQTY&&
with the minimum primary space allocation
in kilobytes for the DB2-managed data set of
the CSQ.ADMIN_MESSAGES_IX1 index
We substituted value : 720
Note: This value should be at least 720

Replace &&IMS1SECQTY&&
with the minimum secondary space allocation
in kilobytes for the DB2-managed data set of
the CSQ.ADMIN_MESSAGES_IX1 index
We substituted value : 720
Note: This value should be at least 720

Replace &&LOBPRIQTY&&
with the minimum primary space allocation
in kilobytes for the DB2-managed data set of
the LOB table space
We substituted value : 7200
Note: This value should be at least 7200

Replace &&LOBSECQTY&&
with the minimum secondary space allocation
in kilobytes for the DB2-managed data set of
the LOB table space
We substituted value : 7200
Note: This value should be at least 7200

Replace &&IMSGPRIQTY&&
with the minimum primary space allocation
in kilobytes for the DB2-managed data set of
the CSQ.ADMIN_MSGS_BAUX1 thru 4 indexes
for the LOB auxiliary tables
We substituted value : 720
Note: This value should be at least 720

Replace &&IMSGSECQTY&&
with the minimum secondary space allocation
in kilobytes for the DB2-managed data set of
the CSQ.ADMIN_MSGS_BAUX1 thru 4 indexes
We substituted value : 720
Note: This value should be at least 720 for the LOB auxiliary tables

Replace &&ICLUPRIQTY&&
with the minimum primary space allocation
in kilobytes for the DB2-managed data set of
the CSQ.CLUS_INDEX cluster index

We substituted value : 720
Note: This value should be at least 720

Replace &&ICLUSECQTY&&
with the minimum secondary space allocation
in kilobytes for the DB2-managed data set of
the CSQ.CLUS_INDEX cluster index
We substituted value : 720
Note: This value should be at least 720

Replace &&ICHAPRIQTY&&
with the minimum primary space allocation
in kilobytes for the DB2-managed data set of
the CSQ.DEL_OBJ_CHANNEL index
We substituted value : 4096

Replace &&ICHASECQTY&&
with the minimum secondary space allocation
in kilobytes for the DB2-managed data set of
the CSQ.DEL_OBJ_CHANNEL index
We substituted value : 4096

Replace &&PQSGUSERID&&
with the user ID that will be used for
the CSQ5PQSG utility.
We substituted value : PUBLIC

When we ran the migration job for the first time it failed with the following abend.

```
CSQ45ATB MIGRDSG - COMPLETION CODE - SYSTEM=5C6 USER=0000 REASON=00C60006
```

The code 00C60006 says that message table CSQFSTAB could not be loaded. Upon investigation we found that this WebSphere MQ V6 loadmod was not found in MQS.V6R0M0.SCSQANLE.PETCPYA. We contacted the support center who identified a V5.3.1 techdoc which related to the same problem seen here in V6R0M0. We also were advised to install ptf UK06892. The resolution was to do a APPLY REDO of the 2 MQ FMIDS. This information can all be found at:

```
http://www.ibm.com/support/docview.wss?rs=171&context=SSFKSJ&q1=csqfstab&uid=swg21207831&loc=en_US&cs=utf-8&lang=en
```

We applied the PTF and also performed the SMP/E APPLY REDO for the MQ FMIDS. We then copied over the libraries again to our local DASD. We submitted the migration job again from the top for QSG MQGT and it ran successfully. The next step was to start up a queue manager using the new libraries. To do this without an ipl required the following.

- APF authorize the required libraries. (We had done this before running the migration job)
- Add MQ libraries to the STEPLIBs of the CSQx started tasks

```
//STEPLIB   DD DSN=MQS.V6R0M0.SCSQANLE.PETCPYA,DISP=SHR
//          DD DSN=MQS.V6R0M0.SCSQAUTH.PETCPYA,DISP=SHR
//          DD DSN=MQS.V6R0M0.SCSQLOAD.PETCPYA,DISP=SHR
//          DD DSN=MQS.V6R0M0.SCSQLINK.PETCPYA,DISP=SHR
//          DD DSN=MQS.LOCAL.SCSQAUTH.V6R0M0,DISP=SHR
```

- Add early code modules to LPA

```
LPA ADD  MODNAME=(CSQ3INI,CSQ3EPX),
     DSNAME(MQS.V6R0M0.SCSQLINK.PETCPYA)
 LPA ADD  MODNAME=(CSQ3ECMX),
     DSNAME(MQS.V6R0M0.SCSQSNLE.PETCPYA)
```

- Refresh the Early Code which MQ uses at startup.

```
!MQZ1 REFRESH QMGR TYPE(EARLY)
```

After this was done we started up the qmgr on system Z1. This system is one of 3 systems in the MQGT QSG. We then ran a test messaging workload and the upgrade looked to be fine with running mixed levels of WebSphere MQ V5R3M1 and V6R0M0 with the MQGT QSG. We then completed the migration to V6 for the other 2 qmgrs in the MQGT QSG.

A few weeks later we did the same migration on the rest of the images in the sysplex of which 7 participated in QSG MQGP. Once we brought up the qmgr's on a couple systems we hit a failure and received the following messages:

```
CSQJ151I !MQJH0 CSQJR003 ERROR READING RBA  000000277000, CONNECTION-ID=CSQH CORRELATION-ID=003.RCRSC 02
REASON  CODE=00D10345
CSQV086E !MQJH0   QUEUE MANAGER ABNORMAL TERMINATION REASON=00D96021
IEF450I CSQHMSTR CSQHMSTR - ABEND=S6C6 U0000 REASON=00000000 TIME=08.36.42
```

After investigating it looked to be that MQ V6 is stricter than V5.3.1 was in that during startup it looked to confirm that there are ARCHIVE logs for Page sets that are defined but are offline. We found that in the CSQxMSTR JCL for startup we had defined PSIDs for 5,6,7 in the JCL . We received the following messages that they were offline:

```
CSQI005I !MQJH0 CSQIECUR PAGE SET 5 OFFLINE
CSQI005I !MQJH0 CSQIECUR PAGE SET 6 OFFLINE
CSQI005I !MQJH0 CSQIECUR PAGE SET 7 OFFLINE

CSQI024I !MQJH0 CSQIDUSE Restart RBA for system as configured=0000532FA226
CSQI025I !MQJH0 CSQIDUSE Restart RBA including offline page sets=00000027789F
```

From what we can tell, MQ V6 now looks to confirm that the archived logs exist for page sets that had been offline. We took out the DD card for PSID's 5,6,7 and started up the qmgr again. The page sets had been added at a prior time for a specific workload test and were no longer needed. This time the qmgr started successfully. At this point we completed the migration of all QMGR's on the plex and have been running our various test messaging workloads successfully.

Using Websphere Message Broker

We have recently migrated from WebSphere Business Integration Message Broker V5.0. to Websphere Message Broker V6.0. Our experiences in testing this version will be in the next version of this document.

Note: To simplify the discussion, we'll refer to WebSphere Message Broker as WMB. Abbreviations are not officially sanctioned by IBM, so you should not use them to try to locate information or for product ordering, for instance.

Updating the Retail_IMS workload for workload sharing and high availability

In an effort to make our broker domain more complex and introduce workload sharing and high availability to our workloads, we created another broker for a total of two brokers on our production systems. We altered the Retail_IMS workload to

utilize both brokers (workload sharing) and to continue processing on one broker if the other one goes down (high availability).

Description of the workload

The Retail_IMS workload uses WebSphere Application Server 6.0 to host a Web front end (html page and java servlet) to receive information from the user. This information is rolled up into a message and placed on the RETAIL.IMS.IN queue, which a broker message flow is monitoring. The message flow extracts some fields from the message, adds an IMS header, and puts the new message on the RETAIL.IMS.OUT queue. The java servlet then takes the message from the RETAIL.IMS.OUT queue and passes it to an html page for display. Any failures message processing result in a message on the RETAIL.IMS.FAIL queue.

Thus, there are three queues that are used in this workload:

1. RETAIL.IMS.IN - holds the input message to the message flow
2. RETAIL.IMS.OUT - holds the output message from the message flow when normal processing occurs
3. RETAIL.IMS.FAIL - holds the output message from the message flow when abnormal processing occurs

Changes to the workload

The MQReply node in the message flow enables you to have multiple candidates for the output queue without having multiple message flows (one for each client). For this reason we decided to make the output queues non-shared and use unique names per client.

Our message flow now looks like:

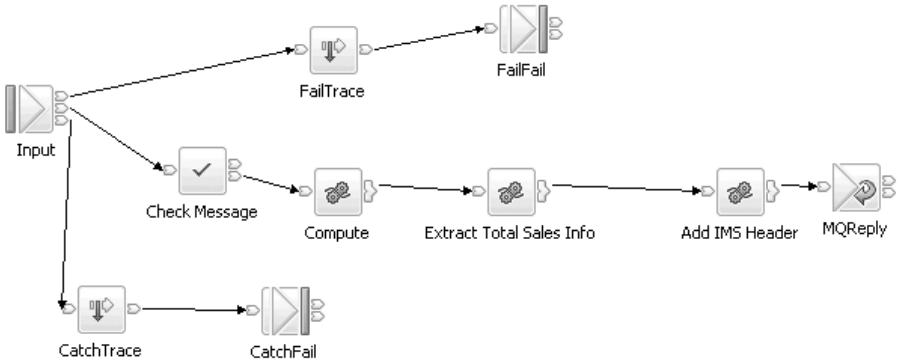


Figure 54. Retail_IMS workload message flow.

We made the following changes queue definitions for the workload.

We created unique output queues for each client putting request messages to the input queue. We made these clustered queues and not shared queues.

```
QUEUE(RETAIL.IMS.OUTJ90)
TYPE(QLOCAL)
QSGDISP(QMGR)
STGCLASS(WMQI)
CFSTRUCT( )
CLUSTER(MQBROKER.CLUSTER)
```

As a result of these changes, the Retail_IMS workload now utilizes both brokers, alternating between brokers for each transaction by using the round robin functionality of MQ clustering. Additionally, if one of the two brokers fails, all of the messages are then processed by the other broker, and if the failed broker returns, the messages again round robin between the brokers. The end result is a workload that utilizes workload sharing and provides some level of high availability.

Additional information about this topic can be found in the *WebSphere Business Integration Message Broker and high availability environments* located at:

`http://www.ibm.com/developerworks/websphere/library/techarticles/0403_humphreys/0403_humphreys.html`

Some useful Web sites

We found the following Web sites useful when working with WebSphere Business Integration Message Broker:

- README files for WebSphere MQ family products: www.ibm.com/software/integration/mqfamily/support/readme/
- Fix Packs for WebSphere Business Integration Brokers: www.ibm.com/software/integration/mqfamily/support/summary/wbib.html
- SupportPacs for WebSphere MQ family products: www.ibm.com/software/integration/support/supportpacs/product.html

  Note: We found SupportPac IP13 for WebSphere Business Integration Brokers to be particularly useful.
- The Websphere Message Broker V6 documentation can be found at:

  `http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/index.jsp`
- IBM Redbooks: www.ibm.com/redbooks/

Chapter 15. Using IBM WebSphere Application Server for z/OS

This chapter describes our experiences using IBM WebSphere Application Server for z/OS and related products. Our test environment is now fully migrated to WebSphere Application Server for z/OS V6.0 running on z/OS V1R7. See "Migrating to WebSphere for z/OS V5.1 to V6" in our previous test report for information on our migration.

Note: References to WebSphere Application Server for z/OS V6.*x* appear in the text as ″WebSphere for z/OS V6.*x*″ or simply ″V6.*x*.″

About our z/OS V1R7 test environment running WebSphere Application Server

In this chapter, we provide a level-set view of our current test environment and provide details about the changes we've made and our experiences along the way.

Our z/OS V1R7 WebSphere test environment

This section provides an overview of our z/OS V1R7 WebSphere test environment, including the set of software products and release levels that we run, the Web application configurations that we support, and the workloads that we use to drive them.

Current software products and release levels

The following information describes the software products and release levels that we use on the z/OS platform and on the workstation platform.

Software products on the z/OS platform: In addition to the elements and features that are included in z/OS V1R7, our WebSphere test environment includes the following products:

- WebSphere Application Server for z/OS Version 6.0.2, service level cf80608.10
- IBM SDK for z/OS, Java 2 Technology Edition V1.4.2 (April 21, 2006 Build Date, PTF UK13868)
- WebSphere Studio Workload Simulator V1.0
- WebSphere MQ for z/OS V6
- WebSphere Message Broker V6
- DB2 V8.1 with JDBC
- CICS TS 3.1
  - CICS Transaction Gateway (CICS TG) V6.0
- IMS V9 with IMS Connector for Java V9
  - IMS Connector for Java V9.1.0.1

Software products on the workstation platform: Software products on the workstation platform: On our workstations, we use the following tools to develop and test our Web applications:

- Rational Application Developer Version 6.0.1.1
- IBM WebSphere Developer for zSeries Version 6.0.1
- WebSphere Studio Workload Simulator V1.0

Our current WebSphere Application Server for z/OS configurations and workloads

The following are our current WebSphere Application Server for z/OS configurations and workloads.

Configuration update highlights: We made the following updates to our test and production configurations:

- Migrated cells to WebSphere Application Server for z/OS V6.0
- Added an additional test cell (T3)
- Migrated from Trade3 to Trade6
- Added our zBank application (and J2EE server 7 for it)
- Added eWLM monitoring
- Security enhancements (TAM, TAI++, WebSeal on zLinux)

Our test and production configurations: In our environment, we have fully migrated to WebSphere for z/OS V6.0.2. Our current V6.0.2 setup contains five cells: T1, T2 and T3 for our test systems, P1 for our WebSphere Application Server for z/OS production systems and QP for WebSphere Application Server for z/OS applications used by MQ team. All cells are configured as network deployment cells.

Our T1 cell is configured as follows:

- Resides entirely on one of our test systems (Z1)
- Contains six different J2EE servers, each running different applications (as described below)

Our T2 cell is configured as follows:

- Generally resides on one of our test systems (Z2), but also has nodes configured on two additional systems (JB0 and JH0)
- Contains six different J2EE servers, each running different applications (as described below)

Our T3 cell is configured as follows:

- Resides entirely on one of our test systems (Z3)
- Contains seven different J2EE servers, each running different applications (as described below)

Our P1 cell is configured as follows:

- Spans four production systems in our sysplex (J80, JB0, JF0, and JH0)
- Contains six different clusters, each of which spans all four systems. Each cluster contains four J2EE servers—one J2EE server per system.
- Each cluster corresponds to one of the single J2EE servers in our T1/T2 cell. Initially, we configure and deploy applications on a test J2EE server in the T1 and/or T2 cell and then deploy them to the corresponding server cluster in the P1 cell.

Our QP cell is configured as follows:

- Spans two production systems in our sysplex (JC0 and J90)
- Contains two different clusters, each of which spans both systems. Each cluster contains two J2EE servers—one J2EE server per system.
- Each cluster hosts various applications that connect WebSphere Application Server for z/OS to MQ as used by the MQ team.

Our Web application workloads: The following applications run in the J2EE servers on our T1, T2 and P1 cells:

- J2EE server 1 runs our workload monitoring application. The application accesses only z/OS UNIX System Services files.
- J2EE server 2 runs our bookstore application, accessing DB2 and WebSphere MQ
- J2EE server 3 runs the Trade6 application, accessing DB2 and WebSphere MQ
- J2EE server 4 runs our PETRTWDB2 application, accessing DB2
- J2EE server 5 runs our PETDSWIMS application, accessing IMS
- J2EE server 6 runs our PETNSTCICS application, accessing CICS

The following application runs in the J2EE Server on our T3 cell in addition to the above six applications:

- J2EE server 7 runs our zBank application used for security testing and accessing DB2

Figure 55 on page 200 shows the server address spaces in our P1 cell.

Note: The wsp1s1 cluster is not shown in the diagram.

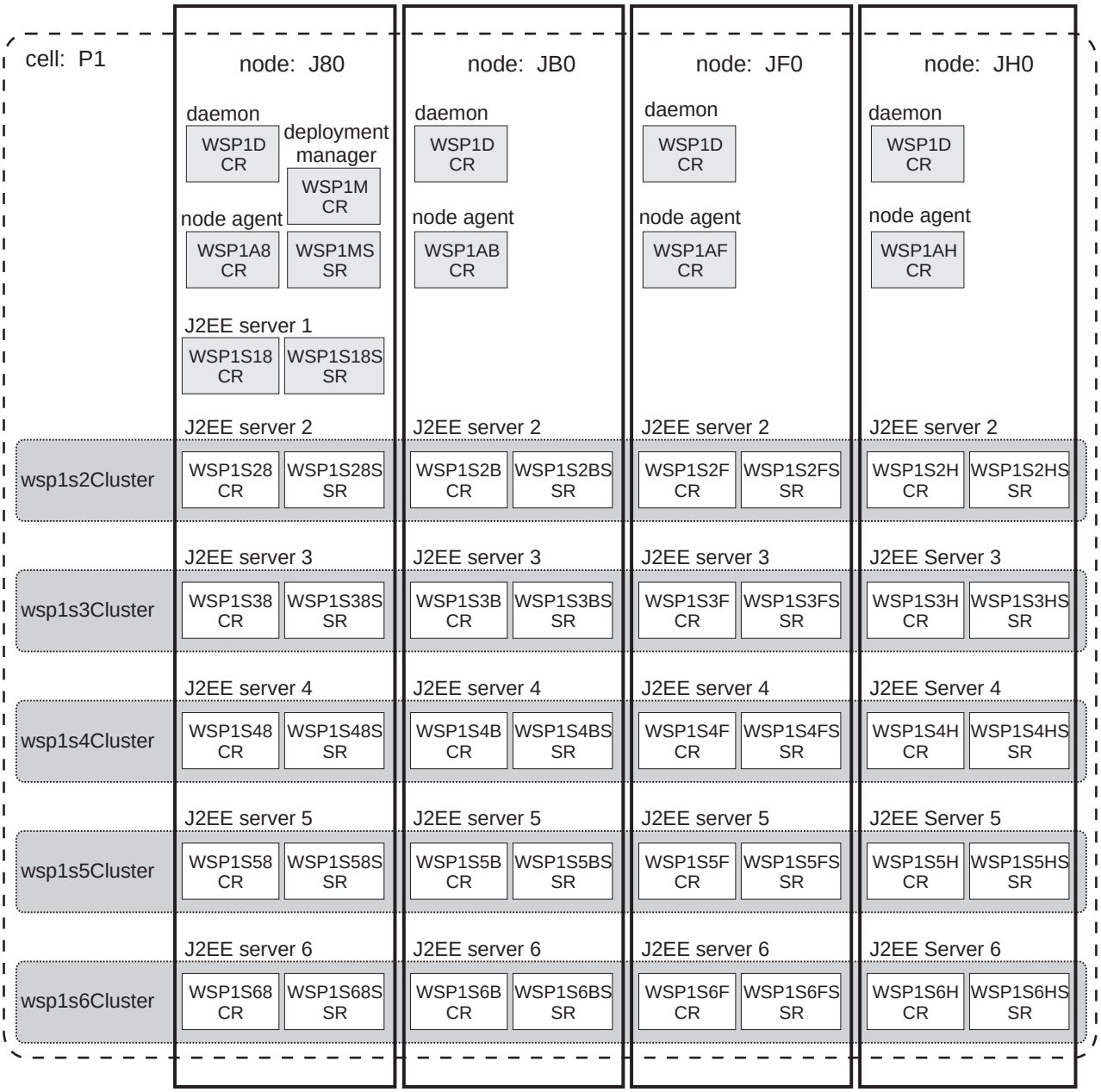


Figure 55. Our WebSphere for z/OS V6 configuration

About our naming conventions: After some experimentation, we settled upon a naming convention for our WebSphere setups. Our address space names are of the following format:

`WS`*`ccs`*`[`*`n`*`]`*`y`*`[S]`

where:

WS The first two characters are always “WS” to identify a WebSphere resource.

cc Cell identifier:
- **T1** Test cell 1
- **T2** Test cell 2
- **P1** Production cell 1
- **QP** MQ Team Production cell

s[*n*] Server type. For J2EE server control regions and server regions, *n* is the instance number of the server within the node:
- **A** Node agent
- **D** Daemon
- **M** Deployment manager
- **S***n* J2EE server control region, instance *n*

y System identifier:
- **1** Z1 (test)
- **2** Z2 (test)
- **8** J80 (production)
- **B** JB0 (production)
- **F** JF0 (production)
- **H** JH0 (production)

[S] Servant flag. This is appended to the name of a J2EE server control region to form the name of the associated servant region(s).

Example: The name WSP1S18S indicates a **W**eb**S**phere **p**roduction cell **1** J2EE server **s**erver region **1** on system J**8**0.

Server short names are specified in upper case. Server long names are the same as the short names, but are specified in lower case.

Other changes and updates to our WebSphere test environment

The following describe other changes and updates to our WebSphere test environment.

Setting up WebSphere for eWLM monitoring of DB2 applications

We have eWLM V2 for z/OS installed with a z/OS Domain Manager, Firewall Broker, and several z/OS Managed Servers. The intent was to use eWLM to monitor WebSphere Application Server applications that access DB2 through DDF. Figure 56 on page 202 is a diagram of our setup.

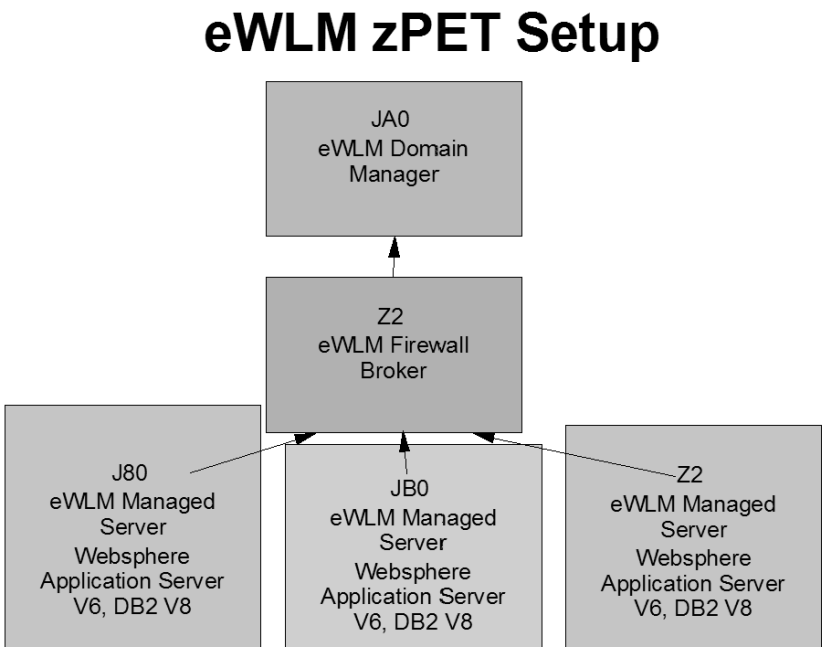


Figure 56. eWLM zPET setup

To set this up we followed the instructions located at:

`http://publib.boulder.ibm.com/infocenter/eserver/v1r2/index.jsp`

The eWLM documentation is under:

IBM Systems Software Information Center
Virtualization
Management collection
Enterprise Workload Manager

Setup instructions for WebSphere Application Server and DB2 can be found by searching for ″Enabling ARM on WebSphere Application Server 6.0″ and ″Enabling ARM on DB2 Universal Database for use by WebSphere Application Server on z/OS″.

In order to see WebSphere Application Server and DB2 transactions in the eWLM Control Center the following needs to be done:

1. Configure the WebSphere Application Servers for arm4 and enable request metrics
2. Ensure the DB2 DDF parameters are defined so that the application HOPs from WebSphere Application Server to DB2 are reflected. The DB2 ZPARM **CMTSTAT** must be set to INACTIVE. The DB2 command *DIS DDF DETAIL* will show you the current settings as shown below in the sample output.

```
-@DB81 DIS DDF DETAIL
 DSNL080I  @DB81 DSNLTDDF DISPLAY DDF REPORT FOLLOWS:
 DSNL081I STATUS=STARTD
 DSNL082I LOCATION           LUNAME            GENERICLU
 DSNL083I USIBMT6PETDB2      USIBMT6.DB2DB81   USIBMT6.PETDB2
 DSNL084I IPADDR          TCPPORT RESPORT
 DSNL085I 192.168.25.180      446     6021
 DSNL086I SQL    DOMAIN=TORDDFSD.TOR.IBM.COM
```

```
DSNL086I RESYNC DOMAIN=J80EIP.TOR.IBM.COM
DSNL090I DT=I  CONDBAT=    500 MDBAT=  500
DSNL092I ADBAT=    3 QUEDBAT=      0 INADBAT=      0 CONQUED=      0
DSNL093I DSCDBAT=      2 INACONN=    262
DSNL099I DSNLTDDF DISPLAY DDF REPORT COMPLETE
```

We use eWLM to view transaction statistics for our Trade application which uses WebSphere Application Server and DB2. Figure 58 on page 204 shows what type of information is provided.

Note: You need to have the Adobe SVG viewer installed to get the next view. You can download it from

`http://www.adobe.com/svg/viewer/install/main.html`

To get the view shown in Figure 57, go to “monitor” then “Transaction Classes”. Select the appropriate Transaction Class then “Application Topology” from the pull-down.

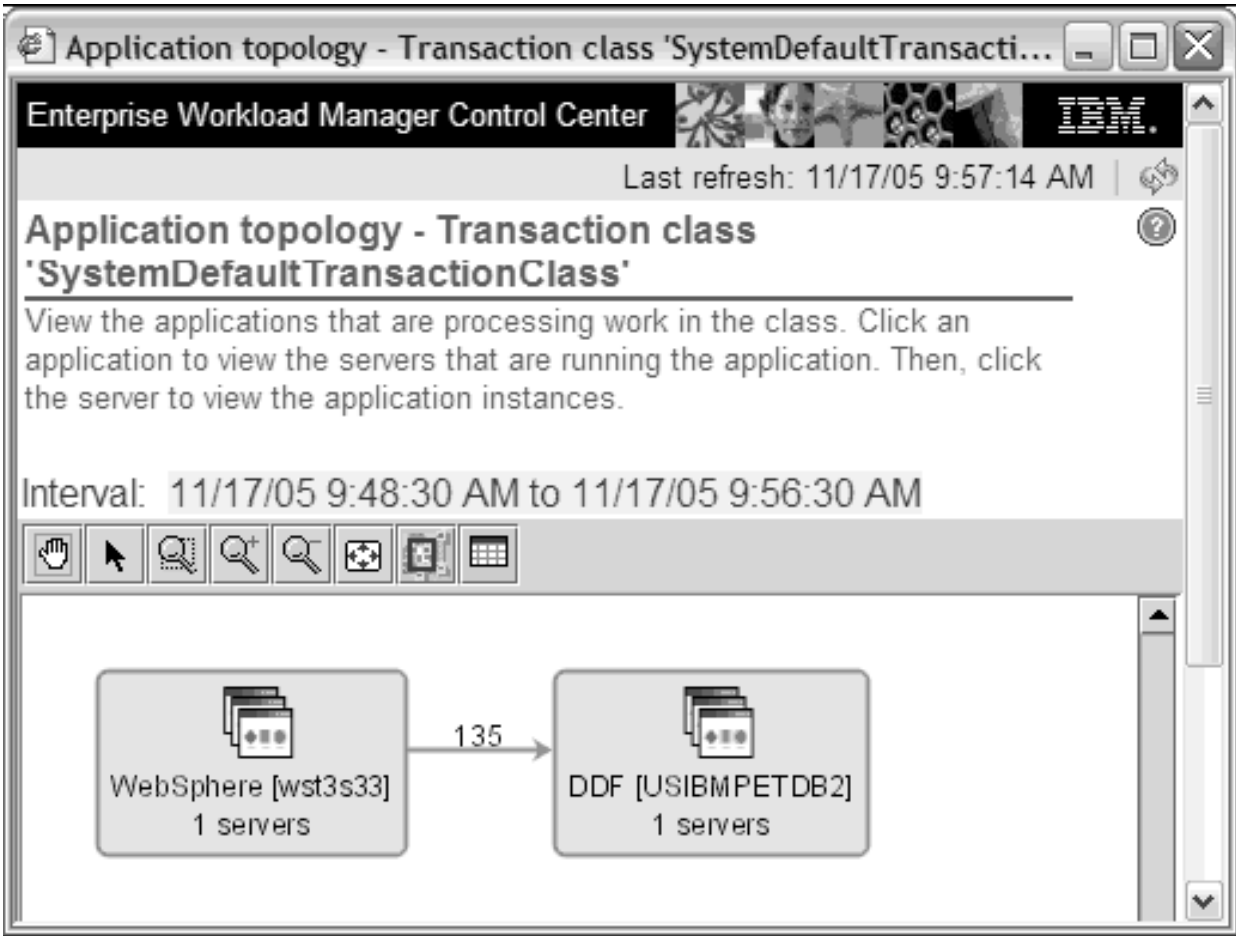


Figure 57. eWLM Control Center

If you click on the table button you will get the view showing Figure 58 on page 204

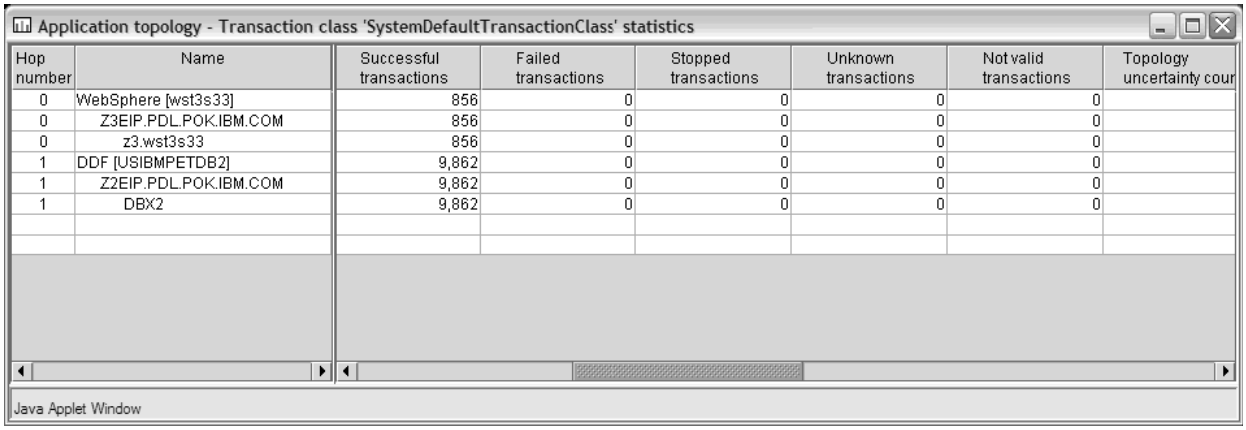


Figure 58. 'SystemDefaultTransactionClass' statistics

Defining JMS and JDBC Resources for Trade6

Trade6 is a benchmarking application that can be found in the "Downloads" section at:

`http://www.ibm.com/software/webservers/appserv/was/performance.html`

We use it to test the Websphere Application Server connections to DB2, WebSphere MQ, and the WebSphere MQ Integration Message Broker, all on z/OS. It simulates a stock trade application. See the website for details on the application.

There are four types of resources that need to be defined:

1. DB2 providers and data sources
2. Websphere MQ Queue Connections
3. WebSphere MQ Topic Connections for the Message Broker pub/sub functions
4. Define listener ports

Setting up the resource

We chose to use a DB2 UDB Type 4 connection to our z/OS DB2 subsystem. We have a sysplex distributor VIPA address defined to a 4-way datasharing group for the Trade6 tables. We also use session management to DB2 tables and setup a UDB Type 2 connection for that just to test the two different types of UDB connections.

Setting up DB2

The steps taken in the Websphere Admin Console are as follows:

Go to **Resources -> JDBC Providers**

1. Add a DB2 Universal JDBC
   - Set the classpath to
     - ${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc.jar
     - ${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cisuz.jar
   - The native library path is
     - ${DB2UNIVERSAL_JDBC_DRIVER_NATIVEPATH}
   - The implementation class name is
     - com.ibm.db2.jcc.DB2ConnectionPoolDataSource

2. Add a data source for the trade6 tables with the following settings.

   Note: Only non-default values are listed.

   a. Add name ″TradeDataSource″ with jndi name ″jdbc/TradeDataSource″
   b. Check the box for ″Use this Data Source in container managed persistence (CMP)″
   c. Select datastore helper ″class db2.universal.datastore.helper″
   d. Set Component-managed authentication alias to ″none″
   e. Set Container-managed alias to a valid userid with authority to connect to DB2.
   f. Set Mapping-configuration alias to ″default-principal mapping″
   g. Set Database name to ″USIBMDBWGDB2″.

      Note: The db2 command 'dis ddf' can be used to determine location name which is the value required here.

   h. Set the Driver to type 4
   i. Set the Server name to the DB2 ″vipa address″ defined for the data sharing group.

3. **Additional Properties -> Custom Properties**
   a. currentSQLID is set to the server started task user WAS5SSR3
4. Add a data source for session tables
   a. Set Name to ″SessionTable″ with jndi name ″jdbc/SessionTable″
   b. Specify a user-defined data store helper = ″com.ibm.websphere.rsadapter.DB2390DataStoreHelper″
   c. Select component and container managed alias with default-principal mapping
   d. Set the database name = USIBMDBWGDB2
   e. Set the driver to type 2
   f. Server name left blank
5. Additional Properties
   a. Set currentSQLID to the session table creator, ″WAS″

Setting up Websphere MQ resources

We defined a connection to a local queue manager using bindings mode. The broker and queue manager are on the same system as the WebSphere Application Server running the trade6 application. We have two brokers, one on a different LPAR. The broker queues are defined as shared queues so either broker can procress the pub/sub messages.

To define the connections go to **Resources -> JMS providers -> Websphere MQ**. Our local names are shown here.

Setup a JMS provider for Websphere MQ:

1. Leave the General properties as the defaults
2. Under Additional Properties:
   a. Define a WebSphere MQ queue connection factories
      1) Name = TradeBrokerQCF with jndi name /jms/ TradeBrokerQCF
      2) Select component and container manages aliases and set to a user with default principal mapping
      3) Qmgr = CSQ8

4) Host = j80.pok.ibm.com
5) Port = 0 for bindings mode
6) Transport type bindings
7) Model queue definition = SYSTEM.JMS.MODEL.QUEUE
8) Leave the rest as defaults

b. Define a queue destination
1) Name = TradeBrokerQueue with jndi name jms/ TradeBrokerQueue
2) Base queue name = TradeBrokerQueue
3) Base qmgr name = CSQ8
4) The rest are defaults

c. Define a Topic Connection Factory
1) Name = TradeStreamerTCF with jndi name jms/ TradeStreamerTCF
2) Select component and container manages aliases and set to a user with default principal mapping
3) Qmgr = CSQ8
4) Port = 0
5) Broker Control Queue = SYSTEM.BROKER.CONTROL.QUEUE
6) Broker qmgr = CSQ8
7) Broker Publication Queue = TRADE6.JMS.PUBLISH.QUEUE.J80
8) Broker Subscription Queue = SYSTEM.JMS.ND.SUBSCRIBER.QUEUE
9) Broker CC subscription queue = SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE
10) Broker version – ″advanced″
11) Model queue definition = SYSTEM.JMS.MODEL.QUEUE
12) The rest are defaults

d. Define a topic destination
1) Name = TradeStreamerTopic with jndi name jms/ TradeStreamerTopic
2) Base topic name = TradeStreamerTopic
3) Broker durable subscription queue = SYSTEM.JMS.D.SUBSCRIBER.QUEUE
4) Broker CC durable subscription queue = SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE
5) The rest are defaults

e. Add listener ports
- Go to servers -> Communications -> Messaging -> Listener Ports
- Add TradeListenerPort
- Connection factory JNDI name jms/TradeBrokerQCF
- Destination JNDI name jms/TradeBrokerQueue
- Add TradeStreamerPort
- Connection Factory jndi name jms/TradeStreamerTCF
- Destination jndi name jms/TradeStreamerTopic

Setting up environment variables

The following are the steps we took to set up our environment variables:

1. Go to **Environment -> WebSphere Variables**
   a. Define server environment variables

1) DB2UNIVERSAL_JDBC_DRIVER_NATIVEPATH = /db2810/currentdbwg/jcc/lib
2) DB2UNIVERSAL_JDBC_DRIVER_PATH = /db2810/currentdbwg/jcc/classes
3) MQJMS_LIB_ROOT ${(MQ_INSTALL_ROOT)}/java/lib
4) MQ_INSTALL_ROOT /was60p1/jmssmpe (The WebSphere MQ java library path)

2. Go to **Application servers > wsp1s38 > Process Definition > Servant > Java Virtual Machine > Custom Properties**
   a. Define a JVM custom property file for JCC
      1) Create db2.jcc.propertiesFile and set to /db2810/dbwg_db2jcc.properties

      This has the statement ″db2.jcc.ssid=DBWG″ which is the DB2 data sharing group name.

Providing authentication, course-grained security, and single sign-on for Web/EJB based applications running on WebSphere Application Server for z/OS

This section demonstrates the use of a single-sign on solution using Tivoli Access Manager (TAM)'s implementation of the Trust Association Interceptor (TAI) interface that is being shipped with WebSphere Application Server for z/OS 5.1.x and 6.0.x. This article is based on a developerWorks article titled "Tivoli Access Manager Trust Association Interceptor (TAI++)" found at:

`http://www.ibm.com/developerworks/tivoli/library/t-tamtai/`

The goal of this scenario was to use TAM to provide authentication, coarse-grained security, and single sign-on for our Web/EJB-based applications running on WebSphere Application Server for z/OS, while using RACF to do fine-grained role-based authorization checking.

Our scenario is based on WebSphere Application Server's support for single sign on through 'Trust Associations' with perimeter authentication services (such as reverse proxies – in our case, WebSEAL).

The perimeter authentication service is expected to:

- Establish ″trust″ with WebSphere Application Server for z/OS
- Perform user authentication
- Insert user credential information into HTTP requests that are then forwarded to WebSphere Application Server for z/OS

WebSphere Application Server for z/OS provides an interface that allows the configuration of a pluggable module called a Trust Association Interceptor (TAI). The job of the TAI is to provide a trustworthy identity to WebSphere using the content of a request that has been forwarded by a perimeter authentication service. The TAI is expected to:

- Validate trust with the perimeter authentication service
- Extract credential information from the request

The Tivoli Access Manager (TAM) Trust Association Interceptor++ module that comes with WebSphere Application Server for z/OS 5.1.1 and 6.0, allows a WebSphere credential (which is used for WebSphere authorization checking) to be built from the Access Manager credential sent by WebSEAL. Previous versions of the WebSEAL Trust Association Interceptor simply provided the trusted userId to

WebSphere as a string, thereby requiring additional registry searches by WebSphere to construct the WebSphere credential.

In our case, Tivoli Access Manager and WebSphere Application Server for z/OS are using two different registries. TAM is using an LDAP on z/OS registry, while WebSphere Application Server for z/OS is configured to use the local OS (RACF) registry. In this case, WebSphere Application Server uses the Subject returned from the TAI to construct a 'Platform Credential,' obtained from RACF. This allows WebSphere Application Server for z/OS to do role-base authorization checking, using the RACF EJBROLE CLASS, with the user's RACF credentials. Figure 59 shows an overview of this environment.

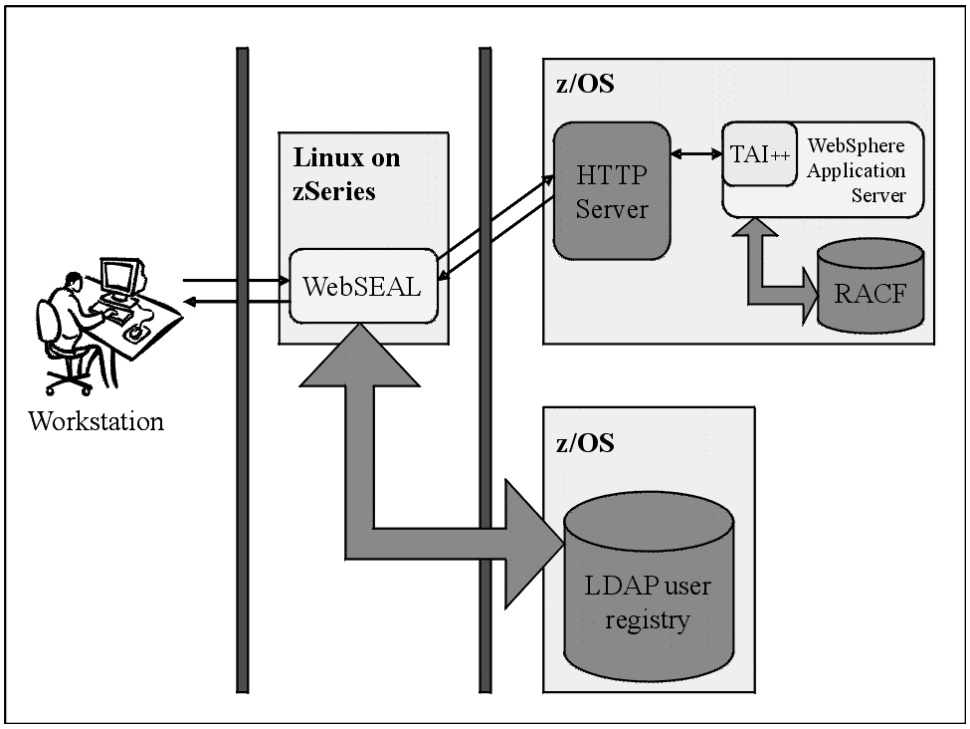


Figure 59. Our TAI++ trust association environment

The following sections describe:

- "Usage scenario"
- "Setting up our TAM scenario" on page 210

Usage scenario

Our scenario begins with a user attempting to access their online banking application, zBank. The zBank web site is a J2EE application running in a WebSphere Application Server for z/OS. The user's web requests first go through a reverse proxy server (WebSEAL)

For this transaction flow to work, the user's TAM ID in LDAP must be identical to their RACF ID. The passwords do not need to be the same, and, unless there is some other reason to do so, the user never needs to know their z/OS RACF password.

The flow of a client's HTTP request to the zBank application is shown in Figure 60.

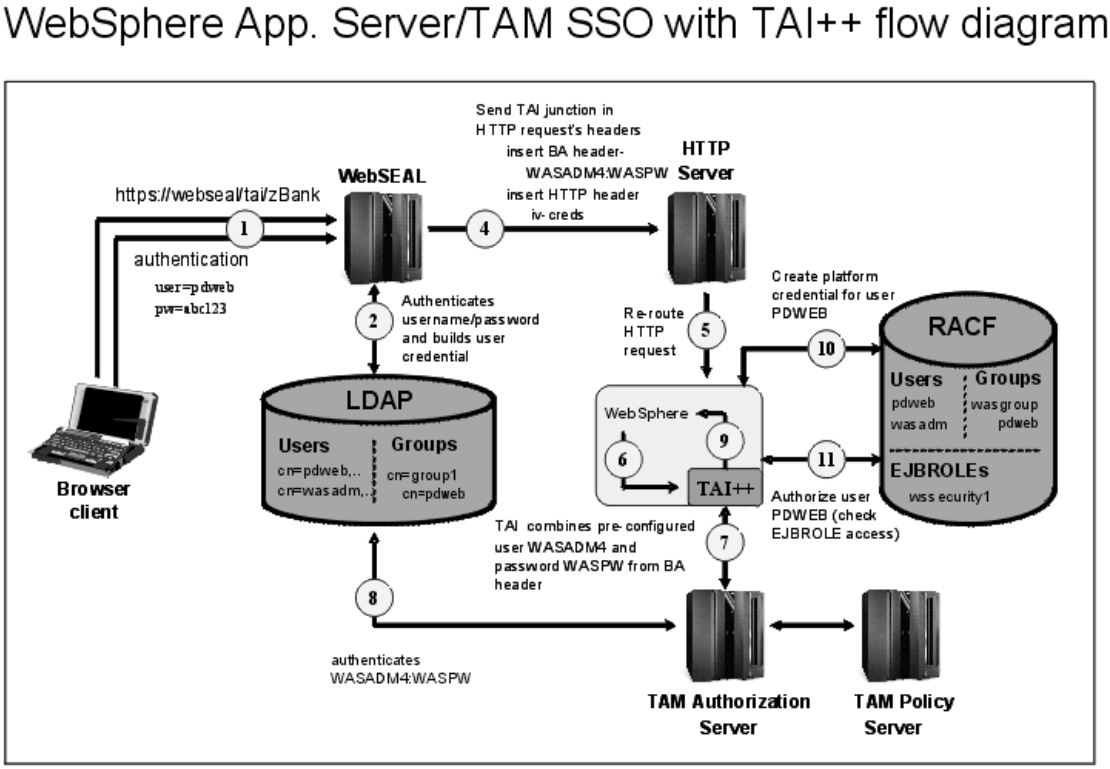


Figure 60. Flow of a client's HTTP authenticationrequest to the WebSphere Application Server application

Following is a detailed description of that flow:

1. The client accesses WebSEAL and requests a URL, containing a 'junction' identifier which has been defined to WebSEAL as a pointer to the HTTP Server sitting in front of WebSphere. The URL is protected by a TAM authorization policy (in our case, an ACL), which results in WebSEAL sending an authentication request to the client. The client then enters a username and password.
2. WebSEAL authenticates the user, and acquires credentials for the user from the TAM user registry - in our case, an LDAP on z/OS registry.
3. WebSEAL checks the authorization policy that has been placed on the URL - and authorizes or denies access. We'll assume the user is granted access to the URL.
4. WebSEAL routes the requested URL to the HTTP Server sitting in front of WebSphere. In the request, WebSEAL creates an HTTP basic authentication (BA) header with a user and password - the user & password values are specified when the 'junction' is defined. WebSEAL also adds an additional HTTP header (iv-creds) containing the user credentials of the client.
5. The HTTP Server routes the request to WebSphere Application Server for z/OS.

6. WebSphere Application Server for z/OS receives the request and calls a TAI method to determine if the request is from a trusted perimeter authentication service.
7. The TAI takes the password from the BA header in the request, combines it with a userid that was specified during configuration of the TAI++ module. An authentication request with this userid/password combination is sent to the TAM authorization server, to validate that WebSEAL is the origin of the iv-creds HTTP header.
8. The TAM Authorization Server authenticates the userid/pw sent from WebSphere Application Server for the TAI module using the TAM registry (LDAP on z/OS).
9. With successful authentication, and the existence of the iv-creds header, the TAI considers the perimeter authentication service 'trusted'. The TAI extracts the iv-creds and returns a Subject to WebSphere Application Server - the iv-creds credentials cannot be used directly since they came from LDAP, and our WebSphere Application Server instance is using the local-OS registry - RACF.
10. WebSphere Application Server maps the Subject to a RACF user, creating a 'Platform Credential', which can later be used for role-base authorization checks.
11. A RACF check is performed based on the user's credentials and the RACF EJBROLE profile for the application being accessed.

    Note: In our case, the user must also have access to an APPL class profile, allowing access to WebSphere Application Server applications.
12. Authorization is granted, and the user accesses the application.

Setting up our TAM scenario

Our setup consisted of the following steps:

1. "Configuring the Tivoli Access Manager Java Runtime in WebSphere Application Server for z/OS"
2. "Configuring a Tivoli Access Manager Java Server" on page 211
3. "Defining the Trust Association Interceptor" on page 212
4. "Creating the WebSEAL junction" on page 215
5. "Creating the users in TAM and RACF to access the WebSphere Application Server for z/OS application" on page 216

Configuring the Tivoli Access Manager Java Runtime in WebSphere Application Server for z/OS: The first step is to configure the Tivoli Access Manager Java Runtime in WebSphere Application Server for z/OS. This is done by running the PDJrteCfg command, using the PD.jar file shipped with WebSphere Application Server:

Commands:

```
export CLASSPATH=/was51/t2java/lib/ext/PD.jar:$CLASSPATH
export JAVA_HOME=/was51/t2java
export PATH=/was51/t2java/bin:$PATH

java -Dfile.encoding=ISO8859-1 -Dws.output.encoding=CP1047 -Xnoargsconversion \
-Dpd.home=/was51/t2cfg/DeploymentManager/java/jre/PolicyDirector –cp \ /was51/t2cfg/DeploymentManager/java/jre/lib/ext/PD.jar \
com.tivoli.pd.jcfg.PDJrteCfg -action config -cfgfiles_path \ /was51/t2cfg/DeploymentManager/java/jre -host 192.168.20.127 -was
```

Output:

```
Configuration of Access Manager Java Runtime Environment is in progress.
This might take several minutes.
Configuration of Access Manager Java Runtime Environment completed success
```

In the *PDJrteCfg* command above:

- *-cfgfiles_path* points to the location of the PolicyDirector directory in WebSphere, where the Tivoli Access Manager configuration files are kept - this is typically the JAVA_HOME directory for the WebSphere Application Server instance
- *-host* points to the hostname or IP address of the Policy Server for the Tivoli Access Manager domain
- *-was* indicates that you're using the Tivoli Access Manager Java Runtime shipped with WebSphere Application Server

As a result of this command, a few properties files and a keystore file are generated in the WebSphere Application Server instance's $JAVA_HOME/PolicyDirector directory:

Command/Output:

```
147:/was51/t2cfg/DeploymentManager/java/jre/PolicyDirector $ ls
PD.properties      PDCA.ks            bin                log
PD.properties.old  PDJLog.properties  etc                nls
```

Configuring a Tivoli Access Manager Java Server: Configuring the TAM Java Server will create the properties and key file necessary for the Trust Association Interceptor to contact the TAM Authorization Server in order to authenticate the WebSEAL making the TAI request.

A TAM Java Server is configured by running the *SvrSslCfg* command.

Command:

```
The following command was run from the directory - /was51/t2cfg/DeploymentManager/java/jre

java -cp $CLASSPATH -Dpd.cfg.home=/was51/t2cfg/DeploymentManager/java/jre -Dfile.encoding=ISO8859-1
-Dws.output.encoding=CP1047 -Xnoargsconversion  \ com.tivoli.pd.jcfg.SvrSslCfg -action config
-admin_id sec_master -admin_pwd \ linux390  -appsvr_id tai -host z2.pok.ibm.com -port 7777
-policysvr \ 192.168.20.127:7135:1 -authzsvr 192.168.20.127:7136:1 -mode remote -cfg_file \ PDPerm.properties
-key_file tai.ks -cfg_action create
```

Output:

```
The configuration completed successfully.
```

In the *SvrSslCfg* command above:

- *-admin_id* is the TAM administrator ID (sec_master)
- *-admin_pwd* is the TAM administrator password (linux390)
- *-appsvr_id* is the name we gave to our TAM Java server (tai)

- *-host* is the hostname where we defininied the TAM Java server (our current hostname is z2l.pok.ibm.com)
- *-port* is the port where the TAM Java server listens for Policy Server notifications
- *-policysvr* gives the host or IP address, port, and rank of the Policy Server in the TAM domain
- *-authzsvr* is the host name or IP address, port, and rank of the Authorization Server the TAI will contact to make the authorization decision to determine if WebSEAL should be 'trusted'
- *-mode* indicates the Java Server is using a remote Authorization Server for authentication/authorization decisions
- *-cfg_file* names the properties file defining the Java Server. Since we didn't specify a full path, it's created in our current directory. This properties file will be specified in the WebSphere Admin console definition we create for the TAI.
- *-key_file* names the key file used by the Java Server to communicate with other TAM servers using mutually authenticated SSL.

In the directory where we ran the SvrSslCfg command, we now see the properties file (PDPerm.properties) and key file (tai.ks) that were generated:

```
Command/Output:

135:/was51/t2cfg/DeploymentManager/java/jre $ ls
PDPerm.properties  PolicyDirector     lib               tai.ks
```

Now, by using the TAM command line administration utility 'pdadmin', we did a 'server list' command, and the 'tai' server we just created by running SvrSslCfg shows up in the list of servers:

```
Command/Output:

pdadmin sec_master> server list
    ivacld-am60
    default-webseald-metlnx30
    tai-z2.pok.ibm.com
```

Defining the Trust Association Interceptor: We defined the Trust Association class and the properties it will use. This is done using the WebSphere Application Server's Administration console.

For WebSphere Application Server 5.1, login to the Admin console, and select

Security -> Authentication Mechanisms -> LTPA -> Additional Properties -> Trust Association

1. On the Trust Association panel, there are Additonal Properties - one of which is Interceptors. Select

   Trust Association -> Additional Properties -> Interceptors

2. On the Interceptors panel, enter the name of the class:

   com.ibm.ws.security.web.TAMTrustAssociationInterceptorPlus

   Note: Remove the 'old' Interceptor, *com.ibm.ws.security.web.WebSealTrustAssociationInterceptor*, if it is listed in the panel.

   Figure 61 on page 213 is a snapshot of the panel after entering the correct class:

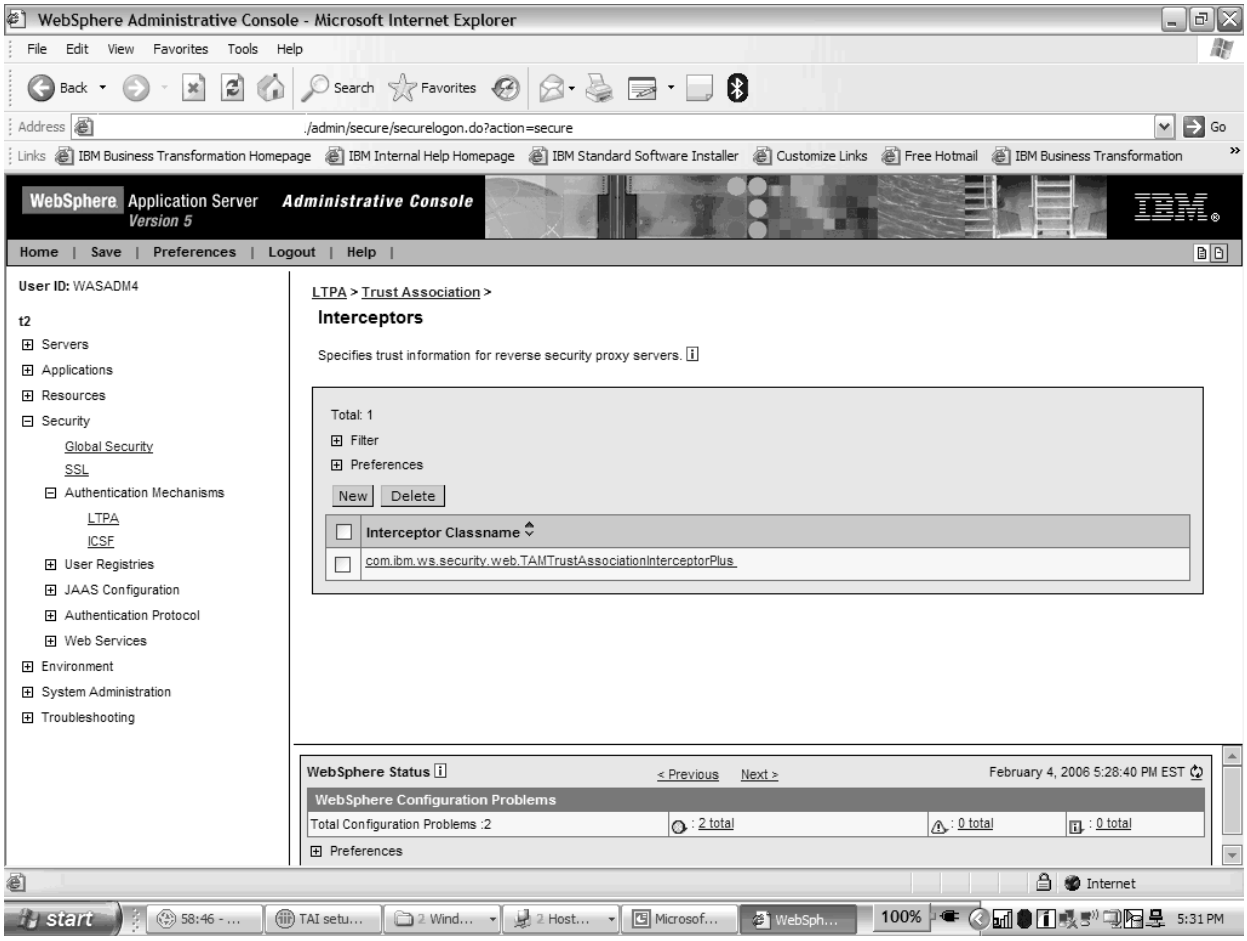


Figure 61. Interceptors panel application

3. Click on the *Classname* to get the configuration panel for the Interceptor class.
4. Click on *Custom Properties*, and add the properties and values as shown in Figure 62 on page 214.

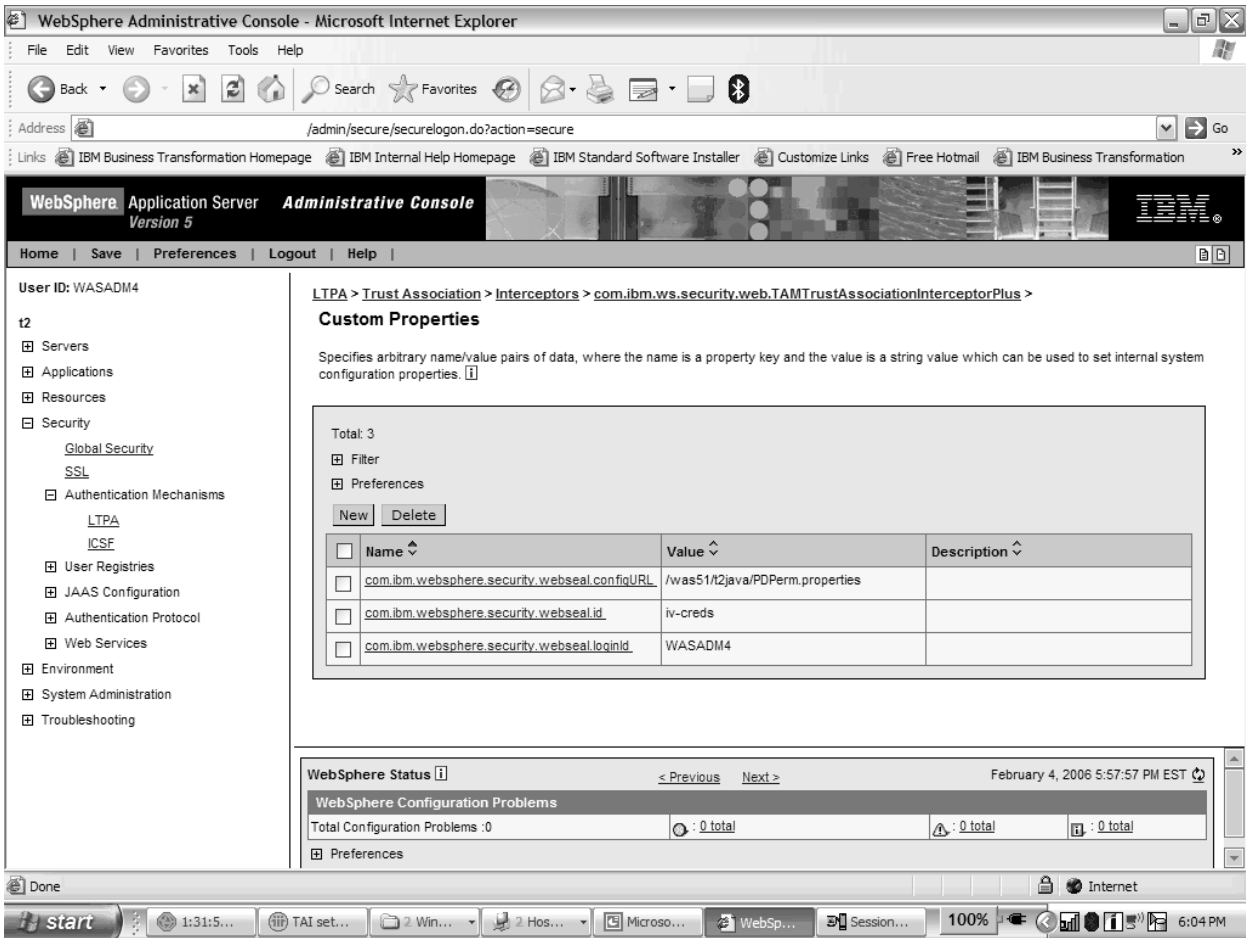


Figure 62. Custom Properties panel

com.ibm.websphere.security.webseal.configURL - **/was51/t2java/PDPerm.properties**

- This property points to the properties file that was created by running the *SvrSslCfg* command. This properties file has the information necessary for the TAI to contact the TAM authorization server.
- This property is mandatory.

com.ibm.websphere.security.webseal.id - **iv-creds**

- This property causes the TAI to ensure that the iv-creds header exists in the request. If it does not, then the TAI will not consider the request as coming from a 'trusted' authentication service.
- This property is mandatory.

com.ibm.websphere.security.webseal.loginId - **WASADM4**

- The TAI takes the userid specified in this property, along with the password from the BA header inserted into the request by WebSEAL, and makes the authentication request to the TAM authorization server.
- This property is mandatory.

There are other Custom Properties that can be set. For a list, please see either the developer works article -

`http://www-128.ibm.com/developerworks/tivoli/library/t-tamtai/`

Or, the WebSphere Application Server InfoCenter -

`http://www.ibm.com/software/webservers/appserv/was/library/`

5. Restart the WebSphere Application Server instance to pick up the changes.

Creating the WebSEAL junction: We created the WebSEAL junction using the *pdadmin* command line:

```
Command:

pdadmin sec_master> s t default-webseald-metlnx30 create -t ssl -h z2.pok
.ibm.com -p 443 -c iv_creds -B -U WASADM4 -W PETWAS -D "CN=z2.pok.ibm.com,
OU=z/OS IT WAS5 J2EE Server 7,O=z/OS IT,L=Poughkeepsie,ST=New York,C=US" /tai


Output:

Created junction at /tai
```

In the *pdadmin* command above:

- *'s t default-webseald-metlnx30'* issues the *pdadmin* 'server task' command to the webseal named 'default-webseald-metlnx30'
- *'create -t ssl'* creates an SSL junction - communication between WebSEAL and the junctioned HTTP Server will be encrypted
- *'-h'* and *'-p'* give the hostname and port of the HTTP server to WebSEAL which will direct requests using the junction.
- *'-c iv_creds'* tells WebSEAL to insert credential information into an HTTP header when making the request to the junctioned server
- *'-B -U WASADM4 -W PETWAS'* tells WebSEAL to insert a basic authentication HTTP header into the request, with the username WASADM4 and password PETWAS
- *'-D "DN of the server certificate" '* tells WebSEAL the expected DN of the server certificate that is received from the junctioned server. This has to match, otherwise the junction create command returns the error:

  ```
  DPWIV1218E   Error in junctioned server DN verification.
  DPWWM1472I   The specified DN for the junctioned server certificate is incorrect.
  ```

- *'/tai'* is the name given to the junction. Requests to WebSEAL with this junction identifier specified in the URL will be directed to the junctioned server.

Note that there are many WebSEAL junction options. Another way the junction could be created is with the *'-b supply'* option, instead of the '-B -U WASADM4 -W PETWAS' options:

```
Command:

pdadmin sec_master> s t default-webseald-metlnx30 create -t ssl -h z2.ibm.fab
rikam123.com -p 9471 -c iv_creds -b supply -D "CN=z2.pok.ibm.com,OU=z/OS
IT WAS5 J2EE Server 7,O=z/OS IT,L=Poughkeepsie,ST=New York,C=US" /tai

Output:

Created junction at /tai
```

In the *pdadmin* command above:

- The **-b supply** option instructs WebSEAL to supply the authenticated Tivoli Access Manager username (client's original identity) with a static, ("dummy") password that is specified in the WebSEAL configuration file. In our case this parameter in the WebSEAL conf file is specifed as:

  basicauth-dummy-passwd = **PETWAS**

The difference in the two different junction commands shown is that in the first case, a valid User ID and Password combination is being sent to the junctioned server in the HTTP BA header. In the second case, the original client User Id from the authenticated user is being sent.

Although, in the second case, the user & password sent in the HTTP BA header is not a valid combination. The TAI only uses the password value from the BA header, and combines it with the User ID specified in the TAI custom property resulting in a successful authentication. So either junction works.

com.ibm.websphere.security.webseal.loginId - **WASADM4**

For more information on WebSEAL junctions, consult the WebSEAL Administration Guide, which can be accessed from the Access Manager for e-business documentation in the IBM Tivoli Information Center located at:

`http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp`

Creating the users in TAM and RACF to access the WebSphere Application Server for z/OS application: The user WASADM4 must be a valid TAM user, since the TAI will attempt to authenticate this ID using the TAM authorization server. We created this user with the TAM *pdadmin* command on Linux on zSeries:

Command/Output:

```
pdadmin sec_master> user create WASADM4 cn=WASADM4,o=ibm,c=us WASADM4 WASADM4 PETWAS
pdadmin sec_master> user modify WASADM4 account-valid yes
```

The client userids must be defined to both TAM and RACF. We created a client userid called PDWEB using the TAM *pdadmin* command on Linux on zSeries:

Command/Output:

```
pdadmin sec_master> user create PDWEB cn=PDWEB,o=ibm,c=us PDWEB PDWEB linux390
pdadmin sec_master> user modify PDWEB account-valid yes
```

Following are the TSO/RACF commands on z/OS. We have a CBS390 resource in the APPL class with group WASCFGGP having READ access. So we connected our client user to the WASCFGGP to allow access to WebSphere Application Server applications:

Commands:

```
adduser pdweb
alu pdweb password(pdweb) noexpired
connect pdweb group(WASCFGGP)
```

Our application - zBank, is protected using the wssecurity1 resource in the RACF ejbrole class. We gave our client ID read access:

Command:

```
permit wssecurity1 class(ejbrole) id(pdweb) acc(read)
```

Output:

```
ICH06011I RACLISTED PROFILES FOR EJBROLE  WILL NOT REFLECT THE UPDATE(S) UNTIL
A SETROPTS REFRESH IS ISSUED
```

To refresh the RACF database with the security permission, we entered the following command:

```
setr raclist(ejbrole) refresh
```

Then we accessed our zBank application in WAS by entering a URL in our browser that points to the webseal system plus the junction identifier 'tai' and the URI for our application /(zBank):

```
https://metlnx30.pdl.pok.ibm.com/tai/zBank
```

We received the TAM login form from WebSEAL. We entered our client userid/password (pdweb/pdweb).

Now we are allowed access to the zBank application, with the credentials of our RACF user.

We created a userid 'NOTAI' the same way as the ID 'PDWEB' was created. The exception was that we did not PERMIT 'NOTAI' to the WSSECURITY1 resource in the EJBROLE class. If we use the 'NOTAI' ID to authenticate to the TAM login form shown above, we get an HTTP Error 403 (Forbidden) response code.

The WebSphere Application Server instance's servant region output shows the messages indicating the RACF check on the resource in the EJBROLE class failed for this user:

```
Output:

  error message: BBOS0105E MSG_BBOSENUS_SEC_REQUESTED_EJBROLES_CHECK_FUNCTION_FUNCTION_FAILED:
SAF Return Code (hex) : 8 The requested FASTAUTHCHECK function failed and could not be performed
for UserID NOTAI using Role Name wssecurity1 and Class Name EJBROLE

  error message: BBOS0103E MSG_BBOSENUS_SEC_EJBROLES_CHECK_FAILED:
The requested EJBROLESAUTHCHECK(RACROUTE) function User NOTAI not permitted to method zBank via Allowed
roles (wssecurity1,.)
```

Results of our testing: As we went through the process of setting up the Trust Interceptor in WebSphere Application Server, we encountered a problem getting the subject extracted from the iv-creds mapped to a RACF id. We were given a temporary fix for this issue, but that fix has not made it into the product yet. We currently have an open PMR and are awaiting a formal APAR.

Where to find more information

During our testing, we used documentation from several sources, listed below. They contain all of the documents that we have cited throughout the course of this chapter.

- IBM WebSphere Application Server for z/OS and OS/390 documentation, available at

  `http://www.ibm.com/software/webservers/appserv/zos_os390/library/`
- Welcome to the WebSphere Application Server, Version 6.0 Information Center, available at publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp
- IBM Techdocs (flashes, white papers, and others), available at www.ibm.com/support/techdocs/
- *Java 2 Platform Enterprise Edition Specification*, available at http://java.sun.com/products/j2ee/
- IBM CICS Transaction Gateway documentation, available at http://www.ibm.com/software/ts/cics/library/

- IBM HTTP Server for OS/390 documentation, available at http://www.ibm.com/software/webservers/httpservers/library/
- IBM WebSphere Studio Workload Simulator documentation, available at www.ibm.com/software/awdtools/studioworkloadsimulator/library/

Specific documentation we used

Documentation to assist you with the usage of your product is available in many places. We have found that the Washington Systems Center documentation is very good and very often this same information is also in the information center. While we offer a set of generic links to documentation, see “Where to find more information” on page 217 for more information, we also wanted to take this opportunity to highlight the specific documentation we used and found especially useful.

For our current WebSphere for z/OS V6 configuration, we found the following documentation was especially good at getting us up and running quickly:

- Setting up WebSphere for eWLM monitoring of DB2 applications found at
  `http://publib.boulder.ibm.com/infocenter/eserver/v1r2/index.jsp`
- Defining JMS and JDBC Resources for Trade6 found at
  `http://www.ibm.com/software/webservers/appserv/was/performance.html`
- Providing authentication, course-grained security, and single sign-on for Web/EJB based applications running on WebSphere Application Server for z/OS found at
  `http://www.ibm.com/developerworks/tivoli/library/t-tamtai/`
- For more information on WebSEAL junctions, consult the WebSEAL Administration Guide, which can be accessed from the Access Manager for e-business documentation in the IBM Tivoli Information Center located at
  `http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp`

Part 3. WebSphere Integration Test (WIT)

The following chapters describe the WebSphere Integration Test (WIT) aspects of our computing environment.

Chapter 16. Introducing our WebSphere Integration Test (WIT) environment

Over the past few months, an additional team, the WebSphere Integration Test (WIT) team, has built an environment within the zSeries Platform Test Sysplex. In this section, we provide a description of the new test team and a level-set view of our current test environment.

Over the past several years, the creators of this team saw a particular need within the IBM zSeries test community. The crux of the argument was that experts from the various software products did not directly communicate on a day to day basis with a common goal in a single environment. How could IBM adequately discover the problems between products without people from the various software products coming together with a common set of goals in an environment that runs 24 by 7 for extended periods of time?

With this in mind, we created a growing team composed of personnel from WebSphere Application Server, DB2, WebSphere MQ, WebSphere XD, application personnel, and test tool support personnel. Together with the existing platform test teams we also cover the base z/OS Operating System and its components, CICS, IMS, and all other topics covered within this document. As the team grows WIT will be able to continue to go down into the base and out among the IBM software stack on z/OS in a customer like environment.

The benefits of the WIT team structure

The benefits of the WIT team structure are:

- It closely emulates a customer, in that:
  - all personnel have a particular role
  - we have test, quality assurance, and production environments
  - we have maintenance scheduling
  - we have a suite of applications
- We have experts discovering how each other think the products work
- We have an already robust existing z environment.

The WIT mission

The WIT mission is to combine applications and products running WebSphere for z/OS solutions emulating a customer environment above and beyond current testing capabilities for WebSphere only or DB2 only test teams.

WIT will be focusing on z/OS, specifically WebSphere Application Server. Along with the application server, WIT will heavily exercise WebSphere MQ, XD, and as we grow move into many of the products that require the application server.

Our purpose is to do both “Integration” and “Test” on page 223.

Integration

Customers require z/OS and WebSphere solutions to be available 24 by 7 with very little disruptive time every year. When customer applications have disruptions,

customers lose money. The WIT Test team will address this by running like the customer. WIT will be running a robust suite of applications 24 by 7 as a customer would. WIT will have to:

- upgrade all the software on scheduled intervals
- add, change, and modify the application suite
- deal with what and how to automate

WIT has a twofold approach to integration:

1. "Integration down into the operating system and its products"
2. "Integration up through the software stack interacting with z/OS and WebSphere Application Server"

Integration down into the operating system and its products

Some of the products required by WebSphere on z/OS are:

- Java SDK
- z/OS Unix Systems Services
- Language Environment (LE)
- z file System (zFS) or Hierarchical File System (HFS)
- System Authorization Facility (SAF)
- Resource Access Control Facility (RACF)
- Resource Recovery Services (RRS)
- Transfer Control Protocol/Internet Protocol (TCP/IP)

These are recognizable products an inexperienced systems programmer would know exist.

Some of the base operating system services exercised by WebSphere solutions are:

- Data Facility Storage Management Subsystem (dfSMS)
- Job Entry Subsystem (JES)
- Workload Management (WLM)
- zSeries Application Assist Processor (zAAP)
- Parallel Sysplex

Every software product and service listed in this section will be exercised heavily by a set of WebSphere solutions running continuously for long periods of time. We will simulate peaks, upgrade code, and continuously add more "customers".

Integration up through the software stack interacting with z/OS and WebSphere Application Server

WebSphere Application Server does not make much sense by itself. Computers exist to manipulate data. Data needs to be created, needs to be stored, needs to be manipulated, and needs to be displayed. Customers have been doing these items for forty years before WebSphere. It is true; the Java 2 Enterprise Edition (J2EE) architecture supported by WebSphere Application Server can handle all of these characteristics of data, but in limited ways.

Customers do not want to create new applications for everything. They want to preserve investment in existing non Java applications. WIT will have CICS and IMS transactions and services in our suite of applications as a customer would to address this realistic need.

Customers keep their data in proven and robust database management systems. WIT will perform specific and inherent testing with DB2 on z/OS to address this realistic need.

Customers want to take advantage of manipulating their data within all types of transactions. To submit a request and have the desired resulting data return at some point in the future is necessary to customer success. Sometimes that point in the future is very close allowing the transaction to remain within WebSphere Application Server, CICS, DB2 and IMS. Sometimes that point in the future is unknown. To satisfy this unknown length of time customers use asynchronous messaging products like WebSphere MQ. WIT will be using WebSphere MQ as well as WebSphere Enterprise Service Bus and WebSphere Message Broker to address this realistic need.

As realistic needs arise WIT will move throughout the IBM software stack to address those needs.

Test

WIT is a group of IBM personnel dedicated to running realistic WebSphere solutions. The environments are described in Chapter 17, “About our z/OS WebSphere Integration Test (WIT) environment,” on page 225; however, notice the term environment is plural. WIT will have an environment that starts in test on a single z/OS Logical Partition (LPAR). The test environment is not burdened with business rules allowing the daily activities for personnel to be unimpeded by approvals. We have the ability to build as many WebSphere Application Server cells as necessary for daily activities on this single LPAR.

When a particular application or configuration has been proven, it is approved and promoted to the quality assurance (QA) environment. QA has multiple WebSphere cells spanning two LPARS in a single sysplex. The first cell is a WebSphere Application Server cell closely resembling what will be in our production environment. QA will have business rules surrounding it so that we can adequately test our applications and configurations for a period of around 2 weeks before we promote them to production. Descriptions of these environments can be found in Chapter 17, “About our z/OS WebSphere Integration Test (WIT) environment,” on page 225.

Chapter 17. About our z/OS WebSphere Integration Test (WIT) environment

This section provides an overview of our z/OS V1R7 WebSphere test environment, including the set of software products and release levels that we run, the Web application configurations that we support, and the workloads that we use to drive them.

Our current software products and release levels

In addition to the elements and features that are included in z/OS V1R7, our WebSphere test environment uses the following software products and release levels:

- WebSphere Application Server for z/OS Version 5.1
- WebSphere Application Server for z/OS Version 6.0.2
- IBM SDK for z/OS, Java 2 Technology Edition V1.4.2
- WebSphere Studio Workload Simulator V1.0
- WebSphere MQ for z/OS V6
- WebSphere Enterprise Service Bus 6.0.1
- WebSphere Message Broker 6.0
- DB2 V8.1 with JDBC
- CICS TS 3.1 – CICS Transaction Gateway (CICS TG) V6.0
- IMS V9 with IMS Connector for Java V9 – IMS Connector for Java V9
- z/OS LDAP V1R7

Note: Future releases of WIT information will list specific PTF levels. At this time we will introduce the concept and general levels only.

Our current WebSphere Application Server for z/OS configurations and workloads

We created the following for our initial WebSphere Application Server for z/OS test and production configurations:

- Created a single cell using WebSphere Application Server for z/OS V5.1
- Created 8 cells using WebSphere Application Server for z/OS V6.0.2
- Altered 2 cells to use LDAP for WebSphere Application Server security backend
- Exploited DB2 UDB JCC Connectors with Sysplex Distributor for higher availability
- Exploited WebSphere Application Server Session Memory Replication.

Our test, quality assurance, and production configurations

Our current environment has focused on WebSphere for z/OS V6. Our current setup contains 9 cells:

- Cells T0 through T6 for our test cells
- Cell Q0 for our quality assurance (QA) cell
- Cell P0 for our production cell

The following sections describe our environment and cells in more detail:

Our test environment

We have six test cells in our WIT configuration, as shown in Figure 63.

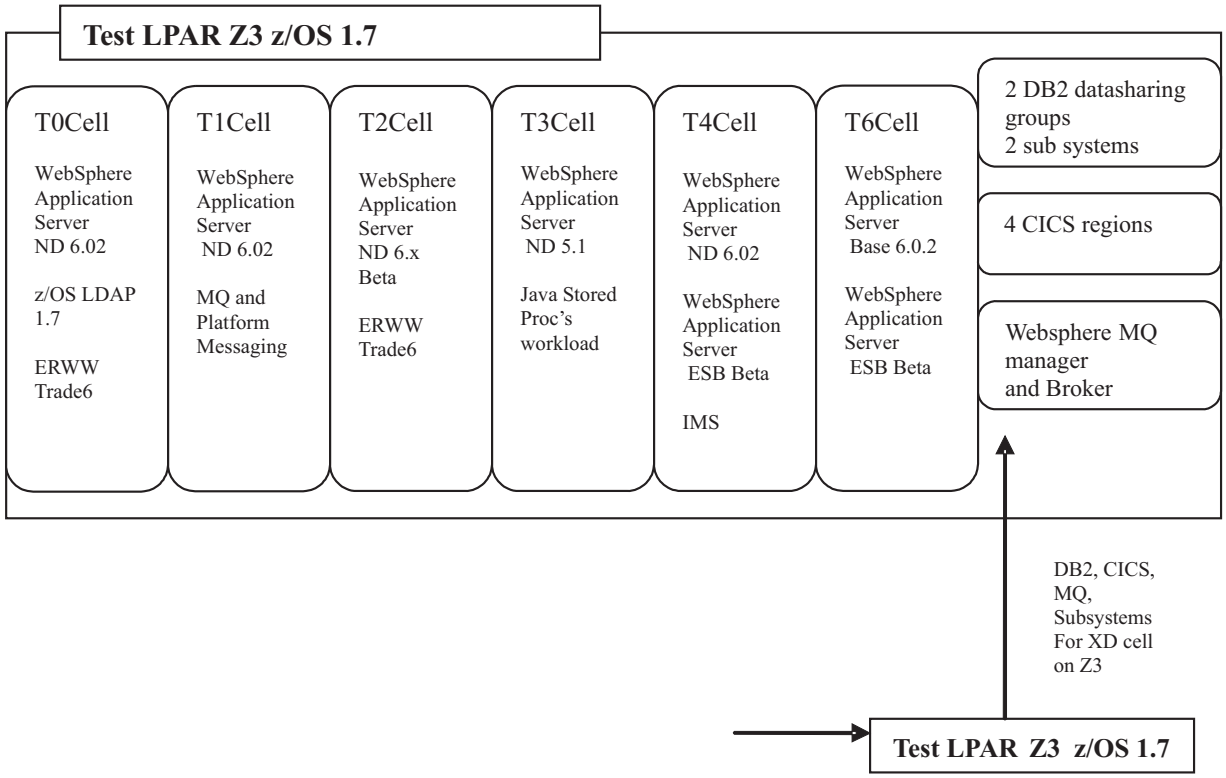


Figure 63. Our WIT test environment

T0 Cell: Our T0 WebSphere Application Server 6.0.2 ND cell resides entirely on one of our test systems (Z3). This single application server cell is used for 2 purposes:

1. To connect and test varying levels of DB2 on systems across the United States
2. To use LDAP as the user registry. Applications used will be described in a later section.

T1 Cell: Our T1 WebSphere Application Server 6.0.2 ND cell resides entirely on one of our test systems (Z3). This 3 application server cell is used for many purposes:

1. To test our naming conventions when having multiples of the same application in a single cell (never in a single server)
2. To test Platform Messaging and WebSphere MQ Messaging using the same application in different servers, one with Platform and one with MQ
3. To test Legacy versus the JCC JDBC provider migration tool
4. We will be migrating this cell to the next level of WebSphere and testing some new function with the Platform Messaging Provider and SIBUS. Applications used will be described in a later section.

T2 Cell: Our T2 WebSphere Application Server 6.x ND cell resides entirely on one of our test systems (Z3). This 2 application server cell is used for 2 purposes thus far:

1. To create a cell using the new version of WebSphere during the Beta time frame.
2. We will be gathering baseline data while running our applications individually on the beta level of WebSphere. Applications used will be described in a later section.

T3 Cell: Our WebSphere Application Server 5.1 ND cell resides entirely on one of our test systems (Z3). This 2 application server cell is used for 2 purposes thus far:

1. We will be migrating this cell to the next level of WebSphere.
2. We will be gathering baseline data from our stored procedure workload using the newest level of WebSphere in Beta. Applications used will be described in a later section.

T4 Cell: Our T4 WebSphere Application Server 6.0.2 ND cell resides entirely on one of our test systems (Z3). This 4 application server cell is used for 2 purposes thus far:

1. We are doing some specific testing with transactions and IMS for development.
2. We will be looking into WebSphere Enterprise Service Bus when it becomes available as Beta. Applications used will be described in a later section.

T5 Cell: Our T5 WebSphere Application Server 6.0.2 ND cell resides on two of our test systems (Z1 and Z2). This 4 application server cell is configured for WebSphere eXtended Deployment:

1. We will be looking into WebSphere eXtended Deployment when it becomes available as a Beta.
2. We will be running batch workloads with this cell.

T6 Cell: Our T6 WebSphere Application Server 6.0.2 Base cell resides entirely on one of our test systems (Z3). This single application server cell is used for 1 purpose thus far:

1. We will be looking into WebSphere Enterprise Service Bus when it becomes available as Beta. Applications used will be described in a later section.

Our quality assurance (QA) environment

Our Q0 WebSphere Application Server 6.0.2 ND cell resides on two of our test systems (Z1 and Z2) as shown in Figure 64 on page 228.

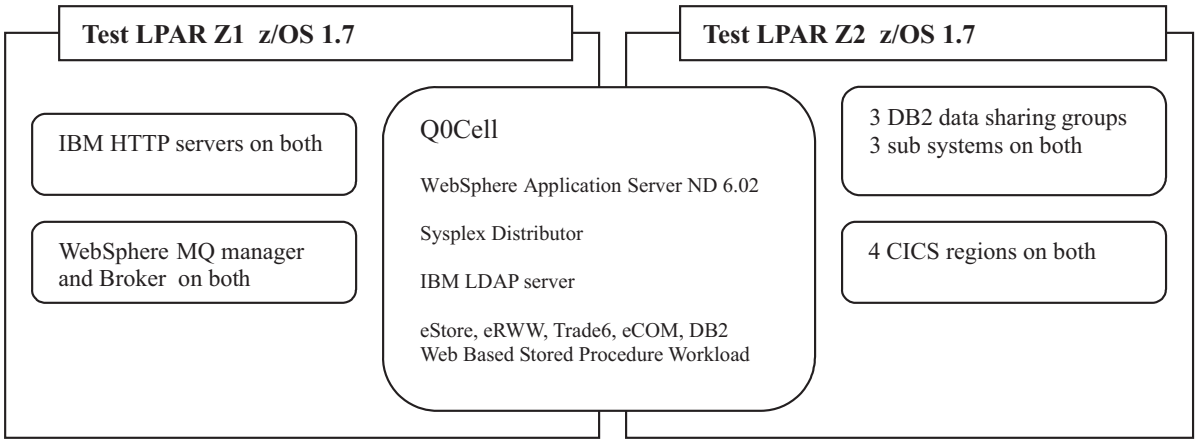


Figure 64. Our WIT quality assurance (QA) environment

This 5 application server cell is configured to be our second stop before promoting anything to production. The configuration of quality assurance is more robust than any of the test cells and consists of the following:

- Multiple LPARs
- IBM HTTP Server front end
- Sysplex and Dynamic VIPA setup for higher availability
- Horizontal and vertical clustering of applications for higher availability
- LDAP custom registry with future work using Tivoli products for security
- WIT owned DB2 and WebSphere MQ
- Shared CICS and IMS among anyone using the z systems we are using
- First point to exercise all of the applications
- First point to exercise all administrative scripting
- First point to exercise all automation.
- First point to exercise continuous availability during maintenance
- First point to exercise continuous availability during product migrations
- First point to exercise procedures for simulating clients in our continuous environment.

Applications used will be described in a later section.

Our production environment

Our production environment is shown in Figure 65 on page 229.

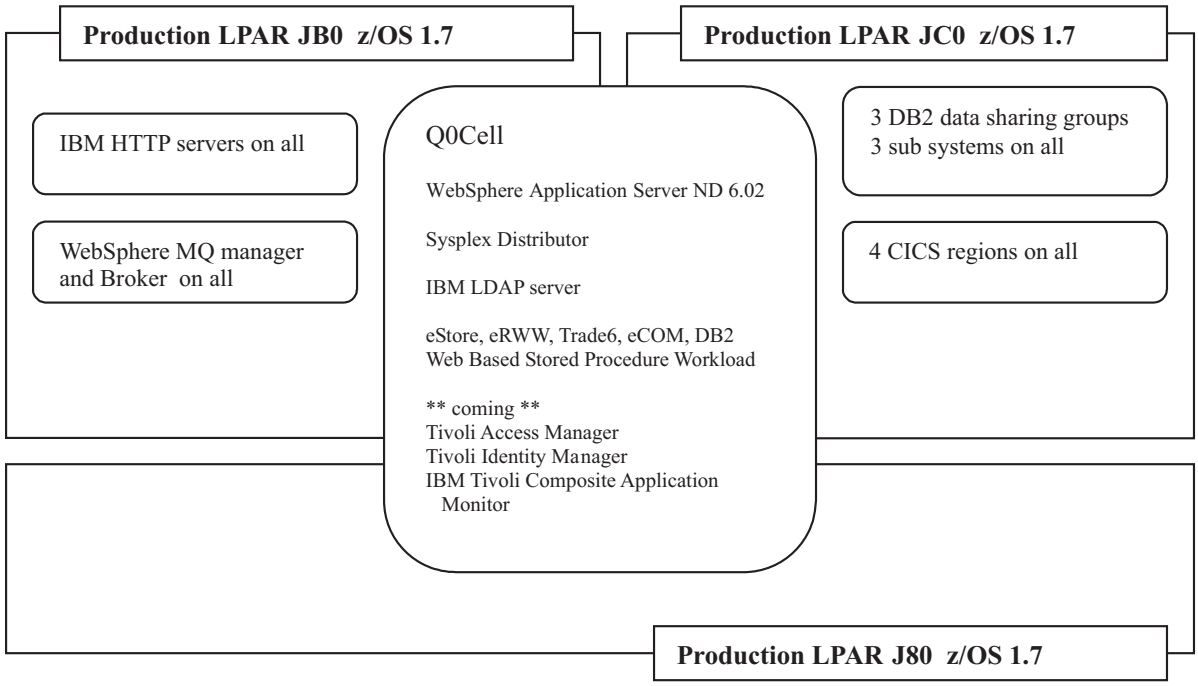


Figure 65. Our WIT production environment

WIT will be using Rational Performance Tester and WebSphere Workload Simulator to drive work into all of these systems.

Our production environment consists of the following applications and workloads:

- "Enhanced Relational Warehouse Workload (ERWW)"
- "Trade6" on page 230
- "eStore" on page 230
- "eCOM and DB2 Web based stored procedure workload" on page 230

Enhanced Relational Warehouse Workload (ERWW):

Enhanced Relational Warehouse Workload (ERWW) Application
Workload based on the TPC-C specification plus additional transactions
Order Inventory Application
Database Medium Tables

Warehouse	25 rows
Item	100,000 rows
NewOrder	225,000 rows
Order	750,000 rows
History/History1	750,000 rows
Customer	750,000 rows
Stock	2,500,000 rows
Orderline	7,501,324 rows

NewOrder Transaction
Consistency Checking
Exception Handling & Tracing
Automation

EJB Workload
J2C CICS Workload

J2C IMS Workload
Messaging Scenarios
Web Services SOAP over HTTP
Web Services SOAP over JMS

** taken from a presentation produced by the developers of eRWW **

Trade6: Trade is the WebSphere end-to-end benchmark and performance sample application. The new Trade benchmark has been re-designed and developed to cover WebSphere's significantly expanding programming model. This provides a real world workload driving WebSphere's implementation of J2EE 1.4 and Web Services including key WebSphere performance components and features.

Trade's new design spans J2EE 1.4 including the new EJB 2.1 component architecture, Message Driven beans, transactions (1-phase, 2-phase commit) and Web Services (SOAP, WSDL). Trade also highlights key WebSphere performance components such as DynaCache, WebSphere Edge Server and Web Services.

eStore: eStore is an altered implementation of the eRWW application. In eStore, the database schema was redesigned to take advantage of every data type and construct introduced in DB2 V8. Along with all of the data types and constructs the use of Java Stored Procedures are logically placed throughout the application. This will be WIT's primary application to begin testing while we continue to grow our application suite.

eCOM and DB2 Web based stored procedure workload: The JSPUDF workload consists of scenarios based on Servlets, Java Stored Procedures, Java User Defined Functions, DB2 Views and Triggers for DB2 stress testing.

N81CS301 exercises accessing long names for various DB2 objects. It exercises the following DB2 for z/OS V8 line items:

- LI537: Support of Long Names for DB2 objects
- LI486: Perform database operations on a tablespace with greater than 254 partitions.

There is one Servlet that drives the execution of the scenario: *JSPUDF/N81CS301*

The workload runs against EBCDIC, ASCII and UNICODE databases. The databases have been replicated (05 replicas) to allow higher numbers of transactions and connections to the DB2 system – JSPB1*xNN*, where *x* indicates de CCSID value (A, E, and U) and *NN* indicates the replica identification (01 thru 05). Each replica has a respective schema represented as JSPB00*NN*, where *NN* indicates the replica identification.

Examples of WIT's early success for the customer

WIT will be using Rational Performance Tester and WebSphere Workload Simulator to drive work into all of these systems.

The following are some examples of WIT's early success for the customer:

- WIT environment expansion
- 50 defects and APARS have been identified to date
- A test environment dedicated to exercising more products together for a longer period of time
- Continued WIT Environment expansion

- Papers and InfoCenter Enhancements, such as those found in Table 16.

Table 16. Examples of WIT's early success for the customer

Tips and enhancements:	Found at:
Tips on Installing and First Steps Configuration of WebSphere for z/OS V6	http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100725
Diagnosing Performance Problems with WebSphere Application Server on z/OS	http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100678
WebSphere Application Server for z/OS Tuning Tips V5 and V6	http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/TD103036

Future WIT on zSeries projects

As we continue to grow the environment and add more of IBM's software stack, the WIT team will exercise and modify our suite of applications to stress and take advantage of both existing and the newest technologies.

Future areas of interest

Following are some of WIT's future areas of interest:

- WebSphere z/OS exploitation of zSeries Qualities of Service
- Security
  - Authentication
  - Authorization
  - Identity management
  - Single sign-on
  - Auditing
  - Compliance
- Migration
  - Migration of z/OS
  - Migration of WebSphere
  - Migration of the applications
  - Migration of the sub-systems
- WebSphere scalability and tuning
- WebSphere monitoring and diagnostic tools (ITCAM, Omegamon, WSAM, RMF, SMF, Dump Diagnostic Tool for Java)
- Integrating products that run on top of WebSphere
- Services Oriented Architecture (SOA) on z/OS

Future WIT overview documents will contain information for IT personnel. The information will detail how and why we chose to construct our environment based on the role of the person responsible for the task.

Part 4. Linux virtual servers

The following chapters describe the Linux virtual servers aspects of our computing environment.

Chapter 18. About our environment

In this release of the test report, we've made significant enhancements to our environment. With the implementation of High Availability Architectures, we took the opportunity to streamline our network. Figure 66 on page 236 is our current network diagram. The shaded area contains our Linux guests which are running on a single z/VM 5.2 image on a z990 processor. We've added a commercial firewall layer, combined the routers on the same LAN segment and consolidated images over to our backoffice LAN. For the previous Linux networking diagram, please refer to the December 2005 zSeries Platform Test Report, section 21.2, Linux on zSeries network configuration.

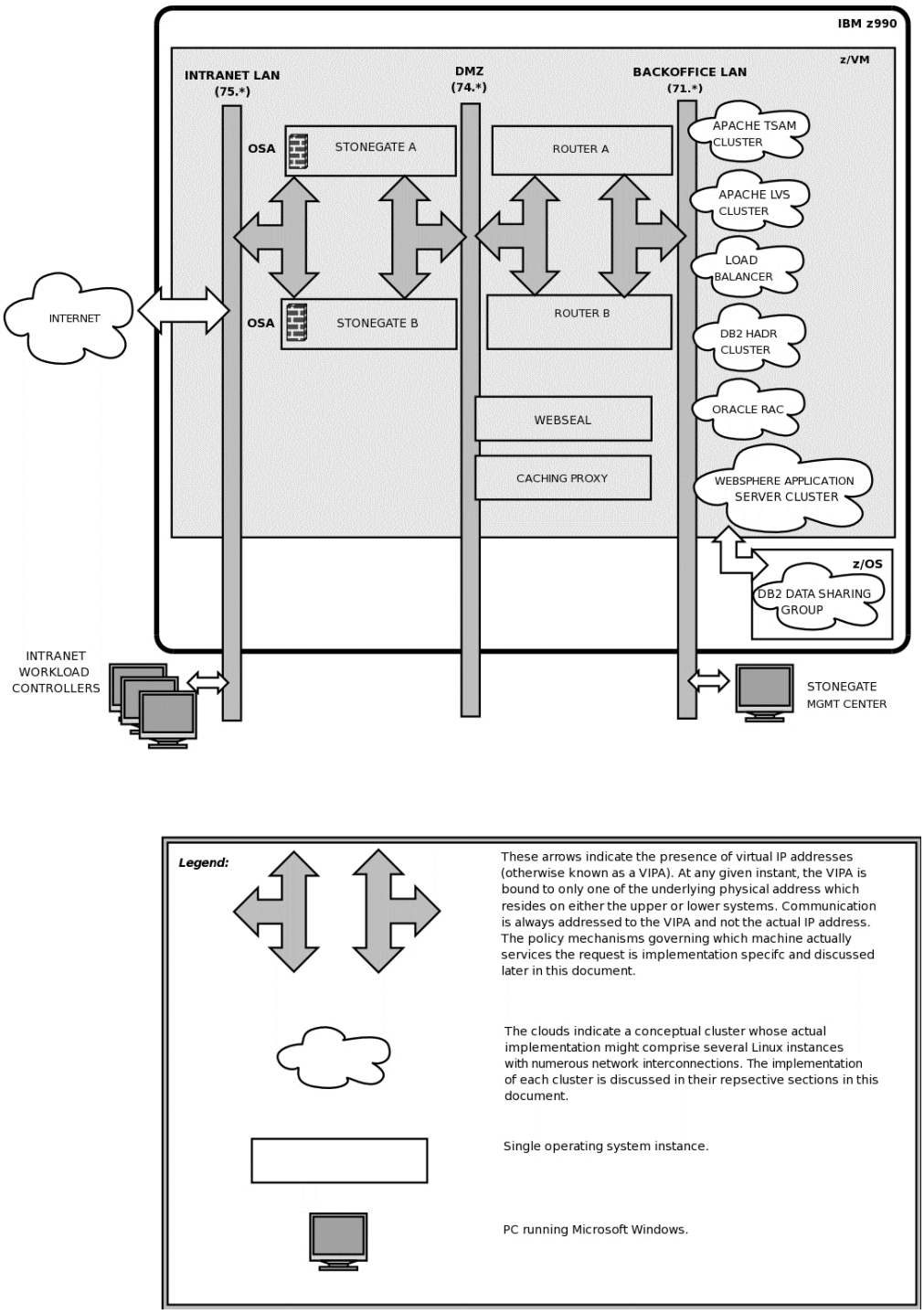


Figure 66. Our Linux on zSeries network configuration.

Chapter 19. Implementing High Availability Architectures

The zLVS PET team tested the set of reference architectures produced by the zSeries New Technology Center and the eServer High Availability Center of Competence. The name of the resulting white paper is: *High Availability Architectures For Linux on IBM System z*. The architectures provide highly availability solutions for applications running in Linux on zSeries and can be referenced here:

`http://www.ibm.com/servers/eserver/zseries/library/whitepapers/pdf/HA_Architectures_for_Linux_on_System_z.pdf`

In the paper, information on the technologies that provide high availability is presented along with descriptions of how they work together to provide end-to-end highly available solutions. Here in the test report, we focus on the implementation details of our test efforts.

Implementing WebSphere Application Server HAManager

The HAManager is a new feature of WebSphere Application Server V6. We referenced the *Redbook SG24-6392: WebSphere Application Server V6 Scalability and Performance Handbook*, section 9.7 “Transaction Manager High Availability”, for details on how to setup the HAManager. We spend most of this section covering the setup of NFS version 4 to be used by the HAManager for recovering in-flight Two Phase Commit (2PC) transactions. Then we’ll cover a couple of areas that aren’t talked about in the Redbook, for configuring WebSphere Application Server to enable the HAManager.

In the event of a server failure, the HAManager will start a recovery process in other members of the WebSphere Application Server cluster. The in-flight transactions are committed and locks released. In order to do this the transaction logs written by each application server must be on network-attached storage or a Storage Area Network (SAN) so that they are readable to the remaining cluster members. There are a few choices for shared storage as indicated by the aforementioned Redbook. We used NFS version 4. Since the book doesn’t talk about this setup we will discuss that here.

Configuring NFSv4 for Use With WebSphere Application Server V6 High Availability Manager

As described in the *WebSphere Application Server V6 Scalability and Performance Handbook*, there are several network filesystems that have the required functionality to allow WebSphere Application Server 6 to rapidly recover transactions when a clustered node goes down, one of which is NFS version 4. The following matrix shows which System z enterprise Linux distributions have NFS4 client and server support in the stock kernel:

Table 17.

Distribution	NFS4 Support
SLES 9 Base	No support
SLES 9 SP1, SP2, SP3	Client-only support (Note 1)
RHEL 4 Base, U1, U2, U3	Full client and server support

Notes:

1. Note that client-support is available in the kernel with SLES 9 service packs but the default mount program doesn't recognize "nfs4" as a valid filesystem type and will refuse to mount an NFS4 share. The solution to this is to rebuild the mount program from source after applying the latest patches from the NFS4 developers.

For our testing we set up an NFS4 server on a 64-bit Linux guest running RHEL 4, and had four 64-bit SLES 9 guests running WebSphere Application Server V6 as the NFS clients.

Configuring RHEL 4 NFS4 Servers

Setting up an NFS server is very easy with RHEL 4. Simply install the latest NFS utilities RPM, edit the exports list, and then start the NFS services.

The first step is to install the latest nfs-utils package (nfs-utils-1.0.6-65.EL4 at the time of this writing) manually or using Red Hat's up2date utility:

```
up2date nfs-utils
```

If you want to do this manually, simply download the nfs-utils RPM and install it using rpm:

```
rpm -Uvh  nfs-utils-1.0.6-65.EL4.s390x.rpm
```

Note: If you have configured a software firewall (iptables) on your guest or performed a default install of RHEL 4, you will need to verify that NFS/RPC traffic is allowed through the firewall. With the default RHEL 4 firewall, it is not. We didn't need local firewall running in our environment because we have other firewall protection. So for our testing, we disabled the firewall as follows:

```
iptables -F
etc/init.d/iptables save
```

Next we added the following services to the default *runlevel: nfs, nfslock, portmap*, and *rpcidmapd*. This can be done as follows:

```
chkconfig –levels 356 nfs on
hkconfig –add nfslock
hkconfig –add portmap
hkconfig –add rpcidmapd
```

Note: On running portmap; even though official NFS4 specifications don't require portmap (for security reasons – since portmap opens a bunch of ports), portmap was still necessary at the point of our testing on RHEL 4. Otherwise the nfs init script would hang or fail to start everything properly. That is the case because of the NFS2/NFS3 support in the kernel on RHEL 4. Additionally, if portmap wasn't running, showmount and rpcinfo would fail because they use RPC services so we couldn't really verify that NFS was working short of trying mount (which didn't give us very much information if it wasn't working).

Now start the four services (some may automatically be started by other services):

```
service portmap start
service nfs start
service nfslock start
service rpcidmapd start
```

The primary reason NFS4 is ideal for sharing WebSphere Application Server transaction logs is that client locks have an associated lease time which allows them to be recovered easily if a client holding any locks goes down or becomes inaccessible. The *WebSphere Application Server V6 Scalability and Performance Handbook (SG246392)* recommends lowering the default lease time though to minimize any service interruptions. This can be done by echoing the desired lease duration to a special pseudo-device. The following command will set the lease time to 10 seconds, as the handbook recommends:

```
echo 10 > /proc/fs/nfsd/nfsv4leasetime
```

Note that this value will be reset if the guest is rebooted so you'll want to add the command to an init script so the setting is applied at startup. For our system, we appended the following line to */etc/rc.local*:

```
[ -e /proc/fs/nfsd/nfsv4leasetime ] && echo 10 > \
/proc/fs/nfsd/nfsv4leasetime
```

You now need to create an export. For our testing we added a separate DASD pack to store transaction logs and mounted it at /nfslogs. Since our environment contains both Red Hat and SuSE systems, we created two different directories to store logs from the respective groups of systems:

```
mkdir -p /nfslogs/export/rhel /nfslogs/export/sles
```

Make sure the export is readable and optionally, writable by the nfsnobody user:

```
chown -R nfsnobody:nfsnobody /nfslogs/export
```

Now edit the NFS exports file, /etc/exports, and add the following:

```
 /nfslogs/export *(rw,fsid=0,insecure,no_subtree_check,sync)
```

The wildcard character will allow any other system to access the export; if you want to restrict this see the NFS documentation for instructions on how to do so.

Note: Changes in the way NFS works with version 4 means that NFS4 clients no longer see distinct exports on the server as separate entities, instead they see them as a single filesystem. As a result we only created a single export to be used by all of the systems, although when configuring the WebSphere Application Server cluster we pointed them to the proper subdirectory based on the distribution. If you want to export multiple NFS4 shares from a server, there is a workaround described by the developers in the "NFSv4 exports on linux" section on this page:

```
http://www.citi.umich.edu/projects/nfsv4/linux/using-nfsv4.html
```

Export the filesystem:

```
exportfs -vr
```

To verify the export occurred, use the showmount command. You should see something resembling this:

```
[root@litrwas4 ~]# showmount -e
Export list for litrwas4.ltic.pok.ibm.com:
/nfslogs/export *
```

Configuring SLES 9 NFS4 Clients

Install the latest nfs-utils RPM (nfs-utils-1.0.6-103.17) manually or using yast2:

```
yast2 --install nfs-utils
```

If you want to do this manually, simply download the nfs-utils RPM and install it using rpm:

```
rpm -Uvh  nfs-utils-1.0.6-103.17.s390x.rpm
```

Now we verified that the RPC service on the NFS server can be contacted using the rpcinfo command. You should see results for portmapper, status, rquotad, nfs, nlockmgr, and mountd, with varying protocols (TCP, UDP) and versions (1 – 4, typically):

```
litwas01:~ # rpcinfo -p litrwas4
   program vers proto   port
    100000    2   tcp    111  portmapper
    100024    1   tcp  32768  status
    100011    2   tcp    796  rquotad
    100003    4   tcp   2049  nfs
    100021    4   tcp  32769  nlockmgr
    100005    3   tcp    837  mountd
```

Note: The output above has been trimmed to only show key examples; it is likely that you'll receive 20 – 25 lines of output from the rpcinfo command if multiple versions of NFS are supported by the remote server.

Next we downloaded the util-linux package from CITI and applied their combined patch. These were obtained from:

```
http://www.citi.umich.edu/projects/nfsv4/linux/
```

```
litwas01:~ # wget http://www.citi.umich.edu/projects/nfsv4/linux/util-linux-tarballs/util-linux-2.12.tar.gz
litwas01:~ # wget http://www.citi.umich.edu/projects/nfsv4/linux/util-linux-patches/2.12-3/util-linux-2.12-CITI_NFS4_ALL.dif
litwas01:~ # tar -xzvf util-linux-2.12.tar.gz
litwas01:~ # cd util-linux-2.12
litwas01:~ # patch -p1 < ../util-linux-2.12-CITI_NFS4_ALL-3.dif
```

We opened the MCONFIG file located in the util-linux source directory in a text editor and commented out the HAVE_SLANG line, then added the following line:

```
#HAVE_SLANG=yes
HAVE_NCURSES=yes
```

We installed *ncurses, ncurses-devel, gcc, autoconf,* and *automake*:

```
yast2 --install ncurses
yast2 --install ncurses-devel
yast2 --install gcc
yast2 --install autoconf
yast2 --install automake
```

Now we built the package:

```
./configure --prefix=/usr
make
```

We didn't run `make install`, instead we just copied the newly-compiled version of mount to /bin (overwriting the default version of mount):

```
cp mount/mount /bin/
```

CITI provides additional information on building and installing the user-space tools here:

```
http://www.citi.umich.edu/projects/nfsv4/linux/user-build.html
```

Note that many of the steps and dependencies mentioned in the link aren't necessary if you're only building util-linux from source.

Now we were able to mount the remote NFS4 export:

```
litwas01:~ # mkdir –p /nfslogs/export
litwas01:~ # mount -v -t nfs4 -o hard litrwas4:/ /nfslogs/export
```

Later, once the NFS configuration had been tested, it was a good idea to add the NFS share to /etc/fstab so that it was automatically mounted at boot-time. We appended the following line to */etc/fstab* to accomplish this:

```
litrwas4:/ /nfslogs/export nfs4 hard 0 0
```

The Redbook recommends that use the hard option in the NFS mount command to avoid data corruption.

We installed the NFSv4 client on all of our WebSphere Application Server member nodes. There is no need to install it on our WebSphere Application Server Network Deployment node.

Testing NFSv4

The easiest way to prove that NFS4 is properly configured is to create a small test case where a client obtains a file lock and then pull the “network cable” by forcing the VM guest or changing the default firewall policies to drop packets so no network traffic goes through.

For our testing we created two small C programs, each of which is run on a different VM guest. The first program opens a file for writing and obtains an advisory lock for it, then holds the lock for some “long” period of time (in our case, 10 minutes). The second program attempts to acquire a lock on the file and will wait until the lock becomes available.

```
litwas01:~ # ./lockhold
16:07:46 Opening nfslocktest.txt.
16:07:46 About to request write lock.
16:07:46 Got write lock, writing to file.
16:07:46 Done writing, holding lock for another 10 minutes.

litwas02:~ # ./lockwait
16:07:49 Opening nfslocktest.txt.
16:07:49 About to request write lock.
16:07:49 Write lock already held, will wait and try again.
```

While the first program is holding the lock and the second program is waiting, we changed the iptables policies on the first guest (the lock-holder) to drop all packets, simulating a network outage:

```
litwas01:~ # iptables -P INPUT DROP
litwas01:~ # iptables -P OUTPUT DROP
litwas01:~ # iptables -P FORWARD DROP
litwas01# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination

Chain FORWARD (policy DROP)
target     prot opt source               destination

Chain OUTPUT (policy DROP)
target     prot opt source               destination
```

Meanwhile, back on litwas02:

```
16:08:16 Finally got write lock.
16:08:16 Attempting to release the write lock.
16:08:16 Write lock released, closing file.
16:08:16 Exiting.
```

As you can see from the two programs' output, the first client's lock lease eventually expires and the second client is then able to acquire the lock and perform the operation it wanted on the file (checking the file's contents confirms this). Had we been using NFS3, the second client would have never been able to get the lock and write to the file after the first client dropped off the network. You can try this yourself by repeating these test steps and mounting the exported file system as NFS3 with this command (note that the syntax is slightly different):

```
litwas01:~ # mount -v -t nfs -o hard litrwas4:/ /nfslogs/export
```

Configuring WebSphere Application Server HAManager

Configuring WebSphere Application Server HAManager is very straightforward once you have setup NFSv4. The configuration and validation procedures are well documented in the *Redbook SG24-6392: WebSphere Application Server V6 Scalability and Performance Handbook*, section 9.7 "Transaction Manager High Availability". We would just like to point out a couple of areas that might appear vague.

By default the transaction logs for each WebSphere Application Server member are located in the following directory: /$WAS_HOME/profiles/<profilename>/tranlog/<cellname>/<nodename>/<servername>/transaction/. We used a different directory that mounted the remote NFS export which was sharable by all our WebSphere Application Server members.

Our WebSphere Application Server nodes were: litwas01, litwas02, and litwas03. We created the directory /nfslogs/export/sles on all three systems. We mounted the shared file system on each system with the following command:

```
# mount -v -t nfs4 -o hard litrwas4:/ /nfslogs/export/sles
```

After mounting, we created subdirectories that corresponded to the WebSphere Application Server nodes:

```
# mkdir /nfslogs/export/sles/litwas01
# mkdir /nfslogs/export/sles/litwas02
# mkdir /nfslogs/export/sles/litwas03
```

We only needed to create these directories from one of the WebSphere Application Server nodes, since the file system was shared we noticed the added directories from the other WebSphere Application Server nodes as well. Each directory would hold the transaction logs for its respective WebSphere Application Server. In the WebSphere Application Server Network Deployment manager admin console, we specified the matching transaction log directory for each cluster member as indicated in the Redbook.

In each of the WebSphere Application Server node's SystemOut.log, you should see messages similar to the following when you start the cluster:

```
[3/21/06 17:30:29:310 EST] 00000034 LogFileHandle I   CWRLS0006I: Creating
new recovery log file /nfslogs/export/sles/litwas02/tranlog/log2.
[3/21/06 17:30:54:340 EST] 00000034 LogHandle     I   CWRLS0007I: No existing
recovery log files found in /nfslogs/export/sles/litwas02/partnerlog. Cold
starting the recovery log.
[3/21/06 17:30:54:353 EST] 00000034 LogFileHandle I   CWRLS0006I: Creating
new recovery log file /nfslogs/export/sles/litwas02/partnerlog/log1.
[3/21/06 17:30:54:550 EST] 00000034 LogFileHandle I   CWRLS0006I: Creating
new recovery log file /nfslogs/export/sles/litwas02/partnerlog/log2.
```

We didn't copy existing transaction logs to the shared directory so WebSphere Application Server created new recovery log files.

Testing WebSphere Application Server HAManager

After the cluster is started, we killed one of the WebSphere Application Server nodes and noticed the following messages in another WebSphere Application Server node's SystemOut.log:

```
[3/21/06 18:05:40:388 EST] 000001aa RecoveryDirec I   CWRLS0011I: Performing recovery processing for a peer WebSphere server
(litwas04Network\litwas01\TradeServer1).
[3/21/06 18:05:40:404 EST] 000001aa RecoveryDirec I   CWRLS0013I: All persistent services have been directed to
perform recovery processing for a peer WebSphere server (litwas04Network\litwas01\TradeServer1).
[3/21/06 18:05:40:474 EST] 000001aa RecoveryDirec I   CWRLS0013I: All persistent services have been directed to
perform recovery processing for a peer WebSphere server (litwas04Network\litwas01\TradeServer1).
[3/21/06 18:05:40:670 EST] 000001ab RecoveryManag A   WTRN0028I: Transaction service recovering 0 transactions.
```

It was able to detect that litwas01 went down, and started the recovery process. At the time this happened, there were 0 transactions in flight so it didn't have to recover any.

Implementing Highly Available WebSphere Application Server Edge Component Load Balancer

Load Balancer creates edge-of-network systems that direct network traffic flow, reducing congestion and balancing the load on various other services and systems. Load Balancer provides site selection, workload management, session affinity, and transparent failover.

The Dispatcher component of the Load Balancer in our workflow balances network traffic between two Web servers, creating highly available Web access so that if one Web server fails, traffic is uninterrupted because it is now routed to the other Web server. However, what if the Load Balancer itself fails? For the HA test, we wanted to eliminate all software single points of failure. Luckily, WebSphere Application Server Edge Component Load Balancer v5.1 also provides failover support.

There are two ways one can configure failover support in the Load Balancer:

1. **Hot standby:** The backup Dispatcher does not perform load balancing as long as the primary one is operational. The primary and backup Dispatcher track each other's status by periodically exchanging messages called heartbeats. If the backup Dispatcher detects that the primary has failed, it automatically takes over the responsibility for load balancing by intercepting requests directed to the primary's cluster host name and IP address.
2. **Mutual high availability:** In this case, each actively performs load balancing for a separate cluster of content hosts, simultaneously acting as the backup for its colleague.

To further enhance Web site availability, we chose the hot standby method and configured another Load Balancer to act as a backup for the primary Load Balancer. If one Load Balancer fails or becomes inaccessible due to a network failure, end users can still reach the content hosts.

Configuring High Availability on Load Balancer

We installed another Load Balancer on a different Linux system. This Load Balancer was going to be the standby Load Balancer. Now we had two Load Balancers, one on system litlb01, and another on litlb02, and they had the same system level (2.6.5-7.151-s390) and middleware level (v5.1.1.41).

We followed the WebSphere Application Server Network Deployment Edge Component Info Center to configure high availability on our Load Balancer:

http://www-306.ibm.com/software/webservers/appserv/doc/v51/ec/infocenter/edge/LBguide.htm#HDRCONHIAVAIL

We followed this technote to address the Linux ARP incompatibility with Load Balancer's MAC forwarding method:

http://www.ibm.com/support/docview.wss?rs=180&context=SSEQTP&q1=sysctl&uid=swg21177105&loc=en_US&cs=utf-8&lang=en

As a result we ran the following command on litlb01 and litlb02 as well as the Web servers:

```
# sysctl -w net.ipv4.conf.all.arp_ignore=3 \ net.ipv4.conf.all.arp_announce=2
```

Additionally, we added the above sysctl parameters to /etc/sysctl.conf on both Load Balancers and the Web servers, and ran *chkconfig boot.sysctl* to have them start at boot time.

In order to avoid a long downtime, we kept the production Load Balancer running and setup scripts on both litlb01 and litlb02 to do the configuration. Then we shut down the primary Load Balancer and ran the HA.config scripts on both machines.

Our scripts on litlb01 (primary Dispatcher)

HA.config: This script first starts the dsserver, then the executor, and then sets up the cluster. Finally it sets up high availability.

```
# start the server and executor and set log levels
dsserver start
dscontrol executor start
dscontrol set loglevel 1

# add cluster address
dscontrol cluster add 192.168.71.98
dscontrol cluster set 192.168.71.98 proportions 49 50 1 0
dscontrol executor configure 192.168.71.98 eth0 255.255.255.0

# setup port number information on cluster
dscontrol port add 192.168.71.98:80 reset no
dscontrol port add 192.168.71.98:443 reset no

# add servers to the cluster
dscontrol server add 192.168.71.98:80:192.168.71.119 address 192.168.71.119
dscontrol server add 192.168.71.98:443:192.168.71.119 address 192.168.71.119
dscontrol server add 192.168.71.98:80:192.168.71.127 address 192.168.71.127
dscontrol server add 192.168.71.98:443:192.168.71.127 address 192.168.71.127

# start the manager and advisor
dscontrol manager start manager.log 10004
dscontrol advisor start Http 80 Http_80.log
dscontrol advisor start Http 443 Http_443.log

# setup heartbeat for HA
# first IP is litlb01, second IP is litlb02
dscontrol highavailability heartbeat add 192.168.71.99 192.168.71.135
# number of seconds that the executor uses to timeout HA heartbeats
dscontrol executor set hatimeout 3
# see if the dispatcher can reach the backend WAS servers
dscontrol highavailability reach add 192.168.71.101 192.168.71.102 192.168.71.105
# set this dispatcher as the primary dispatcher
dscontrol highavailability backup add primary auto 9123
# print the status to screen
dscontrol highavailability status
```

goActive: This script is required for HA. It is run when the Dispatcher goes into active state. Added goActive script in the /opt/ibm/edge/lb/servers/bin/ directory as follows:

```
CLUSTER=192.168.71.98
INTERFACE=eth0:1
NETMASK=255.255.255.0
echo "Adding device alias"
ifconfig $INTERFACE $CLUSTER netmask $NETMASK up
```

goStandby: This script is required for HA. It is run when the Dispatcher switches from active state to standby state. Added goStandby script in the /opt/ibm/edge/lb/servers/bin/ directory as follows:

```
CLUSTER=192.168.71.98
INTERFACE=eth0:1
NETMASK=0xfffff800
echo "Deleting the device alias(es)"
ifconfig $INTERFACE down
```

goInOp: This script is optional. It executes when a Dispatcher executor is stopped and before it is started for the first time. Added goInOp script in the /opt/ibm/edge/lb/servers/bin/ directory as follows:

```
CLUSTER=192.168.71.98
INTERFACE=eth0:1
echo "Removing device alias(es)"
ifconfig $INTERFACE down
```

highavailChange: This script is also optional. It is to log the change in HA state. Added highavailChange script in the /opt/ibm/edge/lb/servers/bin/ directory as follows:

```
DATE=`date`
OUTPUT="$DATE LB just ran $1."
echo $OUTPUT >> /opt/ibm/edge/lb/servers/logs/lb.log
#echo $OUTPUT | mail -s "highavailChange" root@localhost
```

Our scripts on litlb02 (standby Dispatcher)

HA.config: Notice that everything up to the HA configuration is the same as the primary Dispatcher on litlb01:

```
# start the server and executor and set log levels
dsserver start
dscontrol executor start
dscontrol set loglevel 1

# add cluster address
dscontrol cluster add 192.168.71.98
dscontrol cluster set 192.168.71.98 proportions 49 50 1 0
dscontrol executor configure 192.168.71.98 eth0 255.255.255.0

# setup port number information on cluster
dscontrol port add 192.168.71.98:80 reset no
dscontrol port add 192.168.71.98:443 reset no

# add servers to the cluster
dscontrol server add 192.168.71.98:80:192.168.71.119 address 192.168.71.119
dscontrol server add 192.168.71.98:443:192.168.71.119 address 192.168.71.119
dscontrol server add 192.168.71.98:80:192.168.71.127 address 192.168.71.127
dscontrol server add 192.168.71.98:443:192.168.71.127 address 192.168.71.127

# start the manager and advisor
dscontrol manager start manager.log 10004
dscontrol advisor start Http 80 Http_80.log
dscontrol advisor start Http 443 Http_443.log

# setup heartbeat for HA
# first IP is litlb02, second IP is litlb01
dscontrol highavailability heartbeat add 192.168.71.135 192.168.71.99
```

```
# number of seconds that the executor uses to timeout HA heartbeats
dscontrol executor set hatimeout 3
# see if the dispatcher can reach the backend WAS servers
dscontrol highavailability reach add 192.168.71.101 192.168.71.102  192.168.71.105
# set this dispatcher as the primary dispatcher
dscontrol highavailability backup add backup auto 9123
# print the status to screen
dscontrol highavailability status
```

goActive, goStandby, goInOp, and highavailChange: These are all exactly the same as on litlb01.

```
DATE=`date`
OUTPUT="$DATE LB just ran $1."
echo $OUTPUT >> /opt/ibm/edge/lb/servers/logs/lb.log
#echo $OUTPUT | mail -s "highavailChange" root@localhost
```

Some important notes on Dispatcher High Availability

To convert two Dispatcher machines configured for high availability to one machine running alone, stop the executor on one of the machines, then delete the high availability features (the heartbeats, reach, and backup) on the other.

When two Dispatcher machines are run in a high availability configuration and are synchronized, it is recommended that you enter all dscontrol commands (to update the configuration) on the standby machine first, and then on the active machine.

Testing Load Balancer High Availability

After running the HA.config scripts on both litlb01 and litlb02, we ran the following to check their high availability status:

```
litlb01:~ # dscontrol highavailability status

High Availability Status:
-------------------------
Role ................. Primary
Recovery strategy .... Auto
State ................ Active
Sub-state ............ Synchronized
Primary host ......... 192.168.71.99
Port ................. 9123
Preferred target ..... 192.168.71.135

Heartbeat Status:
-----------------
Count ................ 1
Source/destination ... 192.168.71.99/192.168.71.135

Reachability Status:
--------------------
Count ................ 3
Address .............. 192.168.71.101 reachable
Address .............. 192.168.71.102 reachable
Address .............. 192.168.71.105 reachable

litlb02:~ # dscontrol highavailability status

High Availability Status:
-------------------------
Role ................. Backup
Recovery strategy .... Auto
State ................ Standby
Sub-state ............ Synchronized
Primary host ......... 192.168.71.99
Port ................. 9123
Preferred target ..... 192.168.71.99
```

```
Heartbeat Status:
-----------------
Count ................ 1
Source/destination ... 192.168.71.135/192.168.71.99

Reachability Status:
--------------------
Count ................ 3
Address .............. 192.168.71.101 reachable
Address .............. 192.168.71.102 reachable
Address .............. 192.168.71.105 reachable
```

As you can see from the above output, litlb01's role was Primary and its state was Active, litlb02's was Backup and Standby, respectively.

Additionally, the primary dispatcher, because it was active it had the cluster interface eth0:1 up (activated by the script goActive). As below shows:

```
litlb01:~ # ifconfig
eth0      Link encap:Ethernet  HWaddr 02:00:00:00:00:12
          inet addr:192.168.71.99  Bcast:255.255.255.0  Mask:255.255.255.0
          inet6 addr: fe80::200:0:100:12/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1492  Metric:1
          RX packets:32366 errors:0 dropped:0 overruns:0 frame:0
          TX packets:31917 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2646547 (2.5 Mb)  TX bytes:2737535 (2.6 Mb)

eth0:1    Link encap:Ethernet  HWaddr 02:00:00:00:00:12
          inet addr:192.168.71.98  Bcast:192.168.71.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1492  Metric:1

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:1478 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1478 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:142837 (139.4 Kb)  TX bytes:142837 (139.4 Kb)
```

The backup Dispatcher, was on standby, so the cluster interface wasn't up (done by the script goStandby):

```
litlb02:/opt/ibm/edge/lb/servers/bin # ifconfig
eth0      Link encap:Ethernet  HWaddr 02:00:00:00:00:13
          inet addr:192.168.71.135  Bcast:192.168.71.255  Mask:255.255.255.0
          inet6 addr: fe80::200:0:100:13/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:179818 errors:0 dropped:0 overruns:0 frame:0
          TX packets:170163 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:11724244 (11.1 Mb)  TX bytes:12910700 (12.3 Mb)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:1832 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1832 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:177416 (173.2 Kb)  TX bytes:177416 (173.2 Kb)
```

We shutdown the dispatcher on litlb01:

```
litlb01:~ # dscontrol executor stop
Advisor 'Http' stopped on port 80.
The manager has been stopped.
Executor stopped at your request.

litlb01:~ # dsserver stop
Server stopping at your request.
```

Now the interface on litlb02 was active, and we can still access the Web application so the failover worked:

```
litlb02:/opt/ibm/edge/lb/servers/bin # ifconfig
eth0      Link encap:Ethernet  HWaddr 02:00:00:00:00:13
          inet addr:192.168.71.135  Bcast:192.168.71.255  Mask:255.255.255.0
          inet6 addr: fe80::200:0:100:13/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:180596 errors:0 dropped:0 overruns:0 frame:0
          TX packets:170940 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:11769666 (11.2 Mb)  TX bytes:12982112 (12.3 Mb)

eth0:1    Link encap:Ethernet  HWaddr 02:00:00:00:00:13
          inet addr:192.168.71.98  Bcast:192.168.71.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:1832 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1832 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:177416 (173.2 Kb)  TX bytes:177416 (173.2 Kb)
```

Implementing HA Reference Architecture: Non-WebSphere application – Apache Web Server

The following sections describe how we went about implementing HA Reference Architecture: Non-WebSphere application – Apache Web Server:

- "Implementing HA Web servers: Apache and Linux Virtual Server"
- "Building RealServer Images" on page 250
- "Installing LVS Director" on page 251
- "Implementing HA Web servers: Apache and Tivoli Systems Automation for Multiplatforms" on page 258
- "Comparing the two HA technologies: LVS and TSAM" on page 264

Implementing HA Web servers: Apache and Linux Virtual Server

We set out to construct a high availability Apache Web server solution using an open source software stack. Figure 67 on page 249 depicts our implementation using the Linux Virtual Server (LVS) components provided by linux-ha.org. All servers used in the testing of this scenario were SUSE Linux Enterprise Server 9.

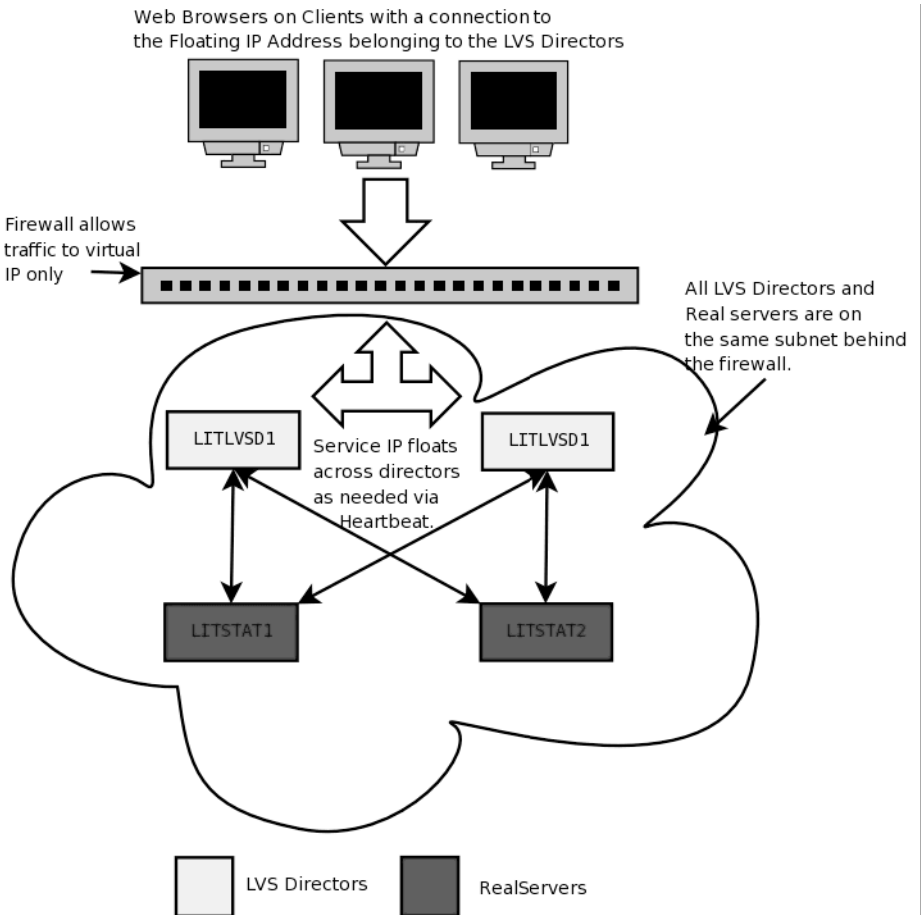


Figure 67. Our Linux Virtual Server (LVS) components.

The following describes some of the terminology used in implementing HA Reference Architecture: Non-WebSphere application – Apache Web Server:

LVS Directors

LVS Directors are systems that accept arbitrary incoming traffic and pass it on to any number of RealServers. They are then capable of receiving the response and passing it back to the clients who initiated the request. The directors need to perform their task in a transparent fashion such that clients never know that RealServers are doing the actual workload processing. LVS directors themselves need the ability to float resources (specifically a virtual IP address on which they listen for incoming traffic) between one another in order to not become a single point of failure. LVS Directors accomplish floating IP addresses by leveraging the Heartbeat component from LVS. This allows each configured director that is running heartbeat to ensure one, and only one, of the directors lays claim to the virtual IP address servicing incoming requests. Beyond the ability to float a service IP address, the Directors need to be able to monitor the status of the RealServers that are doing the actual workload processing. The Directors must keep a working knowledge of what RealServers are available for processing at all times. In order to monitor the RealServers, the LVS Directors use the Mon component of LVS. Specific configuration of Heartbeat and Mon for our implementation are found below in their respective sections.

RealServers

These systems are the actual Web server instances running the actual production workload that we wish to make highly-available. It is vital to have more than one RealServer providing the service you wish to make HA. In our environment we implemented only 2 RealServers, but adding more is trivially easy once the rest of the LVS infrastructure is in place. In our example the RealServers were running Apache Web Server, but other services could just as easily have been implemented (In fact we enabled SSH serving as well during testing). The RealServers we used are stock Apache Web servers with the notable exception that they were configured to respond as if they were using the LVS Directors' floating IP address, or a virtual hostname corresponding to the floating IP address used by the Directors. This is accomplished by altering a single line in the Apache configuration file.

All of our machines used in the LVS scenario described here resided on the same subnet in our network. Numerous other network topographies are described at the linux-ha.org website. We chose ours for simplicity. Since our clients must send requests through a firewall, we simply limit their traffic to the floating IP Address that is passed between the LVS Directors.

The Linux Virtual Server suite provides a few different methods to accomplish a transparent HA backend infrastructure. Each specific method has advantages and disadvantages. We chose LVS-NAT because the incoming packets are rewritten by the director to have the destination address of one of the RealServers and then forwarded to the RealServer. The replies from the RealServer are then sent back to the director where they are rewritten to have the source address of the floating IP address that clients are pointed at.

Unlike the other two methods of forwarding used in LVS (LVS-DR and LVS-Tun) the RealServer only needs a functional TCP/IP stack. This means that the RealServer can have virtually any modern operating system and no modifications are necessary to the configuration of the RealServers (except setting their route tables as we shall see later).

Building RealServer Images

We started by making two Linux server instances, each running Apache Web server. We ensured that the servers were working as designed by pointing a Web browser to each of the RealServers IP addresses. We ensured that each Apache instance was configured to listen on port 80 on its own IP address (i.e. on a different IP for each RealServer).

Next we configured the default Web pages on each server to display a static page containing the hostname of the machine serving the page. This was done to ensure we always knew which machine we were connected to during testing.

As a precaution we also checked that IP forwarding on these systems was OFF. You can do this by issuing the following command:

```
$> cat /proc/sys/net/ipv4/ip_forward
```

The output from this command will show a 0 if IP forwarding is off. If for any reason you need to disable it, simply issue the following

```
command: $> echo "0" >/proc/sys/net/ipv4/ip_forward
```

From an outside system you can run the open source tool nmap to verify that the virtual IP has HTTP either open or filtered in the STATE column. The real address should only show the ports that were open before configuring the virtual address:

```
[dowem@litsmb01 ~]$ nmap -P0 192.168.71.92

Starting nmap 3.70 ( http://www.insecure.org/nmap/ ) at 2006-01-13 16:58 EST
Interesting ports on 192.168.71.92:
(The 1656 ports scanned but not shown below are in state: closed)
PORT    STATE    SERVICE
22/tcp  open     ssh
80/tcp  filtered http
111/tcp open     rpcbind
631/tcp open     ipp
```

Next we pointed a Web browser at the RealServers actual IP addresses to ensure each was serving the appropriate page as expected. Once this was completed, we were able to move on to the LVS Director installation and configuration.

Installing LVS Director

On each of the LVS Directors it is strongly recommended that you add each of the RealServers to the /etc/hosts file. This will ensure there is no DNS related lag when servicing incoming requests.

At this time we double checked that each of the Directors was able to perform a timely ping to each of the RealServers used in our configuration:

```
$> ping -c 1 192.168.71.149
$> ping -c 1 192.168.71.150
```

Once that was completed, we used YaST to install ipvsadm, Heartbeat, and Mon from our install server. Recall that Heartbeat will be used for intra-director communication, and Mon will be used by each director to maintain information about the status of each RealServer.

Installing and Configuring Heartbeat on the Directors

We began by configuring Heartbeat between our Directors. Heartbeat makes the Directors reciprocally monitor one another, taking control of the floating IP address when necessary. To configure Heartbeat requires modifying 3 files. On SLES 9 the files can be found in the /etc/ha.d/ directory. The files are: haresources, ha.cf, and authkeys.

The first file we modified was the ha.cf file. This file controls the heartbeat daemon itself. It sets up logging, specifies time durations to be used before transferring the floating resource IP, specifies how each LVS director node should monitor the others, and lastly enumerates each of the LVS Directors. The sample file provided includes sample comments explaining each of the options in detail. When finished with our customization on our primary director, the file contained the following uncommented and altered lines:

```
debugfile /var/log/ha-debug
logfile /var/log/ha-log
logfacility     local0
keepalive 2
deadtime 8
ucast hsi0 10.2.1.94                    #Note 1
auto_failback on
watchdog /dev/watchdog
node    litlvsd1                        #Note 2
node    litlvsd2                        #Note 2
```

Notes:

1. We provided each of the Directors with an additional interface on a private network used only by the LVS directors. This interface was the Hipersocket interface hsi0 on both of our LVS Directors. In this line we specified that each node should send a unicast, or highly efficient and specifically targeted, ping to the secondary LVS Director (which had the 10.2.1.94 IP address on its hsi0 interface) over this high speed private network.
2. These lines provided the complete listing of all LVS Directors. As per the documentation provided with Heartbeat, these entries were specified according to the exact output of the command “uname -n” as issued on each Director. If the fully qualified domain name was returned, we would have had to use that instead.

Once this was done on the primary LVS director, we copied the file to our secondary director, and altered the unicast line so that it would be pinging the hsi0 device of the primary director. With that completed we turned our attention to the second file we needed to modify.

The second file we modified was the haresources file. This is the file that actually describes the floating IP address. The contents of this file must be identical on all LVS Director images in order for Heartbeat to function correctly. The sample file, as provided by SLES9, contained several examples suitable for modification with numerous comments. In our specific implementation we altered the file to contain a single uncommented line:

```
litlvsd1 192.168.71.92/24/eth0/192.168.71.255 lvs_startup
```

The IP address 192.168.71.92 was the IP address we chose to use as our floating resource IP. The device eth0 was already configured with the normal network address (192.168.71.91) of our primary director. This means that a virtual IP address alias will be created on the eth0:0 device of whichever director is responsible for maintaining the floating service IP at any given instant. The lvs_startup parameter is the name of any arbitrary init script to be invoked with the “start” and “stop” argument after the respective acquisition of the virtual service IP address or release respectively. We constructed such a script later on to trigger additional services to start and stop that were dependent on our floating resource IP (explained in further detail later in this document). Once our editing had been completed, we copied this file verbatim to the other Directors.

The third and final heartbeat related file is the authkeys file. This file needs to be created in the /etc/ha.d directory. The file specifies a shared secret key allowing directors to communicate with one another. We chose to use the sha1 method of encryption, but other methods are available, and documented on the linux-ha.org online documentation. We created our shared secret key by issuing the following command as root on one of the LVS Director images:

```
$> dd if=/dev/urandom count=4 2>/dev/null | md5sum | cut -c1-32
```

The output in our instance was “ca0e08148801f55794b23461eb4106db”. We use this output to create our authkeys file. The file need only contain 2 lines and should resemble the following:

```
auth 1
1 sha1 ca0e08148801f55794b23461eb4106db
```

For your application simply substitute your secret key in place of ours. We then copied this file verbatim to the other Directors.

Now that our configuration was completed, we set heartbeat to start at boot time on each of the directors. This was accomplished by issuing the following command on each Director:

```
$> chkconfig heartbeat on
```

This completed the necessary prep work to configure and install heartbeat. We restarted each of the LVS Directors to ensure the heartbeat service started properly at boot. As expected, we observed that the floating service IP was only on our primary director. By halting the machine that held the floating resource IP address, we watched as the secondary LVS Director image instantiated it within a matter of seconds. Upon bringing the halted director image back online, the floating resource IP was transferred back within seconds. We then retried these steps numerous times, while continuously pinging the floating resource IP address from another system on the same subnet. No packet loss occurred and we were now confident our floating resource IP was HA.

Creating LVS Rules with the IPVSADM Command

Our next step was to take the floating resource IP address and build upon it. Since LVS in our environment is intended to be transparent to remote Web browser clients, all Web requests had to be funneled through the directors, passed on to one of the RealServers, then any results needed to be relayed back to the director who then returns the response to the client who initiated the Web page request.

In order to accomplish that flow of requests and responses, we first configured each of the LVS directors to enable IP forwarding (thus allowing requests to be passed on to the RealServers). We did this by issuing the following commands:

```
$> echo "1" >/proc/sys/net/ipv4/ip_forward
$> cat /proc/sys/net/ipv4/ip_forward
```

If all was successful, the output from the cat command will show a ″1″. To make the change permanent across reboots, you need to modify the file /etc/sysconfig/sysctl and ensure the following line is present:

```
IP_FORWARD="yes"
```

Since we need to take over the floating IP address on this system, we added the following lines to the devices hardware config file ″/etc/sysconfig/hardware/hwcfg-qeth-bus-ccw-0.0.0700″.

```
# This enables IP Takeover
QETH_IPA_TAKEOVER=1
# Add IPA address
echo 192.168.71.92/24 > /sys/bus/ccwgroup/drivers/qeth/0.0.0700/ipa_takeover/add4
```

Next we needed to actually tell the directors to actually relay incoming HTTP requests on our HA floating IP address to the RealServers. This was accomplished with the ipvsadm command.

We began by issuing a command to clear the ipvsadm tables as follows:

```
$> /sbin/ipvsadm -C
```

Before we were able to configure our new tables, we had to consider the varying types of workload distribution the LVS Directors are capable of. On receiving a connect request from a client, the director assigns a RealServer to the client based on a ″schedule″. The scheduler type is set with the Linux command ipvsadm. The schedulers available are:

- Round Robin (rr) – new incoming connections assigned to each RealServer in turn.

- Weighted Round Robin (wrr) – RR scheduling with additional weighting factor to compensate for differences in real server capabilities such as additional CPUs, more memory, and so on.
- Least Connected (lc) - new connections go to RealServer with the least number of connections. This is not necessarily the least busy RealServer but is a step in that direction.
- Weighted Least Connection (wlc) – LC with weighting.

Thus we informed LVS that we want to listen for HTTP service requests on the floating IP address, specifying that we intend to handle these incoming requests using the RoundRobin scheduling algorithm provided by LVS.

Next we had to associate the RealServers:

```
# Forward HTTP to REAL_SERVER_IP_1 using the -m option (masquerading, network access translation, or NAT), with weight=1
$> /sbin/ipvsadm -a -t $VIRTUAL_CLUSTER_ADDRESS:http -r $REAL_SERVER_IP_1:http -m -w 1

# Second Real Server
# Forward HTTP to REAL_SERVER_IP_2 using LVS-NAT (-m), with weight=1
$> /sbin/ipvsadm -a -t $VIRTUAL_CLUSTER_ADDRESS:http -r $REAL_SERVER_IP_2:http -m –w 1
```

Note: If we had created additional RealServer instances, we would have simply issued similar commands for each in the same fashion.

At this time we double-checked our work by printing the ipvsadm table for inspection:

```
$> /sbin/ipvsadm
IP Virtual Server version 1.2.0 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  litlvsd_vip.ltic.pok.ibm.com rr
  -> litstat2.ltic.pok.ibm.com:ht Masq    1      0          0
  -> litstat1.ltic.pok.ibm.com:ht Masq    1      0          0
```

At this point requests coming in on the floating service IP will be rewritten and passed on to the RealServers, but in order to get traffic back from them we need to alter a few networking settings. Since we chose to implement our LVS Directors and RealServers in a flat network topology (ie all on the same subnet), we had to perform the following steps to force the Apache response traffic back through the Directors:

```
echo "0" > /proc/sys/net/ipv4/conf/all/send_redirects
echo "0" > /proc/sys/net/ipv4/conf/default/send_redirects
echo "0" > /proc/sys/net/ipv4/conf/eth0/send_redirects
```

This was done to prevent the active LVS Director from trying to take a TCPIP shortcut by informing the RealServer and floating service IP to talk directly to one another (since they are on the same subnet). Normally redirects are useful, as they improve performance by cutting out unnecessary middlemen in network connections, but in our case, it would have prevented the response traffic from being rewritten as is necessary for transparency to the client. In fact if redirects were not disabled on the LVS Director, the traffic being sent from the RealServer directly to the Client would appear to the client as an unsolicited network response and would be discarded.

Additionally we set the router of each of the RealServers to be the service floating IP address. This ensured all responses are passed back to the Director for packet rewriting before being passed back to the client that originated the request.

Once redirects had been disabled on the Directors, and the RealServers were configured to route all traffic through the floating service IP, we were ready to initially test our HA LVS environment. We accomplished this by pointing a Web browser on a remote client to the floating service address of our LVS Directors.

We used a Gecko based browser (Mozilla) in our testing. To ensure the deployment was successful we disabled caching in the browser, and clicked the refresh button multiple times. With each press of the refresh button we observed the Web page displayed was one of the pages we had configured on the RealServers. As per our decision to use RR, we observed the page cycling back and forth between each of the RealServers. At this point we wanted to ensure our LVS configuration would start automatically at boot. We did this by creating a simple init script, placing it in */etc/init.d*, without adding it to our default runlevel. We instead chose to have it start only when invoked from heartbeat (as it is dependent on the virtual service IP). We include it here for reference:

```
========== LVS DIRECTOR SAMPLE INIT SCRIPT ============
#!/bin/sh

# The virtual address on the Director which acts as a cluster address
VIRTUAL_CLUSTER_ADDRESS=192.168.71.94
REAL_SERVER_IP_1=192.168.71.149
REAL_SERVER_IP_2=192.168.71.150


#set ip_forward ON for vs-nat director (1 on, 0 off).
cat /proc/sys/net/ipv4/ip_forward
echo "1" >/proc/sys/net/ipv4/ip_forward

# Director acts as the gw for realservers
# Turn OFF icmp redirects (1 on, 0 off)
echo "0" >/proc/sys/net/ipv4/conf/all/send_redirects
cat       /proc/sys/net/ipv4/conf/all/send_redirects
echo "0" >/proc/sys/net/ipv4/conf/default/send_redirects
cat       /proc/sys/net/ipv4/conf/default/send_redirects
echo "0" >/proc/sys/net/ipv4/conf/eth0/send_redirects
cat       /proc/sys/net/ipv4/conf/eth0/send_redirects

# Checking if realservers are reachable from director
ping -c 1 $REAL_SERVER_IP_1
ping -c 1 $REAL_SERVER_IP_2

# Clear ipvsadm tables
/sbin/ipvsadm -C

# We install LVS services with ipvsadm
# Specifically we add HTTP to VIP with rr sheduling
/sbin/ipvsadm -A -t $VIRTUAL_CLUSTER_ADDRESS:http -s rr

# First Real Server
# Forward HTTP to REAL_SERVER_IP_1 using LVS-NAT (-m), with weight=1
/sbin/ipvsadm -a -t $VIRTUAL_CLUSTER_ADDRESS:http -r $REAL_SERVER_IP_1:http -m -w 1

# Second Real Server
# Forward HTTP to REAL_SERVER_IP_2 using LVS-NAT (-m), with weight=1
/sbin/ipvsadm -a -t $VIRTUAL_CLUSTER_ADDRESS:http -r $REAL_SERVER_IP_2:http -m -w 1

# We print the new ipvsadm table for inspection
echo "NEW IPVSADM TABLE:"
/sbin/ipvsadm
================= END OF INIT SCRIPT EXAMPLE =============
```

We called this init script “lvs_startup” as was alluded to in our earlier discussion on heartbeat installation. If you are following this guide, you need only alter the variables at the top to reflect your own IP addresses.

At this point we had a basic LVS setup in place, but it was still not completely HA. We had thus far established a highly available service IP address and bound that to our pool of RealServer instances. But this was not enough. Since we had chosen RR scheduling, should either RealServer become disabled, or cease to respond to network traffic for any reason, 50% of our HTTP requests would be failures. We needed to implement monitoring of the RealServers on each of the LVS directors in order to dynamically add or remove them from the service pool. To do this, we used another linux-ha.org component called Mon.

Installing and Configuring Mon on the LVS Directors

The Mon tool is the de facto standard for monitoring LVS RealNodes (but it is not the only solution). We found Mon to be easy to configure, and found it very extensible for those with scripting abilities. We discovered that there were essentially three main steps to get everything working: installation, service monitoring configuration, and creating some alerts. YaST handled the installation of Mon for us automatically, so we only needed to perform the monitoring configuration and creation of some alert scripts. The alert scripts are triggered when the monitors determine a RealServer has gone offline, or come back online.

Configuring Services to Monitor: By default Mon comes with several monitor mechanisms ready to be used. We altered the Mon configuration file to make use of the HTTP service. Mon came with a sample configuration file in */etc/mon.cf*.

In our configuration file we altered the header to reflect the proper paths. We installed Mon on a 64 bit image, and the sample configuration file was for the default (31 bit) locations. The configuration file sample assumed the alerts and monitors are located */usr/lib* which was incorrect for our installation. The parameters we altered were as follows:

```
alertdir     = /usr/lib64/mon/alert.d
mondir       = /usr/lib64/mon/mon.d
```

As you can see, we simply changed ″lib″ to ″lib64″.

The next change to the configuration file was specifying the list of real servers to monitor. This was done with the following 2 directives:

```
hostgroup litstat1 192.168.71.149 # Realserver1
hostgroup litstat2 192.168.71.150 # Realserver2
```

Once we had defined the hosts we were going to watch, we needed to tell Mon how to watch for failure, and what to do in case of failure. To do this, we added the following 2 sections (one for each RealServer).

```
# Watch first real node to add or remove from cluster
watch litstat1
    service http
        interval 3s
        monitor http.monitor -p 80 -t 5 -u /index.html
        allow_empty_group
        period wd {Mon-Sun}
            alert bring-node-down.alert "litstat1"
            upalert bring-node-up.alert "litstat1"
            alertevery 600s
                alertafter 1
```

```
# Watch second real node to add or remove from cluster
watch litstat2
    service http
        interval 3s
        monitor http.monitor -p 80 -t 5 -u /index.html
        allow_empty_group
        period wd {Mon-Sun}
            alert bring-node-down.alert "litstat2"
            upalert bring-node-up.alert "litstat2"
            alertevery 600s
                alertafter 1
```

We are telling Mon to use the http.monitor which is shipped with Mon by default. Additionally we had specified the ports to use and the page to request (thus you can transmit a more efficient small segment of html as proof of success rather than a complicated html page). We defined the ″alert″ and ″upalert″ lines to invoke scripts we wrote and placed in the ″alertdir″ specified at the top of the configuration file. The alerts are responsible for telling LVS to add or remove RealServers from the pool (by invoking the ipvsadm command as we shall see in a moment). When one of the RealServers fails the http test, the bring-node-down.alert script will be executed with a string as argument. We constructed our alerts to be generic, and passed in the name of the host that failed as an argument. Likewise when the monitor determines that a RealSever has come back online it executes the bring-node-up.alert with the hostname as argument.

We saved this file, and created the alerts, using Perl, in the ″alertdir″ (which is specified at the top of the configuration file) as follows:

```
================BRING-NODE-UP.ALERT ============
#!/usr/bin/perl

use Getopt::Std;
getopts ("s:g:h:t:l:u");

# The only arg is the host to bring down

$VIRTUAL_CLUSTER_ADDRESS = "192.168.71.92";

$REAL_SERVER_IP=<STDIN>;
chomp $REAL_SERVER_IP;

# We add the server
system("/sbin/ipvsadm -a -t $VIRTUAL_CLUSTER_ADDRESS:http -r $REAL_SERVER_IP:http -m -w 1");

#print("\n\n\AUTOMATICALLY EXECUTING /sbin/ipvsadm -d -t $VIRTUAL_CLUSTER_ADDRESS:http -r $REAL_SERVER_IP\n");
============== END BRING-NODE-UP.ALERT ==========

============== BRING-NODE-DOWN.ALERT ==========
#!/usr/bin/perl

use Getopt::Std;
getopts ("s:g:h:t:l:u");

# The only arg is the host to bring down
$VIRTUAL_CLUSTER_ADDRESS = "192.168.71.92";

$REAL_CLUSTER_ADDRESS=<STDIN>;
chomp $REAL_CLUSTER_ADDRESS;

# We remove the server
system("/sbin/ipvsadm -d -t $VIRTUAL_CLUSTER_ADDRESS:http -r $REAL_CLUSTER_ADDRESS");
========== END BRING-NODE-DOWN.ALERT ==========
```

Both of those scripts were designed to make use of the ipvsadm command line tool to dynamically add and remove real servers from the LVS tables.

We were not forced to use Perl for scripting, in fact any scripting language would have been appropriate for the task. The examples shipped with Mon were already implemented in Perl, and this script was modified from one of those. In retrospect, for this specific case, a 3 line bash script would have probably been more efficient.

We used chkconfig to make sure Mon starts at boot.

Testing Failover of Linux Virtual Server and Apache

We tested the transfer of the floating resource IP on the directors as indicated in section “Installing and Configuring Heartbeat on the Directors” and found it to be seamless. Now we tested by driving Web traffic to the highly available service IP which was on one of the two LVS directors, which was the same IP address that the Apache servers were configured to listen on. We halted the director with the virtual IP address and there was no noticeable lag or failure observed from the client side driving the traffic.

Each of the directors used Mon (through the ping plugin) to maintain state of which Apache RealServers were available to process work, adding and removing them as they went on or offline. We tested the failover of the Apache RealServers by halting one of the Apache RealServers while driving Web traffic to the highly available service IP that they are bound on. The time to failover a RealServer, and have the directors take the appropriate action to remove said real server from its internal queue of hosts it may send work to, was transparent from our testing. This meant that from the client side we didn’t notice a lag in our Web traffic. When we brought back the failed Apache RealServer, the time to notice a RealServer was back up, or the time that the directors took to add it to the queue of available hosts, was also transparent, and the server began picking up traffic immediately.

Implementing HA Web servers: Apache and Tivoli Systems Automation for Multiplatforms

Implementing HA Web servers: Apache and Tivoli Systems Automation for Multiplatforms consists of the following sections:

Tivoli Systems Automation for Multiplatforms (TSAM)

This section describes the process of instantiating a high-availability Web server on the Linux on zSeries platform using Tivoli System Automation.

It is our contention that the following requirements must be met to qualify as a HA Web Server:

1. The Web server should be able to start on any node in the cluster, but will only run on one node at any point in time.
2. The Web server should be restarted automatically on the same or another node in the cluster in case of a failure. This mechanism also allows a planned outage of nodes for service and maintenance.
3. The Web server should be addressable with the same IP address regardless of the node it currently runs on. Thus the location of the Web server is transparent outside the cluster where no adaptation has to be performed, when the Web server is moved from one node to another.

TSAM works by defining various interdependency and equivalency relationships. Conceptually to create an HA Apache Web server implementation, we need an HA IP address that may float across any of the actual cluster node interfaces. In TSAM parlance, this is referred to as an equivalency relationship among the interfaces. At

whichever machine contains the floating IP address, the Apache process must be started. Thus the Apache application itself has a dependency relationship on the floating IP address.

This section describes the construction of such a system using TSA with Reliable Scalable Cluster Technology, or RSCT. RSCT is a product fully integrated into IBM Tivoli System Automation. RSCT provides three basic components, or layers, of functionality:

1. RMC (Resource Monitoring and Control), provides global access for configuring, monitoring, and controlling resources in a peer domain.
2. HAGS (High Availability Group Services), is a distributed coordination, messaging, and synchronization service.
3. HATS (High Availability Topology Services), provides a scalable heartbeat for adapter and node failure detection, and a reliable messaging service in a peer domain.

As of 2Q 2006, the support statement available at:

`http://www-306.ibm.com/software/tivoli/products/sys-auto-linux/platforms.html`

lists the following supported platforms:

On zSeries:

```
* SuSE Linux Enterprise Server 8 (SLES 8) SP 3 31, 32, 64 bit
* SuSE Linux Enterprise Server 9 (SLES 9) 32, 64 bit
* Red Hat Enterprise Linux 3.0 31/32, 64 bit
```

Following the IBM 64 Bit strategic vision, we chose to move forward with the SLES 9 64 bit approach.

Note: To run IBM Tivoli System Automation on SUSE SLES9 on Linux on zSeries, the Service Pack 1 of SUSE SLES9 has to be installed. This is because the kernel module softdog.o is not available in SuSE SLES9 without any Service Pack on Linux on zSeries.

With respect to networking, the zSeries platform also supports Hipersockets, CTC, and VM Guest LAN as indicated in the PDF *"IBM Tivoli System Automation Guide and Reference"*.

Pre-Requisites for each node

Before beginning installation we had to ensure the Linux images being used had the appropriate prerequisite software and sizing:

1. Korn Shell (pdksh)
2. Perl (NOT 5.8 due to character display problems)
3. 100MB (or more) free in */usr/sbin*
4. 100MB (or more) free in */var*

We next ensured there were no additional operating system specific requirements by checking the following:

`http://www-306.ibm.com/software/tivoli/products/sys-auto-linux/requirements.html`

At the time of this writing there were no zSeries entries were listed.

Next, on each node, we set an environment variable for all users of TSA. This was done by adding the following 2 lines at the very end of the */etc/profile* on each node. This ensured all users would have the definition.

```
# This is needed for TSAM (Tivoli)
export CT_MANAGEMENT_SCOPE=2
```

Once the essential prerequisites were over, we moved our attention to the necessary installation and configuration of a fully working Server installation. In our testing we used the open source Apache Web server. Each node was equipped with a typical fully working installation with a different default Web page. We made the default page show only the hostname of the node supporting the server, which was done to facilitate testing later on.

Installing TSAM software

On each node to be automated, we installed several packages (rpms). Bundled with the TSAM software was a pair of scripts called installSAM and uninstallSAM which are supplied to ensure that packages are installed or uninstalled in the correct order. The scripts and the RPM files were made available on nodes where IBM Tivoli System Automation was being installed. On zSeries the directory containing these files is called s390. According to the PDF, these packages are suitable for the s390 (zSeries 31 bit) and s390x (zSeries 64 bit) architecture.

We obtained our TSAM installation as a download. The installation procedure began with extracting the file by using the following command:

```
$> tar -xvf <tar file>
```

Next we changed into the s390 specific directory:

```
$> cd SAM12/s390/
```

We were then able to begin installation by executing the aforementioned installSAM script:

```
$> ./installSAM
```

Before installation starts, the License Agreement and the License Information is displayed. You can scroll forward line by line using the “Enter”, and page by page using the “spacebar”. Once we had scrolled to the bottom of the License information file we accepted the License Information by typing “y”. Any other input will cancel the installation.

Once installation had completed, we checked the installation as follows:

```
$> rpm -qa | grep -E "^src|^rsct|^sam"
```

This presented a list of the packages that had just been installed. You may wish to make a record of this for your own installation.

Installing the license

License installation should happen automatically when running the install script, but should problems arise, this is included for completeness. You may need to refer to this section if you are upgrading from a ″Try and Buy″ license to a full license.

The license comes with the installation medium in the ″license″ subdirectory.

To install the license execute the following command:

```
$> samlicm -i </path/to/license/file>
```

To view the license:

```
$> samlicm -s
```

Overview of TSA Commands

Before beginning, we familiarized ourselves with the basic TSAM command line tools. They are presented below with brief explanation. It is recommended that anyone reproducing this installation become familiar with these tools at a high level before proceeding.

preprpnode
: This command prepares the security settings for the node to be included in a cluster. When issued, public keys are exchanged among the nodes, and the RMC access control list (ACL) is modified to enable access to cluster resources by all the nodes of the cluster.

mkrpdomain
: This command creates a new cluster definition. It is used to specify the name of the cluster, and the list of nodes to be added to the cluster.

lsrpdomain
: This command lists information about the cluster to which the node where the command runs belongs.

startrpdomain/stoprpdomain
: These commands are used to bring the cluster on-line and offline, respectively.

addrpnode
: Once a cluster has been defined and is operational, this command is used to add new nodes to the cluster.

startrpnode/stoprpnode
: These commands are used to bring individual nodes on-line and offline to the cluster. They often used when performing maintenance to a particular system. The node is stopped, repairs or maintenance is performed, then the node is restarted, at which time it rejoins the cluster.

lsrpnode
: This command is used to view the list of nodes defined to a cluster, as well as the operating state (OpState) of each node. Note that this command is useful only on nodes that are Online in the cluster, otherwise it will not display the list of nodes.

rmrpdomain
: This command removes a defined cluster.

rmrpnode
: This command removes one or more nodes from a cluster definition.

Configuring our own cluster

While performing the installation we had terminals with a SSH session open to each node as root. To ensure that our earlier modification to the /etc/profile had taken effect we performed:

```
$>  echo $CT_MANAGEMENT_SCOPE
```

The resultant output should be ″2″ on each node.

Next, on each node we needed to enable communication with the other cluster nodes by issuing:

```
$> preprpnode <node01> <node02> ... <nodeN>
```

Now that the nodes were prepped for communication, we created a cluster with the name ″SA_Domain″ on our nodes. You can name your cluster anything you like as long as you only use characters {A-Z, a\u2013z, 0-9, ., _}. The command to create a cluster was issued from only one node:

```
$>mkrpdomain SA_Domain <node01> <node02> ... <nodeN>
```

We checked our work so far by examining the status of our ″SA_Domain″:

```
$> lsrpdomain
```

We observed output that indicated our cluster was now defined but in an offline state. Thus it was time to bring the cluster on-line by issuing:

```
$> startrpdomain SA_Domain
```

At this point issuing the lsrpdomain command a few times showed that the cluster was in the process of starting up (OpState is Pending Online) for a few moments. Approximately a minute later the status of the cluster indicated it was started. The status indicating such was “cluster is now online″.

It was then time to create the application resource for our cluster. Application resources require a definition just like the other types of resources used in TSAM, but their definitions must also include the path to a script capable of stopping and starting the application. To facilitate this requirement we created the following script and placed in /opt on each node of our cluster.

```
---------------------------------  TSAM_APACHE.sh ------------------------

#!/bin/bash
OPSTATE_ONLINE=1
OPSTATE_OFFLINE=2
Action=${1}
case ${Action} in
        start)
          /usr/sbin/apachectl start >/dev/null 2>&1
          logger -i -t "SAM-apache" "Apache started"
          RC=0
          ;;
 stop)
          /usr/sbin/apachectl stop >/dev/null 2>&1
          logger -i -t "SAM-apache" "Apache stopped"
          RC=0
          ;;
 status)
          ps -ax |grep -v "grep"|grep "/usr/sbin/httpd">/dev/null
          if [ $? == 0 ]
          then
           RC=${OPSTATE_ONLINE}
          else
           RC=${OPSTATE_OFFLINE}
          fi
          ;;
esac
exit $RC
------------------------ END OF SCRIPT -------------------------------------
```

Once that was finished, we created the actual application definition file. We named ours apache.def, and constructed it to look like the following:

```
PersistentResourceAttributes::
    Name="apache1"
    StartCommand="/opt/TSAM_APACHE.sh start"
    StopCommand="/opt/TSAM_APACHE.sh stop"
    MonitorCommand="/opt/TSAM_APACHE.sh status"
    MonitorCommandPeriod=5
```

```
    MonitorCommandTimeout=5
    NodeNameList={"<node01>","<node02>","<node03>"}
    StartCommandTimeout=10
    StopCommandTimeout=10
    UserName="root"
    ResourceType=1
```

This file only needed to be created and saved on a single node. We placed the file in the /opt directory. Then, from the same node, we used the definition file to create our resource definition with the mkrsrc command:

```
$> mkrsrc -f /opt/apache.def IBM.Application
```

The Web server's floating virtual IP address, which we labeled "apache1IP", is a distinctly separate IP address that is not matched to any IP address assigned to the network adapters of the actual cluster nodes.

When the Web server process must be moved to a new node (for whatever reason), the floating virtual address is brought offline from the failing node, then dynamically recreated on the new node. At that point the Web server process needs to be dynamically started as well. In order to ensure this happens, we defined a resource (to represent the floating virtual IP address of the cluster) that consists of an IP address which may exist on any of our nodes:

```
$>mkrsrc IBM.ServiceIP NodeNameList="{'node1','node2','node3'}"
Name="apache1IP" NetMask=255.255.255.0 IPAddress=<your_floating_IP_addr>
```

The following command creates an equivalency among the actual network interfaces. We chose to name the equivalency relationship "netequ". We defined the equivalency to TSAM by issuing:

```
$> mkequ netequ IBM.NetworkInterface:eth0:node01,eth0:node02,eth0:node03
```

Next, to turn our newly created resources, namely "apache1" and "apache1IP", into managed resources they had to be added to a resource group. Since we had not yet defined such a resource group, we added a new one that we called "apacherg". TSAM creates resource groups with the mkrg command as illustrated here:

```
$> mkrg apacherg</>>
```

We were then able to add the apache process resources (apache1) and the floating IP (apache1IP) to the resource group apacherg by using the addrgmbr command. By adding the resources to the resource group we effectively make them into managed resources:

```
$> addrgmbr -g apacherg IBM.Application:apache1
$> addrgmbr -g apacherg IBM.ServiceIP:apache1IP
```

Next we created a managed relationship named ″apache1_dependson_ip1″ that establishes the dependency of our application resource (apache1) on the floating virtual IP address we called (apache1IP). To make this relationship we simply issued:

```
$> mkrel -p DependsOn -S IBM.Application:apache1 -G IBM.ServiceIP:apache1IP apache1_dependson_ip1
```

Then we defined a second relationship called "apache1IP_dependson_netequ", that solidified the relationship between the virtual floating IP address (apache1IP), and the equivalent physical network interfaces (netequ) on our nodes:

```
$> mkrel -p DependsOn -S IBM.ServiceIP:apache1IP -G IBM.Equivalency:netequ apache1IP_dependson_netequ
```

We observed that the managed groups default to offline, thus we changed it such that our resource group (apacherg) is on-line:

```
$> chrg -o online apacherg
```

After bringing the resource group on-line, we were ready to figure out which machine was running the master daemon. In order to determine this we issued:

```
$> lssrc -ls IBM.RecoveryRM | grep Master
```

The machine running the master instance should have the floating IP address up, and should have the Apache process started. No other machine in the cluster should meet either of those criteria.

From any node, to save the current SAM configuration for our cluster we simply issue:

```
$> samcfg -S
```

or you can opt to use the recommended command which will store the configuration in an XML format:

```
$> sampolicy -s  'filename'
```

This will save your info into a stock location where DOMAIN_NAME is the domain name you used earlier:

```
$> ls /var/ct/DOMAIN_NAME/cfg/DOMAIN_NAME(pile of numbers here)
```

For example:

```
$> ls /var/ct/APACHE_DOMAIN/cfg/APACHE_DOMAIN022206.154007
```

Testing Apache Failover with TSAM

To test the Apache failover we directed Web traffic to the virtual IP and then took down the system with the virtual IP. The failover of the virtual IP address was about 2 seconds (where TSAM brought up the virtual IP dynamically on another node in the cluster) and the failover to HTTP service was about 4 seconds. So the total outage time between a cutover was ~6 seconds or so. The outage time was dominated by starting the new Apache process since TSAM had to start Apache on another node in the cluster on demand. Please keep in mind that the time will vary depending on the application and system resources.

Comparing the two HA technologies: LVS and TSAM

LVS and TSAM can be used to provide similar functionality, but the implementations of each have some interesting ramifications. TSAM is geared towards dependency resolution, and dynamic allocation of resources. For instance when TSAM realizes a failure has happened, it then brings up the virtual IP address dynamically on another node in the cluster, then starts a new instance of Apache on one of the backup servers ″on demand″.

Contrast this to the LVS solution which has full running Apache instances on all nodes at all times, and only cuts over the virtual IP address. Since the LVS approach involves many essentially identical servers running in a hot standby mode. This behavior means faster fail-over times in our testing because there was no overhead incurred from starting additional Apache processes on the fly. However, this approach does require a full instantiation of the Apache processes to be running on the backup servers at all times, and thus can be more resource intensive.

This behavior is expected as they are completely different approaches to HA. In summation, it is a classic example of resources versus time trade off. Dynamically

instantiating server processes can take more time, but might make sense in your environment if you cannot incur the increased system resource requirements.

Additionally, please keep in mind that TSAM has lots of enterprise-class capabilities that aren't demonstrated with the simple Apache failover scenario that we included in the architecture. TSA is quite adaptable and can support very complex topologies, including end-to-end support of heterogeneous clusters. It also has flexible, policy-driven management where user-written policies can control recovery actions. As a side note, it also has nice grouping capabilities that allow disparate resources to be conveniently halted or relocated in groups (with collocation or affinity logic) to prepare for planned outages. With that said, you should take care in planning which HA implementation is best for your environment and business needs.

Implementing Highly Available Stonesoft StoneGate Firewall

Our environment has made use of software firewalls to protect our virtual network's demilitarized zone (DMZ) for some time. However, in the past we had relied exclusively on the open source netfilter/iptables support to provide this function. For our latest test, we wanted to incorporate a commercial firewall and configure it for high availability. We decided to try StoneGate from Stonesoft. Stonesoft notes on their website:

`http://www.stonesoft.com/`

"With the release of StoneGate for IBM zSeries, enterprises can now protect their virtual servers and virtual networks with firewall and VPN technology to secure their data centers."

"StoneGate for IBM zSeries removes the need for external firewall servers between front and back end applications, saving costs in investment, labor and maintenance, enabling multiple simultaneous firewalls in a virtual network environment, and utilizing the unmatched scalability of a mainframe. StoneGate for zSeries is available through your local IBM zSeries/Global services representatives or by contacting sales@stonesoft.com. Visit www.stonesoft.com for more information."

There were basically four steps to implement the StoneGate Firewall in our environment:

1. "Installing the Management Center"
2. "Configuring the Firewall Cluster" on page 266
3. "Installing the Firewall Engines" on page 274
4. "Defining the rules" on page 280

We used the following Stonesoft documentation which can be found on their website:

`http://www.stonesoft.com/`

StoneGate Firewall/VPN 2.2.10 for IBM zSeries: Release Notes
StoneGate High Availability Firewall and Multi-Link VPN Version 2.6 Installation Guide
StoneGate Firewall Engine Clustering on IBM zSeries: How-to Guidelines
StoneGate Administrator's Guide

Installing the Management Center

We reviewed the StoneGate Firewall/VPN 2.2.10 for IBM zSeries: Release Notes document. It discusses system requirements, compatibility, installation instructions, known limitations and issues.

We followed chapter 3 in the *Installation Guide* to install the Management Center on a Windows 2000 Server in our lab. We chose to install all the Management Center components which included the Administration Client, Management Server, Log Server and Monitoring Server. The install went smoothly but we did experience one problem accessing the GUI. In the release notes it instructs you to add the following line into *SGClientConfiguration.txt* in the user home directory on the Administration client machine:

```
GUI_FWPropertiesForS390=True
```

For zSeries, the proper syntax was:

```
FWPropertiesForS390=True
```

Configuring the Firewall Cluster

For this step, we followed the *StoneGate Firewall Engine Clustering on IBM zSeries: How-to Guidelines* document. We configured the firewall elements with the Administration Client using the Configuration View and generated an initial configuration for the engines. When the initial configuration is saved, unique passwords are generated for each engine and are required later during the engine install.

A StoneGate firewall cluster communicates through Network Interface Cards (NIC). There are two types of interfaces, Cluster Virtual Interface (CVI) which handles the operative traffic and Node Dedicated Interfaces (NDI) which are used for control and management communication.

We defined a firewall cluster named LITCLUSTER, as shown in Figure 68 on page 267. The following are screen shots from the Administration Client. They are being provided so that you can see our configuration and possibly use them as a guide. The first displays the active firewall cluster. Under the Status column, you will see a clover shaped icon (which represents the cluster) and two double bars (which represent the engines). If this was in color, the cluster and one engine would be green indicating they are active / online. The other engine would be cyan indicating standby mode.

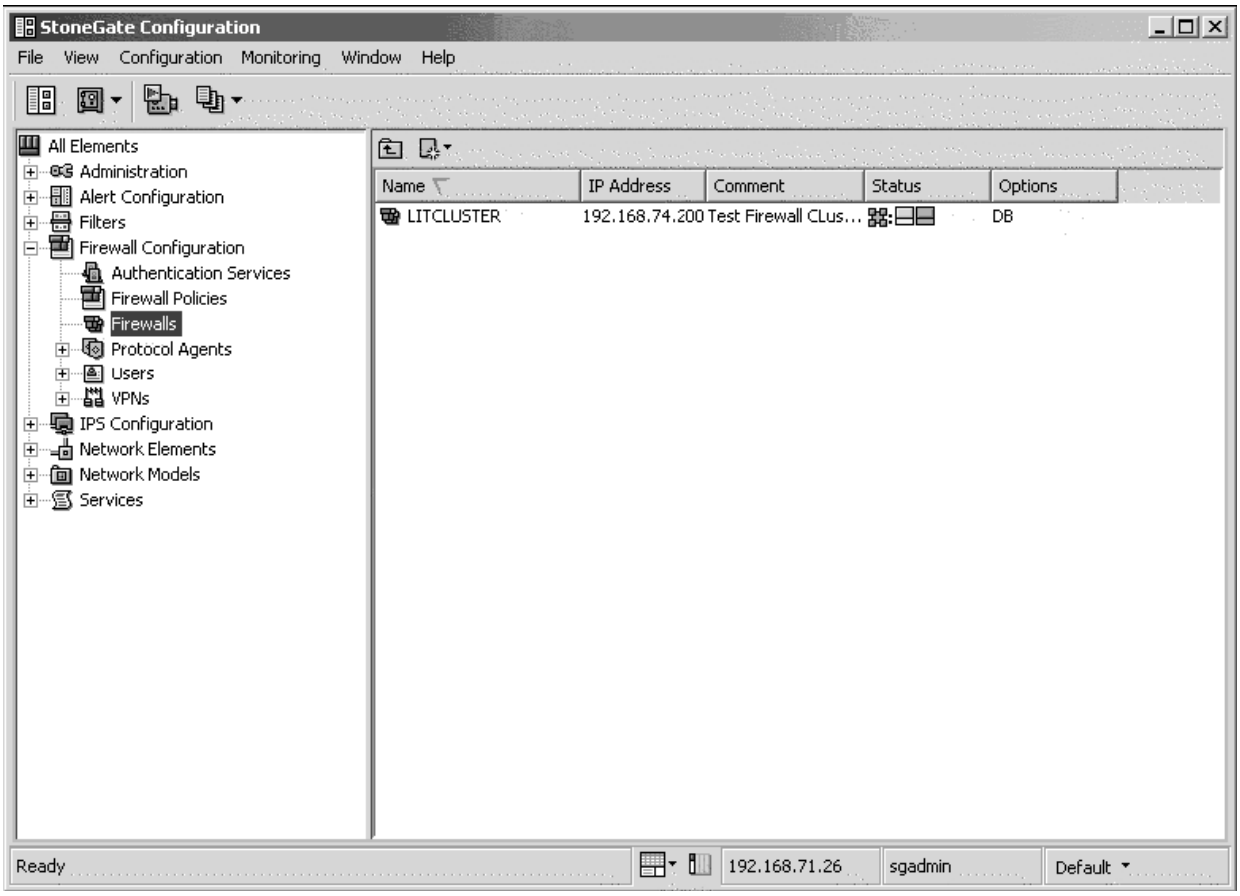


Figure 68. Our Firewall cluster named LITCLUSTER.

We selected the Cluster tab to display all the interfaces we defined, as shown in Figure 69 on page 268.

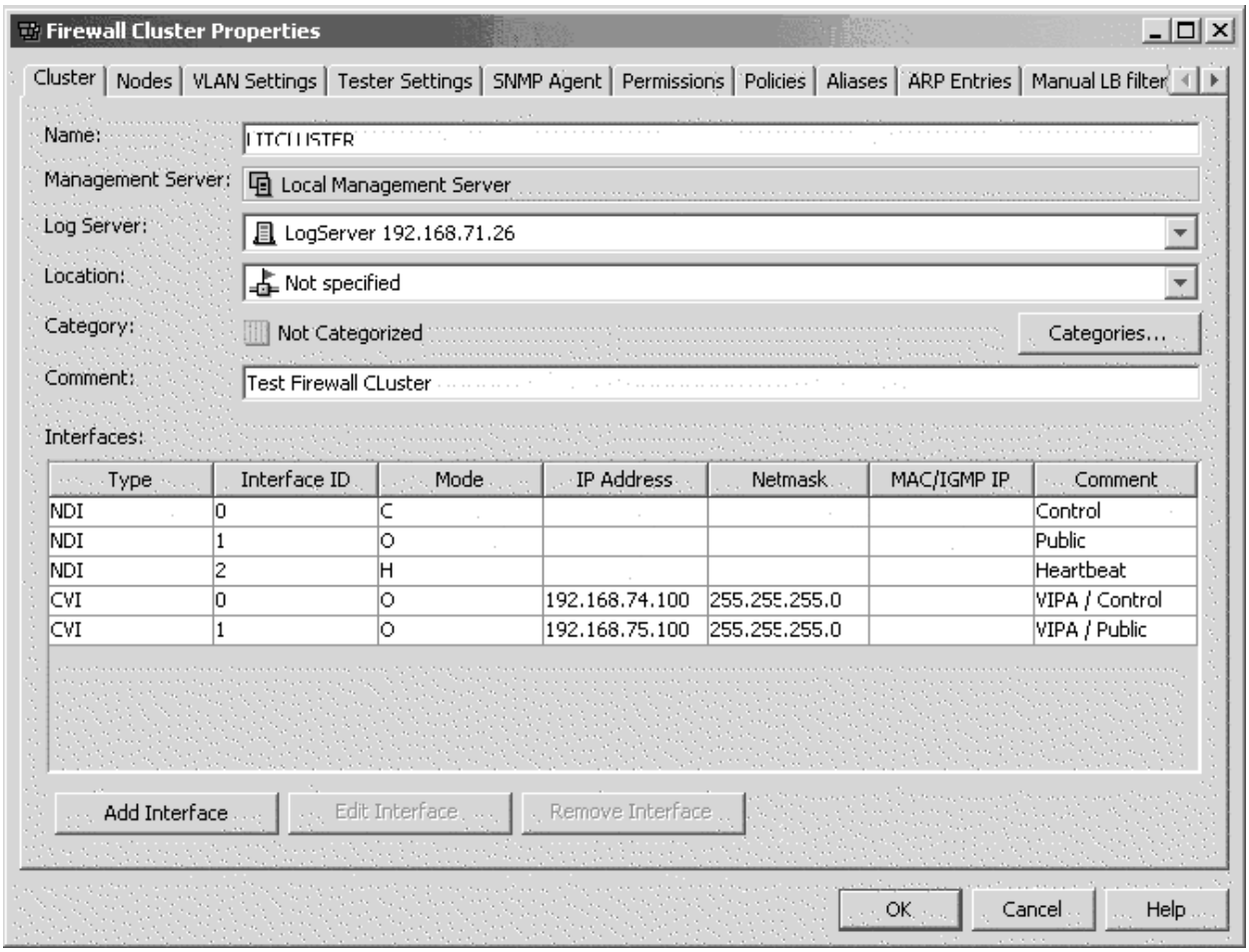


Figure 69. Cluster tab displaying all the interfaces we defined.

We selected the first NDI (Interface 0) to show the Primary Control definition, as shown in Figure 70 on page 269.

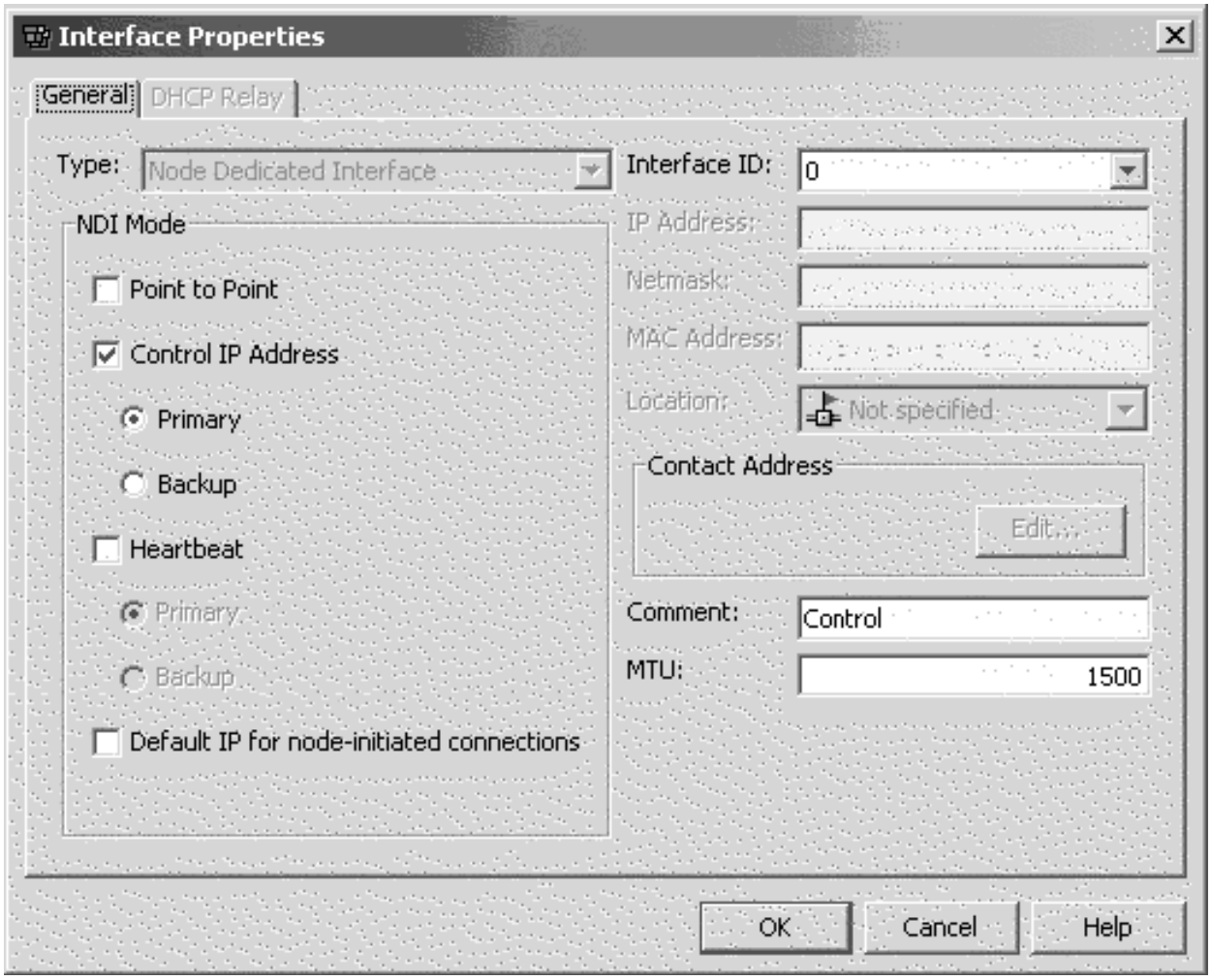


Figure 70. Primary Control definition.

We selected the third NDI (Interface 2) to show the Primary Heartbeat definition, as shown in Figure 71 on page 270.

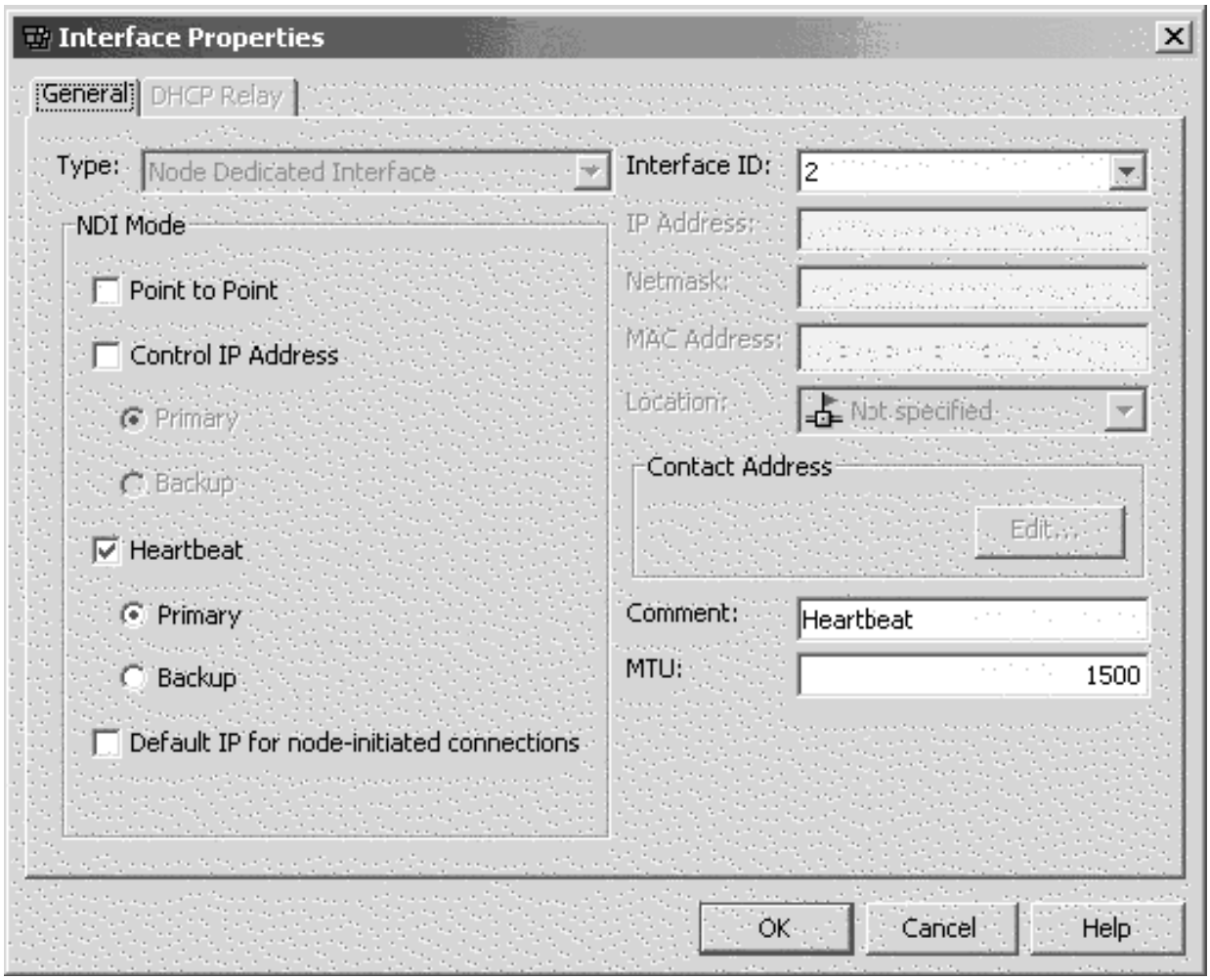


Figure 71. Primary Heartbeat definition.

We selected the first CVI (Interface 0) to show the Private LAN definition. Note that the IP Address is a VIPA (Virtual IP Address) and CVI Mode has the IP Address Take Over option selected, as shown in Figure 72 on page 271.

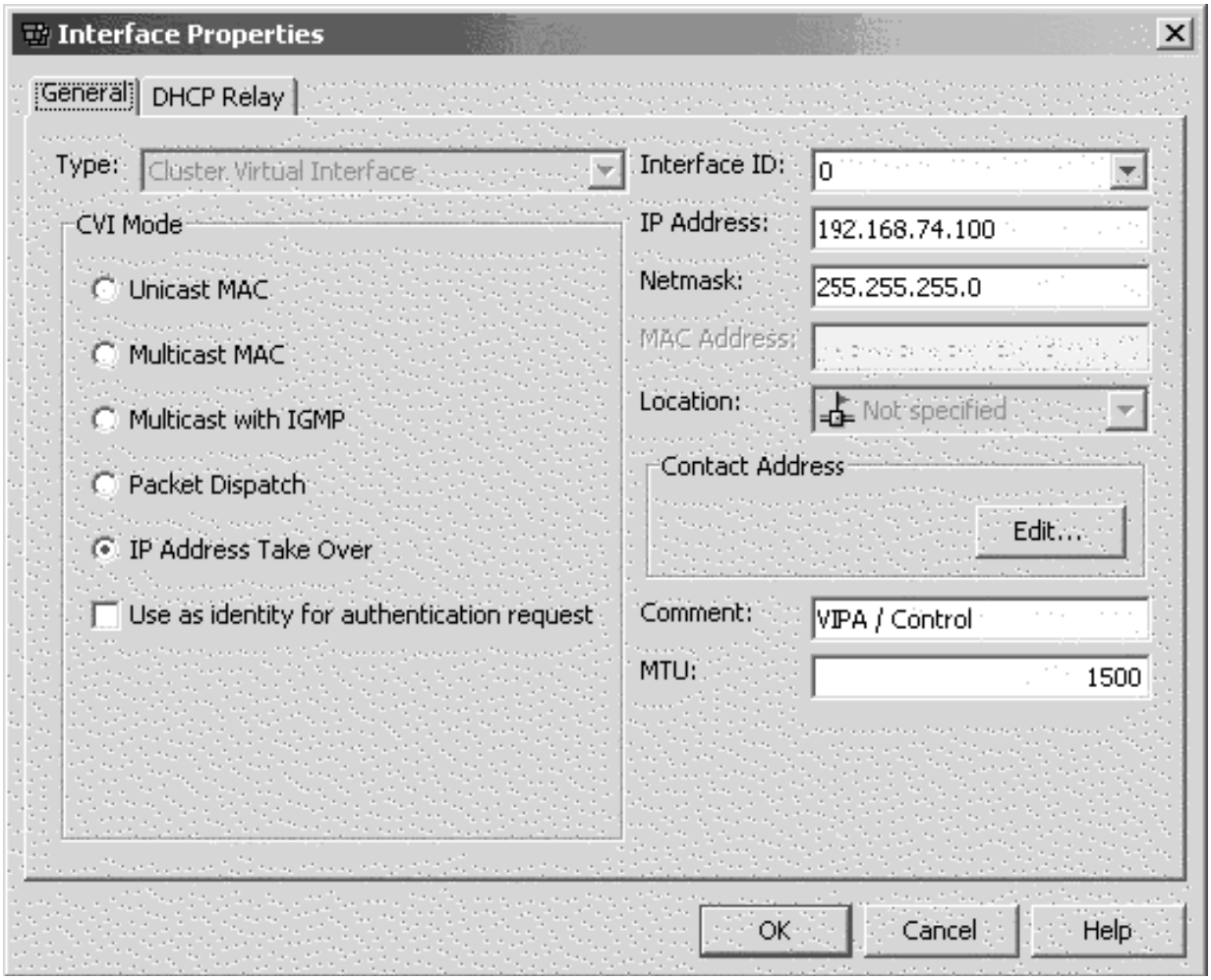


Figure 72. Private LAN definition.

We selected the second CVI (Interface 1) to show the Public LAN definition. Again, the IP Address is a VIPA (Virtual IP Address) and CVI Mode has the IP Address Take Over option selected, as shown in Figure 73 on page 272.

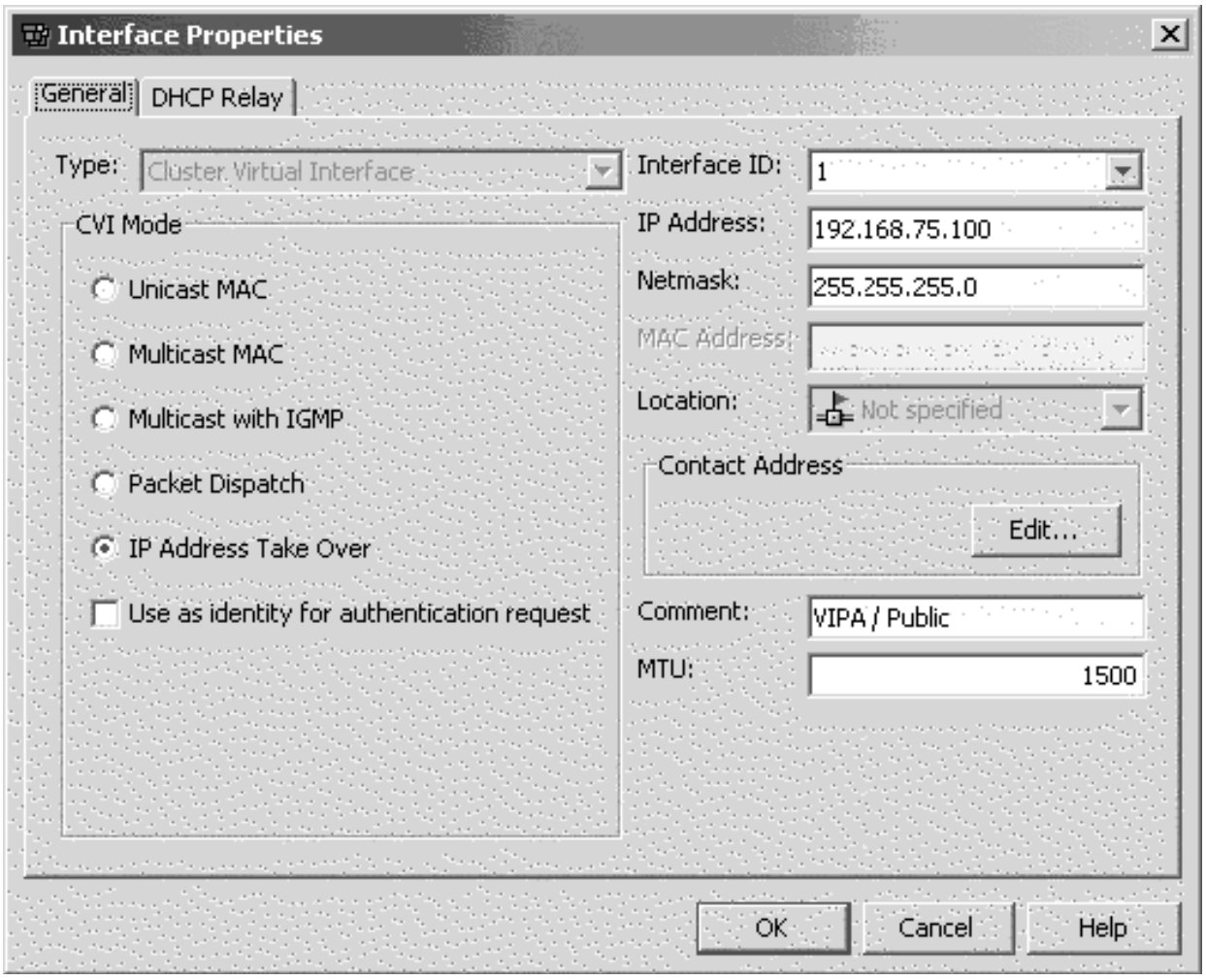


Figure 73. Public LAN definition.

We selected Nodes tab, then LITSGFW1 to show this node's IP definitions, as shown in Figure 74 on page 273.

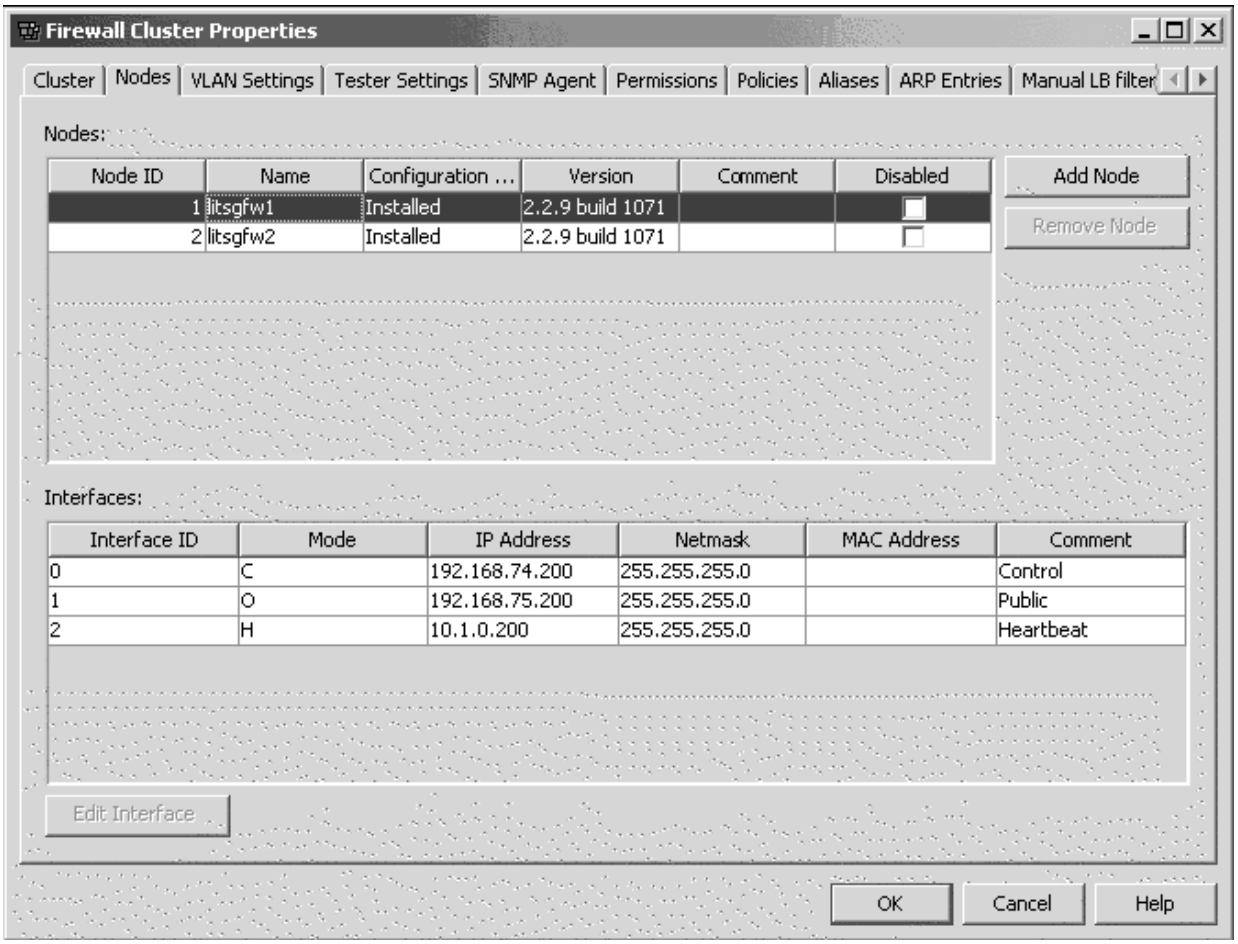


Figure 74. LITSGFW1 IP definitions.

We selected Nodes tab, then LITSGFW2 to show its IP definitions, as shown in Figure 75 on page 274.

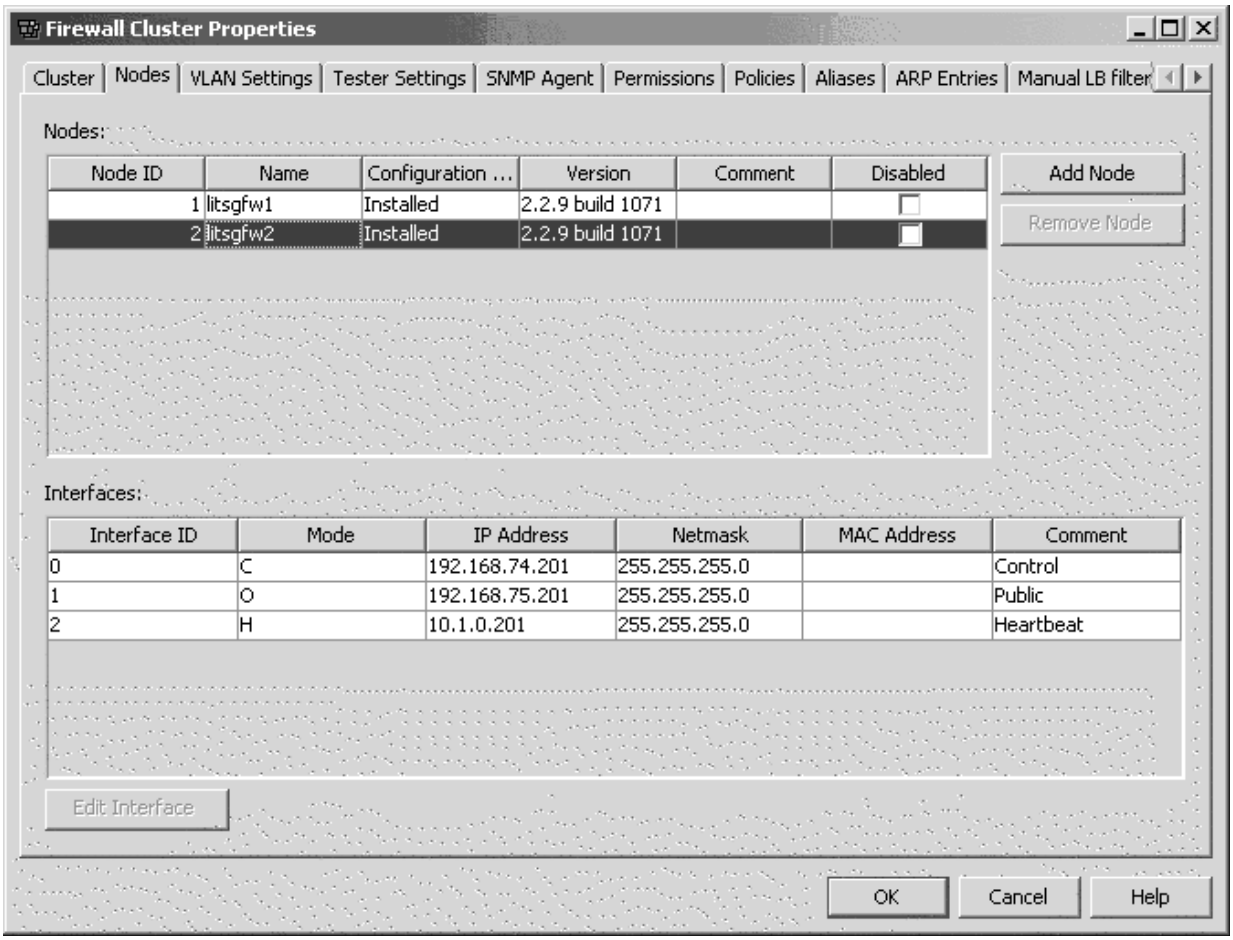


Figure 75. LITSGFW2 IP definitions.

Installing the Firewall Engines

In our environment, we are running z/VM 5.2 on a z990 processor. Our plan was to configure a firewall cluster to enable High Availability functionality. In the cluster, two nodes were defined, one online, the other in standby mode. This particular configuration required an OSA Express with Layer 2 support (Link Layer). For further details of this feature, go to:

`http://www.ibm.com/systems/z/networking/features2.html`

The MAC address associated with the primary server gets transferred over to the standby node in the event of a failure and only OSA Layer 2 provides unique MAC addressing capabilities. We added a second OSA Layer 2 card for redundancy but it's not required. StoneGate also has a load balancing mode available, but at the time of our testing it was not supported on zSeries.

For the install, we followed the StoneGate Installation Guide: Installing the Engine on the IBM zSeries Platform. We defined two VM guests called LITSGFW1 and LITSGFW2. We've listed the z/VM directory entries here to give you an idea of our network definitions and the size of our DASD partitions.

```
USER LITSGFW1 TEST4LIT 256M 512M G
   INCLUDE LINDFLT
   CPU 0
```

```
   CPU 1
   IPL 202
   MACHINE ESA 3
**  OSA Express with Layer 2 support
   DEDICATE 0600 0600
   DEDICATE 0601 0601
   DEDICATE 0602 0602
** HiperSocket for heartbeat
   DEDICATE E000 E000
   DEDICATE E001 E001
   DEDICATE E002 E002
** Guest LAN definition
   NICDEF 0B00 TYPE QDIO LAN SYSTEM PRVV74
   MDISK 0191 3390 420 100 VM3126 MR READPASS WRITPASS MULTPASS
   MDISK 0201 3390 1 1669 VM532E MR READPASS WRITPASS MULTPASS
   MDISK 0202 3390 1670 END VM532E MR READPASS WRITPASS MULTPASS

USER LITSGFW2 TEST4LIT 256M 512M G
   INCLUDE LINDFLT
   CPU 0
   CPU 1
   IPL 202
   MACHINE ESA 3
**  OSA Express with Layer 2 support
   DEDICATE 600 700
   DEDICATE 601 701
   DEDICATE 602 702
** HiperSocket for heartbeat
   DEDICATE E000 E004
   DEDICATE E001 E005
   DEDICATE E002 E006
** Guest LAN definition
   NICDEF 0B00 TYPE QDIO LAN SYSTEM PRVV74
   MDISK 0191 3390 540 100 VM3127 MR READPASS WRITPASS MULTPASS
   MDISK 0201 3390 1 1669 VM832E MR READPASS WRITPASS MULTPASS
   MDISK 0202 3390 1670 END VM832E MR READPASS WRITPASS MULTPASS
```

The instructions said to assign two read/write DASDs, we chose to define two minidisks on a single 3390 Model 3 ECKD device. To find the minimum partition sizes required, we selected the Expert Install mode option to see what it wanted. In the directory entries above, devices 0600-0602 and 0700-0702 are OSA Express with Layer 2 support used to access our Public LAN. Devices E000-E002 and E004-E006 are Hipersocket used for the Heartbeat between the engines. Device 0B00 is a Guest LAN used for our Internal/Private network.

We added the necessary RACF definitions for the guests. We downloaded all the StoneGate code to our local server. Note that StoneGate installs as an entirely self-contained package. It comes with its own, hardened instance of Linux included. As such, it does not install on top of an existing SUSE or RedHat Linux server. We FTP'd the StoneGate files SGINST.KERNEL, SGINST.INITRD, SGINST.PARMFILE and SGINST.EXEC to guest LITSGFW1.

The following are the prompts and our replies. From the LITSGFW1 console we issued:

```
SGINST EXEC

License agreement must be accepted before continuing.
Type YES to continue and NO to cancel. YES

Do you want to continue? YES

Enter the device number of second DASD: 201
Enter the device number of second DASD: 202
```

```
ldm_validate_partition_table(): Disk read failed.
0201(ECKD) at ( 94:  0) is dasda      : active at blocksize: 1024, 826155 blocks
, 806 MB
0202(ECKD) at ( 94:  4) is dasdb      : active at blocksize: 4096, 300420 blocks
, 1173 MB
Check that the DASDs are correct.
Type YES to continue and NO to cancel.
Do you want to continue? YES

Existing StoneGate installation has not been detected.

1. Full install
2. Full install in expert mode

Enter your choice:  1

Formatting volumes.

        root A: /dev/discs/disc0/part1
        root B: /dev/discs/disc0/part2
          swap: /dev/discs/disc0/part3
          data: /dev/discs/disc1/part1
         spool: /dev/discs/disc1/part2

Check that partitions have been assigned correctly.
Type YES to continue and NO to cancel.

Do you want to continue? YES
Creating filesystems...
Extracting StoneGate image...
Installing boot loader...
Executing zIPL (disc0/part1)...
The StoneGate can be started by IPLing the DASD number 0202
Installation finished! Please IPL the DASD to continue.

i 202 cl

Welcome to the StoneGate Engine Configuration Wizard.

This wizard will configure the StoneGate engine and
contact the management server. After this you can perform all
tasks using the administration client.

Step 1 of 3: Configure OS settings
--------------------------------------------------------------------------------
Current OS settings:
 - Host name not set
 - Sshd enabled: no
 - Root password is not set
 - Timezone not set
Are you happy with these settings (Y/N)? N

Host name: litsgfw1
Enable SSH daemon (Y/N)? Y

  === Change root password ===

Enter new root password: xxxxxxxx
Re-Enter it:xxxxxxxx

Select timezone (empty or prefix = list zones): EST
Selected: EST
Current OS settings:
 - Host name: 'litsgfw1'
 - Sshd enabled: yes
 - Root password has been changed
 - Timezone: EST
Are you happy with these settings (Y/N)? Y

Step 2 of 3: Configure network interfaces
--------------------------------------------------------------------------------
Current network interfaces:
```

```
Id Interf. Driver   Devno   MAC              Type  Status
------------------------------------------------------------------------------
------------------------------------------------------------------------------
R)reprobe channels, P)rint channels, A)dd driver, C)lear all,
M)gmt interface, N)IC id mapping, S)niff, rE)move NIC, O)ther options, D)one:
A

Detected following channels:

Type  In use Reg. Channel DevNo(s)
------------------------------------------------------------------------------
qeth (0x10) no no 0x0600,0x0601,0x0602
qeth (0x10) no no 0xe000,0xe001,0xe002
qeth (0x10) no no 0x0b00,0x0b01,0x0b02
Select the device layer:
iucv
qeth
Your selection:
qeth

  QETH for OSA-Express and Hipersockets channel device selected

Parameter syntax:
<devname>,<read_devno>,<write_devno>,data_devno,memusage,port_no,chksum_recv
Please refer to IBM document 'Linux for zSeries and S/390: Device Drivers and
Installation Commands' for more comprehensive syntax.
Please also remember to add 'enable_takeover' paramater to IP takeover
mode cluster interfaces.
HiperSockets example:
qeth1,0x7c00,0x7c01,0x7c02
OSA-Express example:
qeth2,0x500,0x501,0x502;add_parms,0x10,0x500,0x502,portname:PORT123
Parameters (or P)rint channels):
qeth0,0x0600,0x0601,0x0602;add_parms,0x10,0x0600,0x0602,enable_takeover
You have given the following configuration:
- Module: qeth
- Parameters:
qeth0,0x0600,0x0601,0x0602;add_parms,0x10,0x0600,0x0602,enable_takeover

Is it correct (Y/N)? Y

Current network interfaces:
Id Interf. Driver   Devno   MAC              Type  Status
------------------------------------------------------------------------------
 0 eth0    qeth     0x0600 00:09:6b:1a:2b:15 ether no link (mgmt)
------------------------------------------------------------------------------
R)reprobe channels, P)rint channels, A)dd driver, C)lear all,
M)gmt interface, N)IC id mapping, S)niff, rE)move NIC, O)ther options, D)one:
A
Detected following channels:
Type  In use Reg. Channel DevNo(s)
------------------------------------------------------------------------------
qeth (0x10) no no 0x0600,0x0601,0x0602
qeth (0x10) no no 0xe000,0xe001,0xe002
qeth (0x10) no no 0x0b00,0x0b01,0x0b02

Select the device layer:
iucv
  qeth
Your selection:
qeth
  QETH for OSA-Express and Hipersockets channel device selected
Parameter syntax:
  <devname>,<read_devno>,<write_devno>,data_devno,memusage,port_no,chksum_recv
Please refer to IBM document 'Linux for zSeries and S/390: Device Drivers and
Installation Commands' for more comprehensive syntax.
Please also remember to add 'enable_takeover' paramater to IP takeover mode
cluster interfaces.

HiperSockets example:
  qeth1,0x7c00,0x7c01,0x7c02
OSA-Express example:
```

```
  qeth2,0x500,0x501,0x502;add_parms,0x10,0x500,0x502,portname:PORT123

Parameters (or P)rint channels):
qeth1,0xe000,0xe001,0xe002;add_parms,0x10,0xe000,0xe002

You have given the following configuration:
 - Module: qeth
 - Parameters: qeth1,0xe000,0xe001,0xe002;add_parms,0x10,0xe000,0xe002
Is it correct (Y/N)? Y
Current network interfaces:
Id Interf. Driver   Devno   MAC              Type  Status
--------------------------------------------------------------------------------
 0 eth0    qeth     0x0600 00:09:6b:1a:2b:15 ether no link (mgmt)
 1 hsi1    qeth     0xe000 No MAC / GuestLAN ether no link
--------------------------------------------------------------------------------
R)eprobe channels, P)rint channels, A)dd driver, C)lear all,
M)gmt interface, N)IC id mapping, S)niff, rE)move NIC, O)ther options, D)one:
A
Detected following channels:

Type  In use Reg. Channel DevNo(s)
--------------------------------------------------------------------------------
qeth (0x10) no no 0x0600,0x0601,0x0602
qeth (0x10) no no 0xe000,0xe001,0xe002
qeth (0x10) no no 0x0b00,0x0b01,0x0b02

Select the device layer:
  iucv
  qeth
Your selection:
qeth

  QETH for OSA-Express and Hipersockets channel device selected

Parameter syntax:
  <devname>,<read_devno>,<write_devno>,data_devno,memusage,port_no,chksum_recv
Please refer to IBM document 'Linux for zSeries and S/390: Device Drivers and
Installation Commands' for more comprehensive syntax.
Please also remember to add 'enable_takeover' paramater to IP takeover mode
cluster interfaces.

HiperSockets example:
  qeth1,0x7c00,0x7c01,0x7c02
OSA-Express example:
  qeth2,0x500,0x501,0x502;add_parms,0x10,0x500,0x502,portname:PORT123

You have given the following configuration:
Parameters (or P)rint channels):
qeth2,0x0b00,0x0b01,0x0b02;add_parms,0x10,0x0b00,0x0b02

 - Module: qeth
- Parameters: qeth2,0x0b00,0x0b01,0x0b02;add_parms,0x10,0x0b00,0x0b02

Is it correct (Y/N)? Y
Current network interfaces:
Id Interf. Driver   Devno   MAC              Type  Status
--------------------------------------------------------------------------------
 0 eth0    qeth     0x0600 00:09:6b:1a:2b:15 ether no link (mgmt)
 1 hsi1    qeth     0xe000 No MAC / GuestLAN ether no link
 2 eth2    qeth     0x0b00 No MAC / GuestLAN ether no link
--------------------------------------------------------------------------------
R)eprobe channels, P)rint channels, A)dd driver, C)lear all,
M)gmt interface, N)IC id mapping, S)niff, rE)move NIC, O)ther options, D)one:
M
Management interface NIC id: 2
Current network interfaces:
Id Interf. Driver   Devno   MAC              Type  Status
--------------------------------------------------------------------------------
 0 eth0    qeth     0x0600 00:09:6b:1a:2b:15 ether no link
 1 hsi1    qeth     0xe000 No MAC / GuestLAN ether no link
 2 eth2    qeth     0x0b00 No MAC / GuestLAN ether no link (mgmt)
--------------------------------------------------------------------------------
```

```
R)reprobe channels, P)rint channels, A)dd driver, C)lear all,
M)gmt interface, N)IC id mapping, S)niff, rE)move NIC, O)ther options, D)one:
D

Step 3 of 3: Prepare for management contact
------------------------------------------------------------------------------
Currrent management contact settings:
Switch to initial configuration: No
 - Node IP address not set
 - Node netmask not set
 - Gateway IP address not set
Perform initial contact: No
 - Management IP address not set
 - Management One-time password not set
 - Key fingerprint not set
Are you happy with these settings (Y/N)? N

Switch to initial configuration (Y/N)? Y

Enter data for switching to the initial configuration and/or
contacting the management server. Fields marked with * must be filled.
Firewall node:

  Node IP address:* 192.168.74.200
  Netmask:* 255.255.255.0
  Gateway to management: 192.168.74.114
  Use VLAN (Y/N)? N

Perform new initial contact (Y/N)? Y

Enter data required for contacting the management server.
Fields marked with * must be filled, others are optional.
  Management IP address: 192.168.71.26
  One-time password:* xxxxxxxx
```

Note: This one time password was generated when we saved the initial configuration in the Configure the Firewall Cluster step above. A password is generated for each engine defined.

```
  Key fingerprint: enter

Currrent management contact settings:
Switch to initial configuration: Yes
 - Node IP address: '192.168.74.200'
 - Node netmask: '255.255.255.0'
 - Gateway IP address not set
Perform initial contact: Yes
 - Management IP address: '192.168.71.26'
 - Management One-time password: 'xxxxxxxx'
 - Key fingerprint:
Are you happy with these settings (Y/N)? Y
Your choices:
---------------------------------
R) Reconfigure
P) Print current settings
S) Save and exit
C) Cancel
---------------------------------
Your selection: S

Restarting StoneGate...
warning: Couldn't get cp configuration
warning: Could not get hardware MAC for interface eth2 (1663)
Contacting management system...
Contact succeeded!
```

Defining the rules

For this step, we followed chapter 14, Access Rules of the StoneGate Administrator's Guide. Being this step is unique to each customer environment, we are just providing one example of a rule we added. From the Configuration window, we created a new firewall rule base based off the Default. We defined our HOSTS which included our clients and WebSEAL server. We added a rule that allowed the connection for http:// and https:// at port 80 and 443 respectively. These ports were opened to permit traffic between our clients and the WebSEAL server in order to access and execute our workload. Our WebSEAL server is configured with both standard TCP (HTTP) proxy and secure SSL (HTTPS) proxy junctions between our back-end load balancer and Web application server clusters. We verified the firewall rules by running a workload from our clients to the WebSEAL server over TCP ports 80 and 443. We also introduced TCP/UDP traffic utilizing other ports on the LAN. Only ports 80 and 443 traffic were allowed to pass through the firewall verifying the rule worked correctly. Please refer to Figure 76 to see the connections.

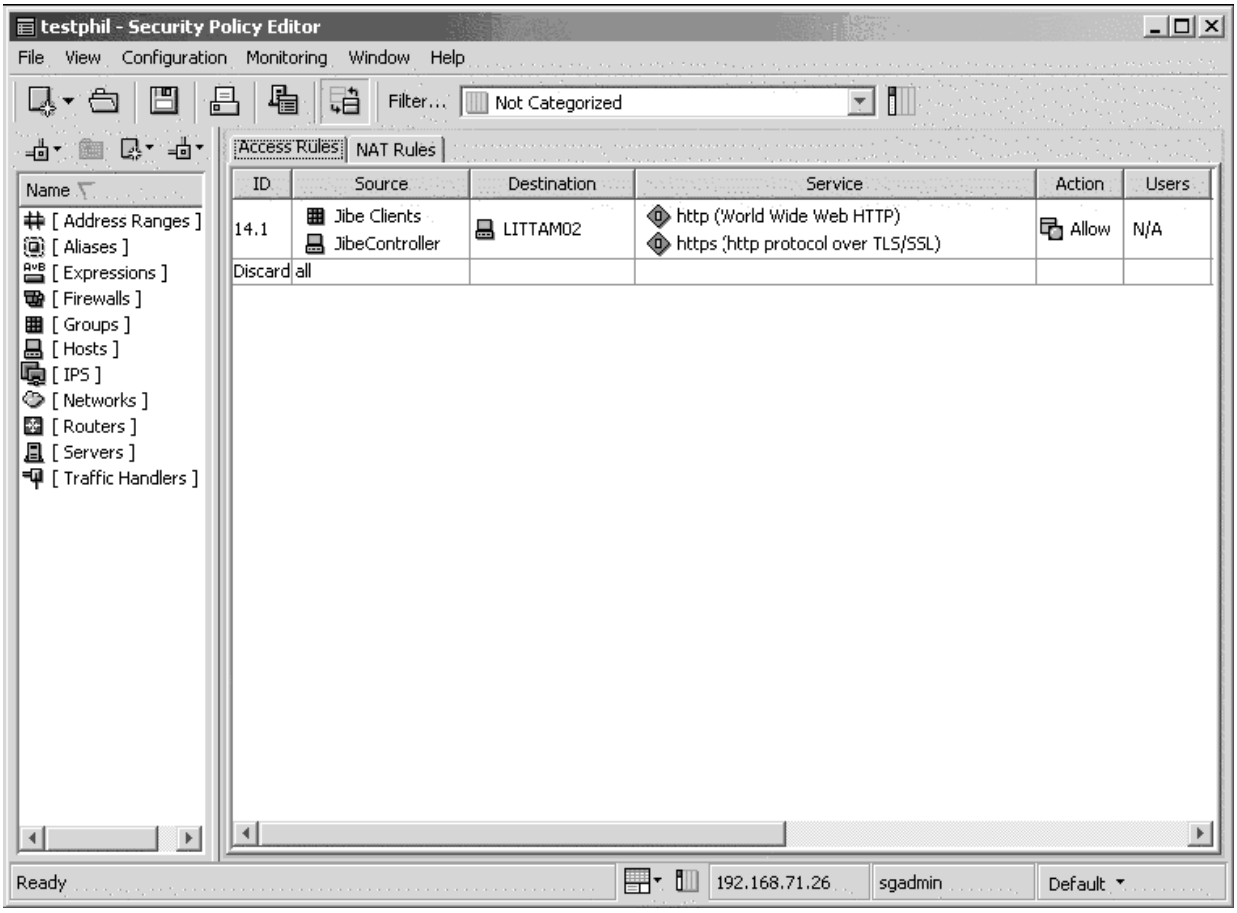


Figure 76. Defining the rules.

HA testing

On numerous occasions during our HA testing, we failed an engine by either issuing halt on the guest (clean take down) or by logging the guest off while active (yank the plug!). Each time the switch over to the standby node was successful with only a slight delay in activity. The nice thing here is that it's all built in to the StoneGate

package - all you do is provide the network connections. We also tried accessing different applications that we weren't permitted to and access was denied.

Summary of implementing Highly Available Stonesoft StoneGate Firewall

Stonesoft provides a lot of documentation that describes how you can use their product to build a very complex security environment. We've only touched on a small piece in this document. As stated previously, our intent was to add another layer of HA to our environment and hopefully pass on some useful information. Our future plans will be to implement some of the other product features like logging, alerts and reporting. We also plan to bring back the ethical hackers to give them another shot at penetrating our environment.

Implementing HA Reference Architectures: WebSphere with Highly Available Database

The following three architectures explore three different database products to be used in the same end to end highly available environment. For these architectures, the following components are the same:

- Highly available StoneGate firewall
- Highly available WebSphere Application Server Network Deployment Edge Component Load Balance
- Highly available HTTP server
- WebSphere Application Server cluster

Alternately, Linux Virtual Server technology or Tivoli Systems Automation for Multiplatforms can be used to create a highly available Web (or HTTP) server.

The key WebSphere application used for testing our HA environment was the Trade 6 benchmark workload. Trade 6 is a generally available application that includes a Web-based stock trading application which exploits Java database connectivity (JDBC™), includes session-based servlets and Enterprise JavaBeans (EJB). The Trade 6 application is installed on WebSphere Application Server, while the workload is generated by the WebSphere Studio Workload Simulator. The common flow between our clients to the backend databases in all three HA architectures is:

1. Requests come in through the workload simulator through the firewall
2. Requests would then be balanced by the Load Balancer and evenly sprayed to our Web servers by the Load Balancer Server
3. The selected Web server will then forward the request through round-robin to the WebSphere Cluster
4. From this point WebSphere would handle the request and use one of the three HA architectures described in further detail in the following sections. The setup involved in using one of these backend HA architectures in our WebSphere environment is usually minor. It is the responsibility of the JDBC driver (DB2 or Oracle) on the WebSphere Application Servers to communicate with the database server to determine the availability of the database server.

In the following sections, we will discuss our experiences with implementing the following three different HA database products to be used in the aforementioned WebSphere Application Server environment:

- DB2 High Availability Disaster Recovery (HADR) on Linux on System z
- Oracle Real Application Clusters

- DB2 z/OS Parallel Sysplex data sharing group

Implementing HA Reference Architecture: WebSphere with DB2 database on Linux

With DB2 High Availability Disaster Recovery (HADR) setup in our WebSphere environment, it provided quick failover in case a failure occurs on our primary DB2 server. There was no additional configuration required in WebSphere to take advantage of the DB2 HADR feature. It is the responsibility of the JDBC driver (on the clients) to determine the availability of the primary DB2 server. We configured HADR to use two DB2 servers and two databases, as shown in Figure 77, to mirror the data from the primary to the standby database. In our case, the data source defined for our Trade 6 workload in WebSphere pointed to the primary database. Here we spend time discussing the implementation of DB2 HADR on Linux and our test results.

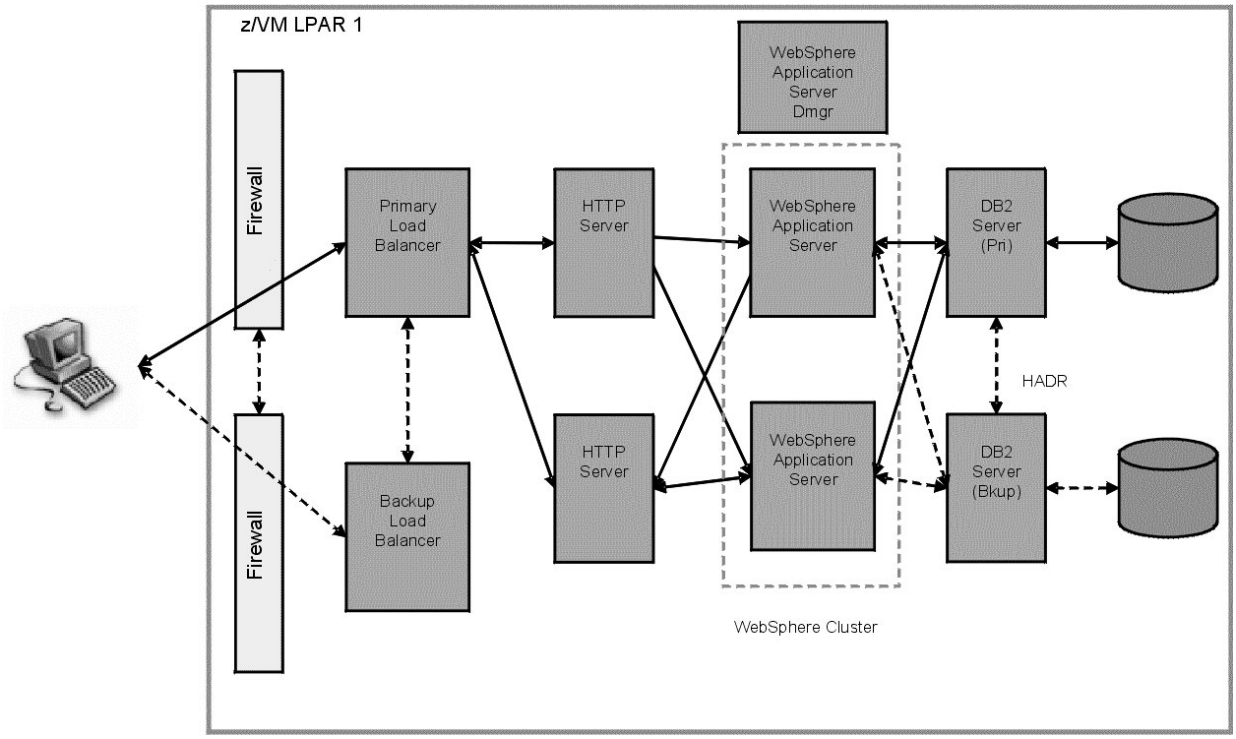


Figure 77. Configuring HADR to use two DB2 servers and two databases.

IBM Tivoli Systems Automation for Multiplatforms v2.1 Base Component, Linux

IBM Tivoli System Automation manages the availability of applications running in Linux systems or clusters on xSeries, zSeries, iSeries, pSeries, and AIX systems or clusters. It improves the availability and quality of critical business processes delivered to end users through self-managing and self-healing autonomic computing capabilities.

Tivoli System Automation for Multiplatforms delivers high availability to applications and middleware spanning any combination of Linux, AIX, or z/OS platforms, by introducing a new product structure with two orderable components:

1. The first is the base component that provides High Availability and Disaster Recovery capabilities for Linux, including Linux on zSeries and AIX server clusters.
2. The second is the new end-to-end automation management component that provides high availability for multi-tiered, or composite, business applications.

Our testing involved working with the base component of Tivoli Systems Automation for Multiplatforms (TSAM). We will describe the installation of IBM Tivoli Systems Automation for Multiplatforms (TSAM), configuration of automated IBM DB2 Universal Database (DB2 UDB) failover solution based on the high availability disaster recovery (HADR) feature, and summarize the problems/issues we found during the test.

Installing DB2 UDB

We had to install DB2 UDB on all the nodes we planned to use to host a DB2 instance. DB2 UDB was already installed and configured with an instance (db2inst1) on our two servers – litdat01.ltic.pok.ibm.com and litdat02.ltic.pok.ibm.com. For general installation instructions, please reference the *Quick Beginnings for DB2 Servers Manual*.

Installing TSAM

Tivoli Systems Automation for Multiplatforms v2.1 Base Component needs to be installed on both servers – litdat01 and litdat02. We obtained the TSAM v2.1 software from an internal code server, where it is packaged as a tar file. The tar archive name is C86P9ML.tar. This is the archive you use to install the product. Use the tar xvf command to extract the archive. When you have extracted the files, you will find the installation wizard in the following directory:

```
SAM2100Base/installSAM
```

Notes:

1. Set the following environment variable for all users of IBM Tivoli System Automation on all nodes:

   CT_MANAGEMENT_SCOPE=2 (peer domain scope). You can set the variable permanently if you set it in the profile.

2. Execute the installation wizard, agree to the license and the following will be displayed:

```
litdat01:/opt/TSAv210/SAM2100Base # ./installSAM

installSAM: A general License Agreement and License Information specifically for
System Automation will be shown. Scroll down using the Enter key (line by line)
or Space bar (page by page). At the end you will be asked to accept the terms to
be allowed to install the product. Select Enter to continue.
.......
y
......
installSAM: The following license is installed:
Product ID: 101
Creation date: Thu Jul  7 20:00:00 2005
Expiration date: Thu Dec 31 18:59:59 2037

installSAM: All packages were installed successfully.

installSAM: Status of System Automation after installation:
 ctrmc            rsct             11928   active
 IBM.ERRM         rsct_rm          12008   active
```

```
IBM.AuditRM      rsct_rm          12029   active
ctcas            rsct                     inoperative
IBM.SensorRM     rsct_rm                  inoperative
```

For more details on the installation procedure, follow the *IBM System Automation for Multiplatforms, Guide and Reference* found at:

`http://publib.boulder.ibm.com/tividd/td/ITSAFL/SC33-8210-05/en_US/PDF/halgre21.pdf`

Setting up TSAM to manage the DB2 instance

This entire section is based on the whitepaper “Automating DB2 HADR Failover on Linux using Tivoli System Automation” by Steve Raspudic. The paper describes the combination of DB2 UDB and TSAM to provide a highly available computing environment using DB2 HADR.

DB2 HADR is a data replication feature that provides an HA solution for both partial and complete site failures. DB2 HADR is designed for quick failover, easy setup, and manageability. HADR protects against data loss by replicating data changes from a source database, called the primary, to a target database, called the standby.

There are two approaches to providing a highly available communications to a DB2 HADR pair:

1. Define an IP address that will be collocated with the location of the HADR Primary database, or
2. Use the DB2 Client Reroute functionality

We used the DB2 Client Reroute approach. If it detected a communications error, it would transparently failover the connection state to the defined alternate server.

Before creating the cluster, the following configuration changes are highly recommended if you are running IBM Tivoli System Automation for Multiplatforms 2.

1. **Create netmon.cf file:**

   In case you run a one or two node cluster you need some additional configuration to detect network interface failures. The cluster software periodically tries to reach each network interface of the cluster. If there is a two node cluster and one interface fails on one node, the other interface on the other node is not able to get response from the peer and will also be flagged offline.

   To avoid this behavior the cluster software must be told to contact a network instance outside the cluster. Best practice is to use the default gateway of the subnet the interface is in.

   On each node create following file:

   `/usr/sbin/cluster/netmon.cf`

   Each line of this file should contain the machine name or IP address of the external instance. An IP address should be specified in dotted decimal format. If the machine is connected to more then one IP sub net using different network interfaces, then an entry for each IP sub net is required in the netmon.cf file.

   The Base Component User Guide recommends creating the netmon.cf file at minimum to monitor and detect failure of the network interfaces. If the interface on a node is not able to get a response, it will be flagged offline. With the netmon.cf file in place, the automation can assign the service IP to another node. The role of the network tie breaker(in which we setup below) decides which node is able to go on with automation once a break down in cluster communication has occurred.

We created and added the following to /usr/sbin/cluster/netmon.cf for litdat01 and litdat02: (Note that our images are on a single IP subnet and have a default router of 71.97)

```
litdat01:/usr/sbin/cluster # cat netmon.cf
# default gateway for all interfaces in 192.168.71.0 network
192.168.71.97

litdat02:/usr/sbin/cluster # cat netmon.cf
# default gateway for all interfaces in 192.168.71.0 network
192.168.71.97
```

2. **Turn off broadcast for all communication groups:**

   The RSCT heartbeat mechanism performs a broadcast ping from time to time. This is especially often the case in situations where a network interface adapter is not available. The reason for this feature is to find out whether the network interface adapter that sends this broadcast ping is still operational (this can be determined based upon whether other machines reply to this broadcast ping or not). This feature is not needed if the netmon.cf file is setup correctly as described above, since in that case there are other well-known network interface adapters to be checked for availability.

   While a broadcast ping on a stand-alone machine is not a performance issue, it will have a negative impact on the performance if the machines are running in a zVM environment. This is because all other systems running under this zVM and within the same network segment (same IP network and net mask) will reply to this broadcast ping request. As a result, even zVM guest systems that are idle and currently paged out will be loaded into the zVM just to reply to this ping. Depending on the number of guest systems running under this zVM this may decrease the performance of the whole z/VM system.

   In order to prevent this situation from happening, the following setup changes are highly recommended:

   - Get the communication group information of the DB2 cluster:

     ```
     # lscomg

     litdat01:/usr/sbin/cluster # lscomg
     Name Sensitivity Period Priority Broadcast SourceRouting NIMPathName NIMParameters
     CG1  4           1      1        Yes       Yes

     litdat02:/usr/sbin/cluster # lscomg
     Name Sensitivity Period Priority Broadcast SourceRouting NIMPathName NIMParameters
     CG1  4           1      1        Yes       Yes
     ```

   - Turn off broadcast for all communication groups:

     ```
     # chcomg -x b <communication group>

     litdat01:/usr/sbin/cluster # chcomg -x b CG1
     litdat02:/usr/sbin/cluster # chcomg -x b CG1
     ```

   - Verify that broadcast is turned off using the *lscomg* command, after the above changes are done.

     ```
     litdat01:/usr/sbin/cluster # lscomg
     Name Sensitivity Period Priority Broadcast SourceRouting NIMPathName NIMParameters
     CG1  4           1      1        No        Yes

     litdat02:/usr/sbin/cluster # lscomg
     Name Sensitivity Period Priority Broadcast SourceRouting NIMPathName NIMParameters
     CG1  4           1      1        No        Yes
     ```

Creating the cluster

On each node we needed to enable communication with the other cluster nodes by issuing:

```
$> preprpnode <node01> <node02> ... <node0N>
```

Now that the nodes were prepped for communication, we created a cluster with the name ″litdat_domain″ on our nodes. You can name your cluster anything you like as long as you only use characters {A-Z, a\u2013z, 0-9, ., _}. The command to create a cluster was issued from only one node: $> mkrpdomain litdat_domain <node01> <node02> ... <node0N>

We checked our work so far by examining the status of our ″litdat_domain″:

```
$> lsrpdomain
```

We observed output that indicated our cluster was now defined but in an offline state. Thus it was time to bring the cluster online by issuing:

```
$> startrpdomain litdat_domain
```

At this point issuing the lsrpdomain command a few times showed that the cluster was in the process of starting up (OpState is Pending Online) for a few moments. Approximately a minute later the status of the cluster indicated it was started.

```
$> ./lsrpdomain
Name          OpState RSCTActiveVersion MixedVersions TSPort GSPort
litdat_domain Online  2.4.3.1           No            12347  12348
```

We also checked to see if all the nodes are online by issuing:

```
$> # ./lsrpnode
Name     OpState RSCTVersion
litdat01 Online  2.4.3.1
litdat02 Online  2.4.3.1
```

For any even number cluster (such as our two-node cluster), a tiebreaker disk is required to automate resource group takeover. A tie-breaker resource (an instance of the IBM.TieBreaker resource class) is used to determine which sub-domain has operational quorum. A ″tie″ situation also occurs when exactly half the nodes of a domain are online, and the other half are inaccessible. You can have a number of IBM.TieBreaker resources defined, but only one can be active at any one time.

The whitepaper uses a disk tiebreaker, but we will be using a network tie breaker.

Setting up the network tie breaker

The network tie breaker uses an external IP (network instance) to resolve a tie situation.

There may be several reasons to use a network tie breaker, for example:

- There is no possibility to use a disk based tie breaker.
- It is the highest priority of a high availability environment to communicate with instances outside the cluster.

For establishing a tie breaker, we did the following:

- Created the network tie breaker:

  ```
  $> ./mkrsrc IBM.TieBreaker Type="EXEC" Name="tb" DeviceInfo='PATHNAME=
  /usr/sbin/rsct/bin/samtb_net Address=192.168.71.97 Log=1' PostReserveWaitTime=30;
  ```

- Activated the network tie breaker:

  ```
  $> ./chrsrc -c IBM.PeerNode OpQuorumTieBreaker="tb"
  ```

Setting up DB2 ID’s and instances

The next step was to ensure the appropriate user and group ID’s were created on each node for the cluster. Our instances were named db2instp for primary UDB and db2insts for standby UDB.

We created the groups on each node in a local server authentication environment:

```
litdat01:/usr/sbin/rsct/bin # groupadd -g 999 db2iadm1
litdat01:/usr/sbin/rsct/bin # groupadd -g 998 db2fadm1
litdat01:/usr/sbin/rsct/bin # groupadd -g 997 db2asgrp

litdat02:/usr/sbin/rsct/bin # groupadd -g 999 db2iadm1
litdat02:/usr/sbin/rsct/bin # groupadd -g 998 db2fadm1
litdat02:/usr/sbin/rsct/bin # groupadd -g 997 db2asgrp
```

We then created the user ID's for primary and standby servers respectively:

```
litdat01:/usr/sbin/rsct/bin # useradd -g db2iadm1 -u 1005 -d /misc/homep/db2instp -m db2instp
litdat01:/usr/sbin/rsct/bin # useradd -g db2iadm1 -u 1004 -d /misc/homes/db2insts -m db2insts
litdat01:/usr/sbin/rsct/bin # useradd -g db2fadm1 -u 1003 -d /misc/home/db2fenc1 -m db2fnc1
litdat01:/usr/sbin/rsct/bin # useradd -g db2asgrp -u 1002 -d /misc/home/db2as -m db2as

litdat02:/usr/sbin/rsct/bin # useradd -g db2iadm1 -u 1005 -d /misc/homep/db2instp -m db2instp
litdat02:/usr/sbin/rsct/bin # useradd -g db2iadm1 -u 1004 -d /misc/homes/db2insts -m db2insts
litdat02:/usr/sbin/rsct/bin # useradd -g db2fadm1 -u 1003 -d /misc/home/db2fenc1 -m db2fnc1
litdat02:/usr/sbin/rsct/bin # useradd -g db2asgrp -u 1002 -d /misc/home/db2as -m db2as
```

We created two DB2 instances, named db2instp and db2insts, the first instance on the primary server and the other on the standy server:

Note: The Whitepaper uses '-u db2tsa' for their fence id, we created and used one called db2fnc1

```
litdat01:/opt/IBM/db2/V8.1/instance # ./db2icrt -w 64 -u db2fnc1 db2instp
DBI1070I Program db2icrt completed successfully.

litdat02:/opt/IBM/db2/V8.1/instance # ./db2icrt -w 64 -u db2fnc1 db2insts
DBI1070I Program db2icrt completed successfully.
```

We created an equivalency resource:

```
litdat01:/ # mkequ virpubnic_litdat01 IBM.NetworkInterface:eth0:litdat01
litdat01:/ # mkequ virpubnic_litdat02 IBM.NetworkInterface:eth0:litdat02
```

We created an instance HA:

Note: The HA scripts (located in /opt/IBM/db2/V8.1/ha) are to be included with DB2 UDB v8.1 Fixpak 7(also known as DB2 UDB v8.2) and above. The scripts are missing in the 64bit stream, an APAR LI70832 has already been created to handle this problem, and the scripts will be included with DB2 v8.1 Fixpak 11 or later.

```
litdat01:/opt/IBM/db2/V8.1/ha/salinux # ./regdb2salin -a db2instp -r -i 192.168.71.146
Stopping instance db2instp...

About to register db2instp with TSA

Checking cluster validity...
Checking configuration ...

Making resource group  db2_db2instp_0-rg  ...
Making IP resource  db2_db2instp_0-rs_ip ...
Making DB2 resource  db2_db2instp_0-rs  ...

Online resource group db2_db2instp_0-rg ...

db2_db2instp_0-rg is now online

Done, instance is now HA
```

Setting up DB2 HADR

We turned the resource group offline:

```
litdat01:/opt/IBM/db2/V8.1/ha/salinux # chrg -o Offline db2_db2instp_0-rg
```

We created a dependency between the HA IP resource and the network equivalency group:

```
litdat01:/opt/IBM/db2/V8.1/ha/salinux # mkrel -p DependsOn -S
IBM.ServiceIP:db2_db2instp_0-rs_ip -G IBM.Equivalency:virpubnic_litdat01
db2_db2instp_0-rs_ip_equiv_do
```

Next, we performed the same steps for the other instance node – db2insts, on litdat02:

```
litdat02:/opt/IBM/db2/V8.1/ha/salinux # ./regdb2salin -a db2insts -r -i 192.168.71.147
Stopping instance db2insts...

About to register db2insts with TSA

Checking cluster validity...
Checking configuration ...

Making resource group  db2_db2insts_0-rg  ...
Making IP resource  db2_db2insts_0-rs_ip ...
Making DB2 resource  db2_db2insts_0-rs  ...

Online resource group db2_db2insts_0-rg ...

db2_db2insts_0-rg is now online

Done, instance is now HA

litdat02:/opt/IBM/db2/V8.1/ha/salinux # chrg -o Offline db2_db2insts_0-rg

litdat02:/opt/IBM/db2/V8.1/ha/salinux # mkrel -p DependsOn -S
IBM.ServiceIP:db2_db2insts_0-rs_ip -G IBM.Equivalency:virpubnic_litdat02
db2_db2insts_0-rs_ip_equiv_do
```

We turned the resource group online:

```
litdat01:/opt/IBM/db2/V8.1/ha/salinux # chrg -o Online db2_db2instp_0-rg
litdat02:/opt/IBM/db2/V8.1/ha/salinux # chrg -o Online db2_db2insts_0-rg
```

We created a DB2 HADR Database. We first logged on to litdat01 using db2instp instance, and issued the following command to create the database:

```
db2instp@litdat01:~> db2 create database hadrdb
DB20000I  The CREATE DATABASE command completed successfully.
```

We turned on LOGRETAIN for this db and took a database backup image as follows:

```
db2instp@litdat01:~> db2 update db cfg for hadrdb using LOGRETAIN ON
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.

db2instp@litdat01:~> db2 backup database hadrdb

Backup successful. The timestamp for this backup image is : 20051213101448
```

We transferred the backup database to litdat02, and restored it under the db2insts instance as follows:

```
db2instp@litdat01:~> scp HADRDB.0.db2instp.NODE0000.CATN0000.20051213101448.001 db2insts@litdat02:/misc/homes/db2insts
Password:
HADRDB.0.db2instp.NODE0000.CATN0000.200512131 100%   35MB   3.8MB/s   00:09

db2insts@litdat02:~> db2 restore database hadrdb
DB20000I  The RESTORE DATABASE command completed successfully.
```

We ensured that TCP/IP listener is started and listening on a well-known port for db2instp instance on litdat01 and db2insts instance on litdat02, issued the following:

```
db2instp@litdat01:~> db2set DB2COMM=tcpip
db2instp@litdat01:~> db2 update dbm cfg using SVCENAME db2instp

db2insts@litdat02:~> db2set DB2COMM=tcpip
db2insts@litdat02:~> db2 update dbm cfg using SVCENAME DB2_db2insts
```

We updated the database level configuration parameters required for an HADR pair to be established. View the file /etc/services to determine which ports are free and can be used. We used port 60014 for our HADR port number.

We issued the following commands from db2instp on litdat01:

```
db2instp@litdat01:~> db2 update db cfg for hadrdb using HADR_LOCAL_HOST litdat01
db2instp@litdat01:~> db2 update db cfg for hadrdb using HADR_REMOTE_HOST litdat02
db2instp@litdat01:~> db2 update db cfg for hadrdb using HADR_LOCAL_SVC 60014
db2instp@litdat01:~> db2 update db cfg for hadrdb using HADR_REMOTE_SVC 60014
db2instp@litdat01:~> db2 update db cfg for hadrdb using HADR_REMOTE_INST db2insts
```

We issued the following commands from db2insts on litdat02:

```
db2insts@litdat02:~> db2 update db cfg for hadrdb using HADR_LOCAL_HOST litdat02
db2insts@litdat02:~> db2 update db cfg for hadrdb using HADR_REMOTE_HOST litdat01
db2insts@litdat02:~> db2 update db cfg for hadrdb using HADR_LOCAL_SVC 60014
db2insts@litdat02:~> db2 update db cfg for hadrdb using HADR_REMOTE_SVC 60014
db2insts@litdat02:~> db2 update db cfg for hadrdb using HADR_REMOTE_INST db2instp
```

We updated the alternate server for each instance. This is required in order to support the Client Reroute functionality in DB2 UDB v8.2.

We issued the following commands from db2instp on litdat01:

```
db2instp@litdat01:~> db2 update alternate server for database hadrdb using hostname 192.168.71.147 port 60004
```

We issued the following commands from db2insts on litdat02:

```
db2insts@litdat02:~> db2 update alternate server for database hadrdb using hostname 192.168.71.146 port 60004
```

It is also recommended that the database configuration parameter LOGINDEXBUILD is set to *ON* before HADR is started.

We issued the following commands from db2instp on litdat01:

```
db2instp@litdat01:~> db2 update db cfg for hadrdb using logindexbuild on
```

We issued the following commands from db2insts on litdat02:

```
db2insts@litdat02:~> db2 update db cfg for hadrdb using logindexbuild on
```

We restarted the resource groups on both systems to be sure all changes take effect:

```
litdat01:/opt/IBM/db2/V8.1/ha/salinux # chrg -o Offline db2_db2instp_0-rg
litdat02:/opt/IBM/db2/V8.1/ha/salinux # chrg -o Offline db2_db2insts_0-rg
litdat01:/opt/IBM/db2/V8.1/ha/salinux # chrg -o Online db2_db2instp_0-rg
litdat02:/opt/IBM/db2/V8.1/ha/salinux # chrg -o Online db2_db2insts_0-rg
```

We started HADR on standby and primary servers:

Note: Standby must be started before Primary

We started HADR as standby on db2insts on litdat02:

```
db2insts@litdat02:~/sqllib/db2dump> db2 start hadr on db hadrdb as standby
DB20000I  The START HADR ON DATABASE command completed successfully.
```

We started HADR as primary on db2instp on litdat01:

```
db2instp@litdat01:~> db2 start hadr on db hadrdb as primary
DB20000I  The START HADR ON DATABASE command completed successfully.
```

We then verified if the HADR pair was in peer state with the instance db2instp hosting the primary database hadrdb on the physical host litdat01 and the instance

db2insts hosting the standby database hadrdb on the physical host litdat02. We used the DB2 snapshot command to do this:

```
db2instp@litdat01:~> db2 get snapshot for all on hadrdb

HADR Status
  Role                   = Primary
  State                  = Peer
  Synchronization mode   = Nearsync
  Connection status      = Connected, 12/14/2005 00:34:32.860961
  Heartbeats missed      = 0
  Local host             = litdat01
  Local service          = 60014
  Remote host            = litdat02
  Remote service         = 60014
  Remote instance        = db2insts
  timeout(seconds)       = 120
  Primary log position(file, page, LSN) = S0000000.LOG, 0, 0000000000BB8000
  Standby log position(file, page, LSN) = S0000000.LOG, 0, 0000000000BB8000
  Log gap running average(bytes) = 0
```

The output is large, but you should see something similar to the above example.

We created the HADR Resource Group controlling the HADR state. We registered the resource group at the node currently hosting the primary on litdat01.

We issued the following commands from /opt/IBM/db2/V8.1/ha/salinux:

```
litdat01:/opt/IBM/db2/V8.1/ha/salinux # ./reghadrsalin -a db2instp -b db2insts -d hadrdb

About to register db2hadr_hadrdb with TSA ...
Creating HADR resource group ...
Making resource group  db2hadr_hadrdb-rg  ...
Making resource  db2hadr_hadrdb-rs  ...
Online resource group db2hadr_hadrdb-rg ...
db2hadr_hadrdb-rg is now online

Done, HADR is now registered with the TSA framework
```

The HADR pair state was now controlled by the cluster manager. We viewed the resource group and resources using getstatus:

```
litdat01:/opt/IBM/db2/V8.1/ha/salinux # ./getstatus

-- Resource Groups and Resources --

                      Group Name                Resources
                      ----------                ---------
               db2_db2instp_0-rg      db2_db2instp_0-rs_ip
               db2_db2instp_0-rg         db2_db2instp_0-rs
                                 -                         -
               db2_db2insts_0-rg      db2_db2insts_0-rs_ip
               db2_db2insts_0-rg         db2_db2insts_0-rs
                                 -                         -
               db2hadr_hadrdb-rg         db2hadr_hadrdb-rs
                                 -                         -


-- Resources --

                    Resource Name                Node Name             State
                    -------------                ---------             -----
            db2_db2instp_0-rs_ip                  litdat01            Online
                                 -                         -                 -
               db2_db2instp_0-rs                  litdat01            Online
                                 -                         -                 -
```

```
                db2_db2insts_0-rs_ip                litdat02            Online
                                   -                    -                 -
                   db2_db2insts_0-rs                litdat02            Online
                                   -                    -                 -
                   db2hadr_hadrdb-rs                litdat01            Online
                   db2hadr_hadrdb-rs                litdat02           Offline
                                   -                    -                 -
```

The output should be similar to the above, taken from our primary server - litdat01.

Now, if the physical host litdat01 fails, the resource group db2hadr_hadr-rg will fail over to be hosted by physical host litdat02, which will cause the HADR primary database to be hosted now on litdat02.

Note: From our experience, the getstatus command did not always report back the correct status detail. The release notes say it's probably necessary to do a reset if we ever get 'Failed Offline' also. If this happens, simply run the following commands:

```
resetrsrc -s "Name like '%' " IBM.ServiceIP
resetrsrc -s "Name like '%'" IBM.Application
```

We cataloged the DB2 instances. On each server, ensured it was cataloged with the node directory information of the other node:

```
db2instp@litdat01:~> db2 catalog tcpip node hadrdb remote 192.168.71.146 server 60004

db2insts@litdat02:~> db2 catalog tcpip node hadrdb remote 192.168.71.147 server 60004
```

Both instances were now protected by IBM Tivoli SAM, as was the HADR database.

Testing and verifying DB2 HADR

To test and verify DB2 HADR we did the following:

- "Controlled Failover"
- "Complete Failover" on page 292

Controlled Failover: We changed the location of the node hosting the HADR primary database using the following command on litdat01, the primary UDB:

```
$ > rgreq -o Move -n litdat01 db2hadr_hadrdb-rg
```

We issued the getstatus command and see if the HADR primary database is now hosted by litdat02. You should see something similar to the following example:

```
litdat01:/opt/IBM/db2/V8.1/ha/salinux # ./getstatus

-- Resource Groups and Resources --

                      Group Name                Resources
                      ----------                ---------
                db2_db2instp_0-rg      db2_db2instp_0-rs_ip
                db2_db2instp_0-rg         db2_db2instp_0-rs
                                -                         -
                db2_db2insts_0-rg      db2_db2insts_0-rs_ip
                db2_db2insts_0-rg         db2_db2insts_0-rs
                                -                         -
                db2hadr_hadrdb-rg         db2hadr_hadrdb-rs
                                -                         -


-- Resources --

                    Resource Name                Node Name              State
```

```
                -------------                  ---------              -----
        db2_db2instp_0-rs_ip                   litdat01             Online
                           -                         -                  -
           db2_db2instp_0-rs                   litdat01             Online
                           -                         -                  -
        db2_db2insts_0-rs_ip                   litdat02             Online
                           -                         -                  -
           db2_db2insts_0-rs                   litdat02             Online
                           -                         -                  -
           db2hadr_hadrdb-rs                   litdat01            Offline
           db2hadr_hadrdb-rs                   litdat02             Online
```

We issued the following command to bring the HADR primary database back to being hosted by litdat01:

```
$ > rgreq -o Move -n litdat02 db2hadr_hadrdb-rg
```

Complete Failover: The next set of tests verifies our HADR configuration which requires additional tuning. We had to set the following before starting such tests:

1. We updated the DB2 DBHEAP to 100000 for hadrdb database:

   ```
   db2instp@litdat01:~/sqllib/db2dump> db2 update db cfg for hadrdb using DBHEAP 100000
   ```

   Otherwise, you may see the following error in the ~db2instp/sqllib/db2dump.log:

   ```
   2006-01-12-10.48.52.431984-300 I16331A452        LEVEL: Severe
   PID     : 8323                 TID  : 2199083223840PROC : db2hadrs (HADRDB) 0
   INSTANCE: db2insts             NODE : 000         DB   : HADRDB
   FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrInitBuf, probe:30110
   RETCODE : ZRC=0x8B0F0002=-1961951230=SQLO_NOMEM_DBH
             "No memory available in 'Database Heap'"
             DIA8302C No memory available in the database heap.
   ```

2. We updated the DB2 LOCKLIST and MAXLOCKS for hadrdb database:

   ```
   db2instp@litdat01:~> db2 update db cfg for hadrdb using locklist 1000
   db2instp@litdat01:~> db2 update db cfg for hadrdb using maxlocks 100
   ```

   Otherwise, you may see the following exceptions and errors in WebSphere's SystemOut.log when running the Trade workload:

   ```
   [1/27/06 16:57:09:135 EST] 00000015 TimeoutManage I   WTRN0006W:
   Transaction 000001090DDCD2BE00000001000091A3F1121D27DBECE10FA191E42B1611098D122C157F000001
   090DDCD2BE00000001000091A3F1121D27DBECE10FA191E42B1611098D122C157F00000001
   has timed out after 120 seconds.
   ...
          at com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java:1455)
   Caused by: com.ibm.db2.jcc.a.SqlException: DB2 SQL error: SQLCODE: -911,
   SQLSTATE: 40001, SQLERRMC: 2
   ```

3. We updated the TCP/IP timeout values at the Linux kernel level. To make the changes persistent across reboots, we added the changes to /etc/sysctl.conf. The updates must be made on the client side, in our case, the WebSphere Application Server nodes:

   ```
   litwas01:~ # cat /etc/sysctl.conf
   net.ipv4.tcp_keepalive_intvl = 2
   net.ipv4.tcp_keepalive_probes = 3
   net.ipv4.tcp_keepalive_time=30
   net.ipv4.tcp_fin_timeout=30
   net.ipv4.tcp_retries2 = 1
   net.ipv4.tcp_retries1 = 1
   net.ipv4.tcp_syn_retries = 1
   ```

Testing complete DB2 UDB Server failure while Trade 6 workload is running: While requests and transactions were being made between our WebSphere nodes and the DB2 UDB Server using JDBC Type 4, we logged off the litdat01 guest on z/VM to simulate a complete power outage to the machine.

1. We started the JIBE workload to begin sending requests to our WebSphere Cluster.
2. After about 1-2 minutes of successful transactions, we logged off the DB2 UDB server guest – litdat01
3. We viewed the transaction rate fall to 0 on the JIBE controller. At the same time, we were issuing the DB2 snapshot command from the db2insts – standby node – litdat02 to determine the HADR status:

```
HADR Status
  Role                   = Standby
  State                  = Disconnected
  Synchronization mode   = Nearsync
  Connection status      = Disconnected, 01/16/2006 16:57:01.059668
  Heartbeats missed      = 0
  Local host             = litdat02
  Local service          = 60014
  Remote host            = litdat01
  Remote service         = 60014
  Remote instance        = db2instp
  timeout(seconds)       = 120
  Primary log position(file, page, LSN) = S0000014.LOG, 735, 000000000454782B
  Standby log position(file, page, LSN) = S0000000.LOG, 0, 0000000000000000
  Log gap running average(bytes) = 0
```

4. After about 1 minute, the DB2 snapshot showed that db2insts had become the primary:

```
HADR Status
  Role                   = Primary
  State                  = Disconnected
  Synchronization mode   = Nearsync
  Connection status      = Disconnected, 01/16/2006 16:57:01.059668
  Heartbeats missed      = 0
  Local host             = litdat02
  Local service          = 60014
  Remote host            = litdat01
  Remote service         = 60014
  Remote instance        = db2instp
  timeout(seconds)       = 120
  Primary log position(file, page, LSN) = S0000014.LOG, 735, 000000000454782B
  Standby log position(file, page, LSN) = S0000000.LOG, 0, 0000000000000000
  Log gap running average(bytes) = 0
```

At which time, the transactions began to pick up and start running again. The WebSphere SystemOut.log had the following messages that showed connections were re-established:

```
---- Begin backtrace for Nested Throwables
        at com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java(Compiled Code))
Caused by: javax.ejb.EJBException: TradeBean.getClosedOrders - error; nested
exception is: javax.ejb.TransactionRolledbackLocalException:  ; nested
exception is: javax.ejb.TransactionRolledbackLocalException:  ; nested
exception is: com.ibm.websphere.ce.cm.StaleConnectionException: A connection
failed but has been re-established. The hostname or IP address is
"192.168.71.147" and the service name or port number is 60004 . Special
registers may or may not be re-attempted (Reason code = 1
DB2ConnectionCorrelator: C0A84766.B46C.060222203605DSRA0010E: SQL State =
08506, Error Code = -4,498

java.sql.SQLException: A connection failed but has been re-established. The
hostname or IP address is "192.168.71.147" and the service name or port
number is 60004 . Special registers may or may not be re-attempted (Reason
code = 1  DB2ConnectionCorrelator: C0A84766.B46C.060222203605DSRA0010E: SQL
State = 08506, Error Code = -4,498
```

5. Now that we knew all the work was being directed successfully to the standby node. We started the primary guest back up – litdat01. Once started, to re-establish the HADR pair with litdat01, we issued the following commands:

```
litdat01:/opt/IBM/db2/V8.1/ha/salinux # chrg -o Offline db2_db2instp_0-rg
litdat01:/opt/IBM/db2/V8.1/ha/salinux # su – db2instp
litdat01:/opt/IBM/db2/V8.1/ha/salinux # db2 start hadr on db hadrdb as standby
```

6. The HADR pair was now re-established. We verified using the getstatus command:

```
litdat01:/opt/IBM/db2/V8.1/ha/salinux # ./getstatus

-- Resource Groups and Resources --

                      Group Name                Resources
                      ----------                ---------
                 db2_db2instp_0-rg      db2_db2instp_0-rs_ip
                 db2_db2instp_0-rg         db2_db2instp_0-rs
                                 -                         -
                 db2_db2insts_0-rg      db2_db2insts_0-rs_ip
                 db2_db2insts_0-rg         db2_db2insts_0-rs
                                 -                         -
                 db2hadr_hadrdb-rg         db2hadr_hadrdb-rs
                                 -                         -


-- Resources --

                       Resource Name                Node Name             State
                       -------------                ---------             -----
                db2_db2instp_0-rs_ip                 litdat01            Online
                                   -                        -                 -
                   db2_db2instp_0-rs                 litdat01            Online
                                   -                        -                 -
                db2_db2insts_0-rs_ip                 litdat02            Online
                                   -                        -                 -
                   db2_db2insts_0-rs                 litdat02            Online
                                   -                        -                 -
                   db2hadr_hadrdb-rs                 litdat01           Offline
                   db2hadr_hadrdb-rs                 litdat02            Online
```

7. We moved the HADR resource group to node litdat01 to bring the primary back to being hosted by the instance db2instp, using the following command:

```
$ > rgreq -o Move -n litdat02 db2hadr_hadrdb-rg
```

We noticed the transaction rate fall to 0 on the JIBE controller. Since this second time was a controlled failover, the time of failover from litdat02 to litdat01 only took 10 – 15 seconds.

Implementing HA Reference Architecture: WebSphere with Oracle RAC database on Linux 62

In this architecture, Oracle Real Application Cluster (RAC), as shown in Figure 78 on page 295, was used as the highly available backend database in our WebSphere Application Server environment. The configuration required on the WebSphere Application Servers is minimal. So we spend most of our time here discussing the implementation Oracle RAC and our testing experiences.

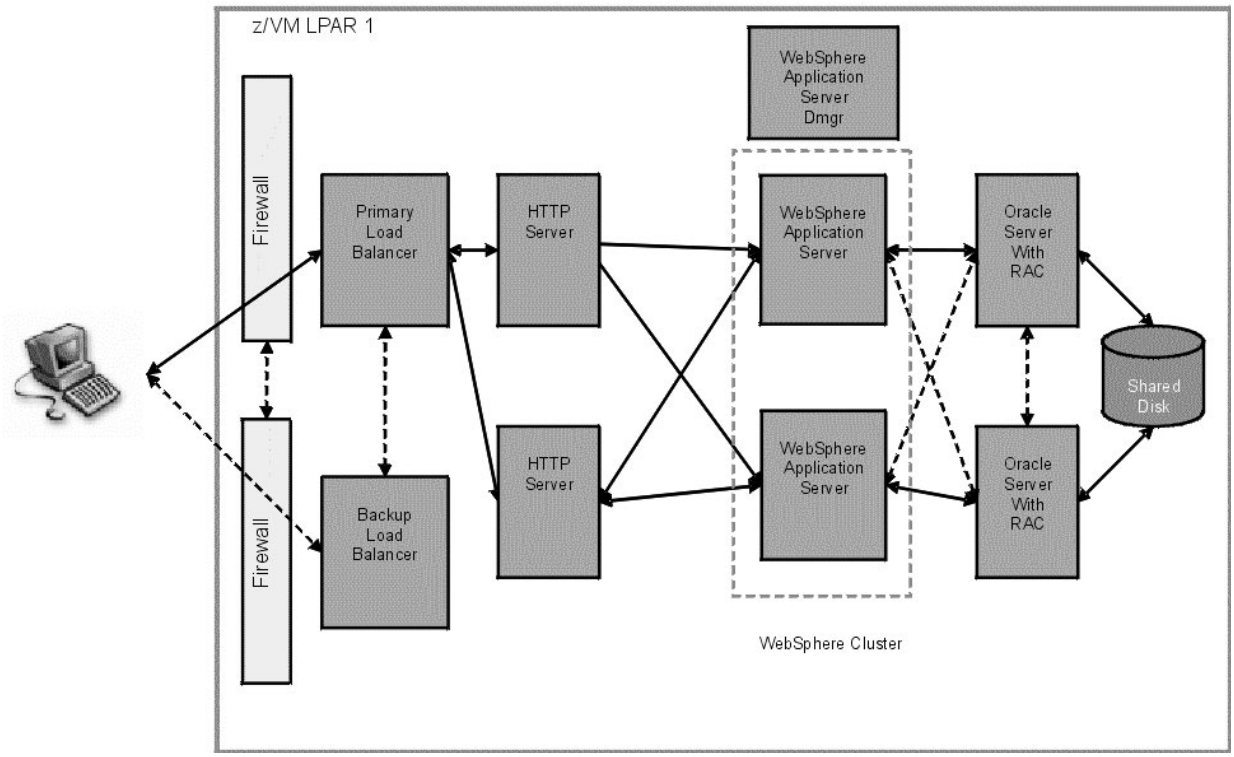


Figure 78. WebSphere with Oracle RAC database on Linux 62.

Installation overview of Oracle RAC

Deploying Oracle 10g Database on zSeries Linux in a high availability cluster is a rather straightforward procedure once the preparatory work has been completed. This document describes our preparatory work, installation, and configuration of Oracle 10g Real Application Clusters (RAC) in a 2 node cluster providing an HA back-end to our HA WebSphere Application Server clusters.

After viewing the Oracle documentation and release notes, the initial steps we completed for our installation was a basic Oracle 10g RAC installation as outlined in the IBM Redbook: Experiences with Oracle 10g Database on Linux for zSeries (SG24-6482). At the time of this writing Oracle 10G RAC was only supported on SUSE Linux Enterprise Server 9 (64 bit). We used the aforementioned Redbook to guide us through the majority of the installation procedure. Any problems we encountered with the installation by using the draft version of that Redbook were submitted for inclusion in the final Redbook which has since been made available.

Please note that this document is not meant to replace Oracle documentation. Please follow the current Oracle RAC installation documentation, available on www.otn.oracle.com, especially the release notes, prior to your installation, as there might be recent updates. The aforementioned Redbook and our document are just a point in time installation experience. The Redbook is on www.redbooks.ibm.com. There is also a document from the IBM Oracle Joint Solution Center in Montpellier available on http://www.oracleracsig.org. These are meant to give examples but the Oracle documentation should be the prime source of information.

Our installation utilized shared raw devices created on our DASD as the backing file system for the 2 Oracle server instances. We chose this method over the alternative Oracle Cluster File-system solution as we were more familiar with the technologies used to implement the former. We describe the basic steps of implementing the shared back-end later. Those steps include making a Logical Volume Manager (LVM) Group, partitioning the LVM group into logical volumes, creating identical raw devices on the Oracle server instances, binding the raw devices to the logical volumes from our volume group, then creating some convenient symbolic links from raw devices to our oracle installation directory.

Recommended Kernel Parameter Tuning

To begin our installation as we performed a stock installation of SLES 9 64 service pack 2. Issuing a *uname –a* command on the resulting systems showed it was at the following level:

```
Linux litsora1 2.6.5-7.191-s390x #1 SMP Tue Jun 28 14:58:56 UTC 2005 s390x
s390x s390x GNU/Linux
```

The following RPMs were needed in addition to the minimum graphical installation option:

- glibc-devel-2.3.3-98.47.s390x.rpm
- glibc-devel-32bit-9-200506070135.s390x.rpm
- cpp-3.3.3-43.34.s390x.rpm
- gcc-3.3.3-43.34.s390x.rpm
- gcc-c++-3.3.3-43.34.s390x.rpm
- libstdc++-3.3.3-43.34.s390x.rpm
- libstdc++-devel-3.3.3-43.34.s390x.rpm

We reviewed the Redbook and determined that the installation we had performed did not match the kernel parameter specification outlined in the Redbook. We placed the necessary overrides in the /etc/sysctl.conf file so they would be enabled at boot time. The following are the modifications we had to make on our stock SLES9 installation to perform our 10g installation successfully:

- net.ipv4.ip_forward = 0
- net.ipv4.conf.default.rp_filter = 1
- net.ipv4.conf.default.accept_source_route = 0
- kernel.core_uses_pid = 1
- kernel.shmall = 2097152
- kernel.shmmax = 2147483648
- kernel.shmmni = 4096
- kernel.sem = 250 32000 100 128
- fs.file-max = 65536
- net.ipv4.ip_local_port_range = 1024 65000
- net.core.rmem_default = 262144
- net.core.rmem_max = 262144
- net.core.wmem_default = 262144
- net.core.wmem_max = 262144

To make the changes take immediate effect, we issued “sysctl -p”. It is strongly recommended that you reboot to ensure the changes are made at every boot as the installer will fail if the overrides are not respected.

Creating a logical volume group

Physically, a RAC system consists of several servers connected to each other by a private interconnect. The database files are stored on a shared storage subsystem where they are made accessible to all nodes. The method we chose for our storage subsystem was raw devices on Logical Volume Manager (LVM) volumes comprised of ECKD DASD.

We created an LVM group called VGraw that was approximately 6 GB in capacity. We began by attaching some DASD devices to one of our Oracle servers. We then ensured it made appropriate /dev/ entries at boot time that were persistent. For each of our devices, which were the corresponding entries in /dev, we will generically call $DEVICE in the description that follows. In our specific example we added 4 3390 mod3 DASD packs, /dev/dasdc, /dev/dasdd, /dev/dasde, /dev/dasdf. Each of those entries was solely intended to be members of our volume group, and thus needed to be dasdfmt'ed as follows:

```
$> dasdfmt -y -b 4096 -p -f /dev/$DEVICE
```

Note that the -y starts immediately, and -p prints a progress bar for inspection. This procedure took several minutes to complete.

Then for each of the devices we dasdfmt'ed, we performed an fdasd as follows:

```
$> fdasd -a /dev/$DEVICE
```

Note that -a automatically makes one large partition which was sufficient for our needs, and prevented unnecessary prompting.

After that had completed successfully, we created the physical volumes on each of the devices:

```
$> pvcreate /dev/${DEVICE}1
```

To finish up our volume group creation, we then ran vgscan to setup the LVM environment. This command scanned all disks for volume groups and rebuilt any outdated LVM caches:

```
$> vgscan
```

The volume group needed Logical volumes carved out of it as per the Redbook instructions. We created logical volumes, associating them with our volume group "VGraw" as follows:

```
lvcreate -L  500M -n oracl_system_raw_500m       VGraw
lvcreate -L  800M -n oracl_sysaux_raw_800m       VGraw
lvcreate -L  500M -n oracl_undotbs1_raw_500m   VGraw
lvcreate -L  500M -n oracl_undotbs2_raw_500m   VGraw
lvcreate -L  250M -n oracl_temp_raw_250m          VGraw
lvcreate -L  160M -n oracl_example_raw_160m     VGraw
lvcreate -L  120M -n oracl_users_raw_120m         VGraw
lvcreate -L  120M -n oracl_redo1_1_raw_120m     VGraw
lvcreate -L  120M -n oracl_redo1_2_raw_120m     VGraw
lvcreate -L  120M -n oracl_redo2_1_raw_120m     VGraw
lvcreate -L  120M -n oracl_redo2_2_raw_120m     VGraw
lvcreate -L  110M -n oracl_control1_raw_110m    VGraw
lvcreate -L  110M -n oracl_control2_raw_110m    VGraw
lvcreate -L      5M -n oracl_spfile_raw_5m             VGraw
lvcreate -L      5M -n oracl_pwdfile_raw_5m          VGraw
lvcreate -L  100M -n ora_ocr_raw_100m            VGraw
lvcreate -L  100M -n ora_vote_raw_100m          VGraw
lvcreate -L  500M -n oracl_asm1_500m             VGraw
```

```
lvcreate -L  500M -n oracl_asm2_500m            VGraw
lvcreate -L  500M -n oracl_asm3_500m            VGraw
lvcreate -L  500M -n oracl_asm4_500m            Vgraw
```

The names we used include the sizing requested during the actual creation of the LVM volume. We observed that in some cases LVM rounded up to the next largest extent, and that had no observable impact on our installation.

With that process completed, we were now left with block devices on our volumes for use in our Oracle installation.

Creating raw devices

For performance reasons Oracle uses the block devices on our shared DASD volume only after they are bound as raw character devices. Once bound to a block device, a raw device can be opened, read and written, just like the block device it is bound to. However, the raw device does not behave exactly like the block device. In particular, access to the raw device bypasses the kernel's block buffer cache entirely: all I/O is done directly to and from the address space of the process performing the I/O. Thus we created some raw devices for use on each of our Oracle servers:

```
S> mknod /dev/raw/raw$i c 162 $i
```

We issued that command 100 times, substituting 0 – 100 for $i to ensure we would have enough devices.

To make sure these raw devices were persistent across reboots:

```
$> chkconfig raw on
```

Binding raw devices

Now that we actually had some raw devices created, we needed to bind our raw devices to the block devices we made earlier. To do this and ensure it would be persistent across reboots, we constructed an init script /etc/init.d/rawbind. By doing this, we ensured that the init process executes the binding before any of the Oracle init scripts, which are not yet installed. The file was created with the following contents:

```
raw /dev/raw/raw1  /dev/VGraw/oracl_system_raw_500m
raw /dev/raw/raw2  /dev/VGraw/oracl_sysaux_raw_800m
raw /dev/raw/raw3  /dev/VGraw/oracl_undotbs1_raw_500m
raw /dev/raw/raw4  /dev/VGraw/oracl_undotbs2_raw_500m
raw /dev/raw/raw5  /dev/VGraw/oracl_temp_raw_250m
raw /dev/raw/raw6  /dev/VGraw/oracl_example_raw_160m
raw /dev/raw/raw7  /dev/VGraw/oracl_users_raw_120m
raw /dev/raw/raw8  /dev/VGraw/oracl_redo1_1_raw_120m
raw /dev/raw/raw9  /dev/VGraw/oracl_redo1_2_raw_120m
raw /dev/raw/raw10 /dev/VGraw/oracl_redo2_1_raw_120m
raw /dev/raw/raw11 /dev/VGraw/oracl_redo2_2_raw_120m
raw /dev/raw/raw12 /dev/VGraw/oracl_control1_raw_110m
raw /dev/raw/raw13 /dev/VGraw/oracl_control2_raw_110m
raw /dev/raw/raw14 /dev/VGraw/oracl_spfile_raw_5m
raw /dev/raw/raw15 /dev/VGraw/oracl_pwdfile_raw_5m
raw /dev/raw/raw16 /dev/VGraw/ora_ocr_raw_100m
raw /dev/raw/raw17 /dev/VGraw/ora_vote_raw_100m
raw /dev/raw/raw18 /dev/VGraw/oracl_asm1_500m
raw /dev/raw/raw19 /dev/VGraw/oracl_asm2_500m
raw /dev/raw/raw20 /dev/VGraw/oracl_asm3_500m
raw /dev/raw/raw21 /dev/VGraw/oracl_asm4_500m
raw /dev/raw/raw1  /dev/VGraw/oracl_system_raw_500m
$> ln -s /etc/init.d/rawbind    /etc/rc.d/rc3.d/S57rawbind
$> chmod 0755 /etc/init.d/rawbind
```

After completing the init script, we made sure it was placed early in our default runlevel (runlevel 3) and was executable:

Then we ensure the script would be run at boot time:

```
$> chkconfig rawbind on
```

When this step was completed, we rebooted our servers to ensure the changes had taken effect and were persistent across reboots.

Since we were now dealing directly with raw devices that had non-obvious names, like raw12, we made some convenience symlinks to our raw devices to make the actual Oracle installation procedure more intuitive. We planned our installation to be inside the /oracle/10g/ directory on our servers, so we defined the convenience symlinks with the following commands:

```
ln -s /dev/raw/raw1  /oracle/10g/oradata/oracl_system_raw_500m
ln -s /dev/raw/raw2  /oracle/10g/oradata/oracl_sysaux_raw_800m
ln -s /dev/raw/raw3  /oracle/10g/oradata/oracl_undotbs1_raw_500m
ln -s /dev/raw/raw4  /oracle/10g/oradata/oracl_undotbs2_raw_500m
ln -s /dev/raw/raw5  /oracle/10g/oradata/oracl_temp_raw_250m
ln -s /dev/raw/raw6  /oracle/10g/oradata/oracl_example_raw_160m
ln -s /dev/raw/raw7  /oracle/10g/oradata/oracl_users_raw_120m
ln -s /dev/raw/raw8  /oracle/10g/oradata/oracl_redo1_1_raw_120m
ln -s /dev/raw/raw9  /oracle/10g/oradata/oracl_redo1_2_raw_120m
ln -s /dev/raw/raw10 /oracle/10g/oradata/oracl_redo2_1_raw_120m
ln -s /dev/raw/raw11 /oracle/10g/oradata/oracl_redo2_2_raw_120m
ln -s /dev/raw/raw12 /oracle/10g/oradata/oracl_control1_raw_110m
ln -s /dev/raw/raw13 /oracle/10g/oradata/oracl_control2_raw_110m
ln -s /dev/raw/raw14 /oracle/10g/oradata/oracl_spfile_raw_5m
ln -s /dev/raw/raw15 /oracle/10g/oradata/oracl_pwdfile_raw_5m
ln -s /dev/raw/raw16 /oracle/10g/oradata/ora_ocr_raw_100m
ln -s /dev/raw/raw17 /oracle/10g/oradata/ora_vote_raw_100m
```

Once again, we labeled the raw device with the rough size and intent of the underlying shared volume block devices they map to. These symlinks are what we will actually use in the Oracle installation.

The Oracle installation knows about these symlinks only after you have created a dbca mapfile that the Oracle installer can examine. We constructed our map file with the following contents that correspond to the symlink locations we chose:

```
system=/oracle/10g/oradata/oracl_system_raw_500m
sysaux=/oracle/10g/oradata/oracl_sysaux_raw_800m
undotbs1=/oracle/10g/oradata/oracl_undotbs1_raw_500m
undotbs2=/oracle/10g/oradata/oracl_undotbs2_raw_500m
temp=/oracle/10g/oradata/oracl_temp_raw_250m
example=/oracle/10g/oradata/oracl_example_raw_160m
users=/oracle/10g/oradata/oracl_users_raw_120m
redo1_1=/oracle/10g/oradata/oracl_redo1_1_raw_120m
redo1_2=/oracle/10g/oradata/oracl_redo1_2_raw_120m
redo2_1=/oracle/10g/oradata/oracl_redo2_1_raw_120m
redo2_2=/oracle/10g/oradata/oracl_redo2_2_raw_120m
control1=/oracle/10g/oradata/oracl_control1_raw_110m
control2=/oracle/10g/oradata/oracl_control2_raw_110m
spfile=/oracle/10g/oradata/oracl_spfile_raw_5m
```

That completed our definitions in the dbca map file.

Configuring the host file

We then needed to alter the /etc/hosts file on each RAC node for proper inter-node communication. We discovered that the name of the RAC node must not be listed as an alias for the loopback address in the /etc/hosts file.

For example on any of the RAC nodes that is incorrect:

```
# 127.0.0.1        rac1pub localhost.localdomain localhost
```

The entry should look like this:

```
 # 127.0.0.1        localhost.localdomain localhost
```

If the RAC node is listed for the loopback address, you might later get the following errors:

- ORA-00603: ORACLE server session terminated by fatal error
- ORA-29702: error occurred in Cluster Group Service operation

Creating oracle users and groups

The next step we needed to perform was the creation of the user who would own the Oracle services and installation file. These instructions were carried out on all Oracle servers.

In order to ensure proper permissions, we began with the creation of the groups associated with a typical Oracle installation, “DBA” and “OINSTALL”.

```
$> groupadd dba
$> groupadd oinstall
```

We created a user ID called ″oracle″ with the primary group as ″DBA″ by issuing the following:

```
$> useradd -m -g dba -G dba -p $PASSWORD oracle
```

We chose a password our administrators had agreed upon, and used it in place of the “$PASSWORD” in the command above.

Then we ensured the user ID oracle had write privileges to our installation directory “/oracle”:

```
$> chown -R oracle:dba /oracle
```

Then we were able to proceed with the creation of the requisite ssh keys for the oracle user. We used the following commands:

```
$> /usr/bin/ssh-keygen -q -t rsa -f ~/.ssh/id_rsa -C ''-N ''
```

Then we set the permissions appropriately with the following:

```
$> chmod 600 ~/.ssh/id_rsa
$> chmod 644 ~/.ssh/id_rsa.pub
```

Modifying the Oracle environment setup

With the previous steps completed, we went to each server individually and performed the system specific customization required for an Oracle RAC installation.

We needed to set the Oracle environment variables for the RAC nodes, including the proper assignment of a unique Oracle SID. We modified the /etc/profile on each of the RAC nodes. For example, the database name we chose was ″orcl″, but the SIDs are ″orcl1″ for RAC node 1, ″orcl2″ for RAC node 2, and so on. We essentially copied the following to the bottom of the /etc/profile on each node, substituting the appropriate SID for each:

```
# Set the SID
# THIS IS DIFFERENT FOR EACH NODE
export ORACLE_SID=orcl1
```

```
# We also set the library path to include the oracle libs needed later
export LD_LIBRARY_PATH=$ORACLE_HOME/lib

# Lastly, we made sure the following env vars were NOT set.
unset ORACLE_HOME
unset TNS_ADMIN
```

Setting up SSH Key exchange

Now that the prerequisite users and environment work had been completed, we moved on to exchanging the cryptographic keys we made earlier. We did this to enable the password-less ssh communication between RAC nodes in our cluster as the oracle user we created earlier. To do this, as the oracle user on each Oracle RAC node:

```
$> cat ~/.ssh/id_rsa.pub | ssh $OTHER_NODE_IP \"cat -
>>.ssh/authorized_keys\"
```

We did this for each of the other nodes in the cluster, substituting the IP address of the cluster nodes for the $OTHER_NODE_IP during each iteration until we had exchanged every key with every node. During the graphical portion of the installation, the software will copy components to each node. Thus it is vital that this passwordless communication works without error

Stepping through the GUI installation procedure

At this point we ran the Oracle graphical installation tool. We followed the prompts on screen as recommended in the Redbook. We made sure to have our Oracle technical support rep available to confirm our selections. We continued through the graphical installation until it had completed satisfactorily.

Configuring Oracle HA

Before we continue with our description of the basic installation procedures, let us take a moment to familiarize ourselves with the terminology used in the remainder of this document.

Oracle Connection-time failover (CTF)

Connection-time fail-over refers to a client attempting to connect to a second listener when the attempt to connect to the first listener fails. To implement this, we created a new net service name with the database name as its identifier. Then we simply put the address information of all nodes in the cluster in this newly created global net service name.

When implementing CTF you may use one of the following two options:

FAILOVER=ON

(Setting this means: try each address in order until one succeeds.)

FAILOVER=ON LOAD_BALANCE=ON

(Setting this means: try each address randomly until one succeeds.)

Transparent application fail-over (TAF)

Transparent application fail-over enables an application to automatically reconnect to a database if the connection is broken. Active transactions roll back, and the new database connection is identical to the original one no matter how the connection is lost.

Implementing transparent application fail-over requires you to manually configure tnsnames.ora with the specification of a primary node, a backup node, fail-over type, and fail-over method. You can also combine RAC CTF with RAC TAF. We did not test all possible configurations for the RAC TAF implementations as there are 12 of them in total, 2 CTF x 3 TYPE x 2 METHOD.

The parameter TYPE specifies the type of failover. This is a required parameter for the TAF fail-over. There are three options: SESSION, SELECT, and NONE. Selection of SESSION means that RAC fails over the session. A new session will be created automatically for use on the backup if a user's connection is lost. This type of failover does not attempt to recover selects after a fail-over. Choosing SELECT means that RAC allows users with open cursors to continue fetching on them after a fail-over. Although the performance for this type of TAF during a fail-over is good, it has an overhead during normal select operations. Selection of NONE means that no fail-over function is used. This is the default. For all our testing the type SESSION was specified in the tnsnames.ora file.

The parameter METHOD specifies how fast a fail-over occurs from the primary node to the backup node. There are two options: BASIC and PRECONNECT. Selection of BASIC means that RAC establishes connection at fail-over time. This option requires almost no work on the backup server until fail-over time. Selection of PRECONNECT means that RAC pre-establishes connections all the time. This is good for failover performance, but can slow down normal operation, since it requires the backup instance to support all connections from every supported instance. For all of our testing the method PRECONNECT was specified in the *tnsnames.ora* file.

Configuring RAC: Deploying Oracle 10g Database on zSeries Linux in a high availability is a rather straightforward procedure if you are experienced with Oracle server administration.

When you have completed installing Oracle 10g Real Application Cluster (RAC) on your servers, you must then enable Transparent Application Failover (TAF) on each. To accomplish this we needed to create/modify the tnsnames.ora files on the cluster nodes. Each should have identical configurations when finished. In our case the file was created as follows:

```
ORCLCLUSTER=
   (description=
      (address=
          (protocol=tcp)
          (host=litsora1)
          (port=1521))
      (connect_data=
          (service_name=ORCLCLUSTER)
          (failover_mode=
             (backup=litsora2)
             (type=session)
             (method=preconnect)
             (retries=0)
             (delay=10))))

LISTENERS_ORCL =
  (ADDRESS_LIST =
    (ADDRESS = (PROTOCOL = TCP)(HOST = vip_litsora1)(PORT = 1521))
    (ADDRESS = (PROTOCOL = TCP)(HOST = vip_litsora2)(PORT = 1521))
  )
```

We had a cluster of 2 systems with hostname litsora1 and litsora2 respectively. We selected PRECONNECT as our failover method among the various choices offered by Oracle RAC. Your application needs may require you to select different parameters, but should otherwise operate in a similar fashion.

Once this file is in place you can restart the lsnrctrl process to ensure your configuration is valid. We found that running the lsnrctrl process on one of the servers is sufficient. Any syntax errors will issue warnings when trying to connect to your database instance.

When you have completed the installation of TAF you are then ready to try and connect your WebSphere Application Server to the Oracle backend.

Configuring WebSphere Application Server to use Oracle RAC: In order to do this you can use the thin client JDBC driver provided with the Oracle database installation you have performed. From any of your machines with a RAC installation, locate the appropriate driver ″ojdbc14_g.jar″ or ″ojdbc14.jar″. The former contains debugging capabilities, while the latter is sufficient in most deployments.

Once you have located the appropriate driver you need to place it on your WebSphere Application server. We put ojdbc14.jar under /opt/oracle on all of our WebSphere Application Server nodes. If you are running in a cluster and would like to use ″Test Connection″ from the WebSphere Application Server Network Deployment manager node, then you need to copy the driver onto that system as well.

Then, log onto the WebSphere Application Server admin console (if you are running in a WebSphere Application Server cluster configuration, log onto the WebSphere Application Server Network Deployment manager admin console). Select Resources -> JDBC Providers. Create a new JDBC provider under the scope appropriate for your WebSphere Application Server configuration.

In the JDBC Providers new provider panel, select Oracle as the database type. Select Oracle JDBC Driver as the provider type. Then select the implementation type you want for your application. We used Connection Pool datasource for our test application. Now in the summary panel, change the Class path to point to the location of the JDBC driver. In our case the path was /opt/oracle/ojdbc14.jar. Click Apply and Save to Master Configuration.

Now you are ready to create a datasource for the JDBC Provider you just created. In the Oracle provider panel, click Data Sources under Additional Properties. Click the New button to create a new datasource. Change the Name and JNDI name appropriately, and select Oracle 10g data store helper under Data store helper class name. Select the authentication alias appropriate for connecting to the Oracle RAC under Component-managed authentication alias. In our case we created a J2C authentication data entry with the user name and password needed to connect to our Oracle database, and chose that as the Component-managed authentication alias.

Finally for the Oracle data source properties, we used the following as the connection URL (in one long string):

```
jdbc:oracle:thin:@(description=(address=(protocol=tcp)(host=litsora1)(port=1521))
(connect_data=(service_name=ORCLCLUSTER)(failover_mode=(backup=litsora2)(
type=session)(method=preconnect)(retries=0)(delay=10))))
```

We are not Oracle experts but we believe that what you pass as the connection URL indicates the HA mode of RAC. See below for an explanation of each mode, followed by our test results for each.

Oracle RAC offers three different methods of high availability. Please see the Redbook IBM WebSphere V5.1 Performance, Scalability, and High Availability WebSphere Handbook Series (SG24-6198-01), chapter 12, for explanations on each.

1. "Failover PRECONNECT"
2. "Failover BASIC" on page 305
3. "Load balancing" on page 305

Testing our WebSphere Application Server application with Oracle RAC:
Before testing each of the HA modes, we did the following to ensure a clean start.

1. Stopped the application in WebSphere Application Server
2. Updated the WebSphere Application Server application datasource to use the connection URL that corresponds to the HA mode. The steps are described under ″WebSphere Application Server configuration″
3. Restarted WebSphere Application Server and cluster members to reload the JDBC driver with the change made in step 2
4. Tested the datasource by clicking the ″Test Connection″ button on the datasource page in the WebSphere Application Server admin console. We saw a ″Test successful″ message on the admin console when WebSphere Application Server was able to connect to the Oracle database successfully.
5. Started the application in WebSphere Application Server
6. Tested the connectivity by opening a browser, pointing it to the application Web page, and performing simple transactions with the Web application that required communication with the database
7. In order to test the high availability of Oracle RAC, we needed to simulate many users logging on to the application and performing transactions at the same time. For this we used the WebSphere Studio Workload Simulator engine to generate a workload simulating 500 users. For monitoring the throughput, we used the controller from the same product suite.

Note: Because we are an integration test team - we are sharing our test systems with other workloads. Keep in mind that we did not have exclusive use of system resources while you interpret the performance aspects of the test results presented below.

Also note that our servers were never fully stressed. Oracle RAC servers running as high as 80% are common, but our workload infrastructure was insufficient to drive that much stress, so results might be different at higher CPU loads.

Failover PRECONNECT: The first HA mode we tried was failover PRECONNECT. The connection URL in the WebSphere Application Server application datasource used is:

```
jdbc:oracle:thin:@(description=(address=(protocol=tcp)(host=litsora1)(port=1521))
(connect_data=(service_name=ORCLCLUSTER)(failover_mode=(backup=litsora2)(type=session)
(method=preconnect)(retries=0)(delay=10))))
```

We noticed a 15% CPU utilization on litsora1 the primary RAC server, 2.5% on litsora2 the standby RAC server, we saw a 35 pages/sec throughput on the WebSphere Studio Workload Simulator monitor. Then we brought down litsora1 by halting the system, and the throughput went to 0 pages/sec for 2 minutes (with ″HTTP timeout″ errors seen on the monitor). Then transaction rate picked up and connections were established with litsora2. We noticed a 10% CPU util on litsora2 and we started seeing a steady throughput on the monitor.

Then we brought back litsora1 and noticed very poor throughput for about 2 minutes (about 0 - 4 pages/sec) until the listener on litsora1 was back to normal and throughput went back to normal.

At this point we noticed that transactions were still all going to litsora2 even though litsora1 was back. Then we brought down litsora2, and transactions dropped to 0 and never recovered. It seemed like the JDBC driver and hence the application never knew when litsora1 came back so this second failover didn't work.

Failover BASIC: Then we tested the Fail Over mode with passive connect method, using the following Oracle data source URL in WebSphere Application Server application datasource:

```
jdbc:oracle:thin:@(description=(address=(protocol=tcp)(host=litsora1)(port=1521))
(connect_data=(service_name=ORCLCLUSTER)(failover_mode=(backup=litsora2)(type=session)
(method=basic)(retries=0)(delay=10))))
```

We noticed a 11-18% CPU utilization on litsora1 the primary RAC server, and 4-10% on litsora2 the standby RAC server, we were seeing a 35 pages/sec throughput on the WebSphere Studio Workload Simulator monitor. Then we brought down litsora1 by halting the system, and the throughput went to 0 pages/sec for 2 minutes (with ″HTTP timeout″ errors seen on the monitor). Then transaction rate picked up and connections were established with litsora2. We noticed a 10-20% CPU utilization on litsora2 and we started seeing a steady throughput the JIBE monitor. So it seemed that litsora2 was picking up where litsora1 left off.

Then we brought back litsora1 and noticed a little dip in the throughput for a few seconds. When the listener on litsora1 was back to normal, the throughput went back to normal.

Most of our transactions were still going to litsora2 but some were going to litsora1. So we decided to bring down litsora2 to see if transactions would go to litsora1. When this happened, transaction rate dropped to 0 pages/sec for 2 minutes and then it started to pick up again going to litsora1. When we brought litsora2 back, we didn't notice a difference in transaction rate.

Now both litsora1 and litsora2 were up and running but transactions were only going to litsora1. When we killed litsora1 for the second time, transactions stopped going through completely.

Load balancing: We also tested the Load Balancing mode of Oracle RAC by using the following Oracle data source URL in the WebSphere Application Server application datasource:

```
jdbc:oracle:thin:@(DESCRIPTION=(LOAD_BALANCE=ON)(ADDRESS=(PROTOCOL=TCP)
(HOST=litsora1)(PORT=1521))(ADDRESS=(PROTOCOL=TCP)(HOST=litsora2)(PORT=1521))
(CONNECT_DATA=(SERVICE_NAME=orclcluster)))
```

Transactions were going to the two servers about equally. We noticed about equal CPU utilization on litsora1 and on litsora2. When we brought litsora1 down, transactions went to 0 pages/sec (with ″HTTP timeout″ errors seen on the monitor) but picked up again after 2 minutes. At this point all transactions were going to litsora2. After a few minutes of steady throughput and no errors, we brought litsora1 back. Throughput took a dip (about 4 - 7 pages / sec) for about 4 minutes, then transaction rate went back to normal. All transactions were still going to litsora2. Now that both litsora1 and litsora2 were back, we killed litsora2 this time, and transactions stopped going through completely.

Implementing HA Reference Architecture: WebSphere with DB2 database on z/OS

For this architecture we set up and tested a highly available database using a DB2 z/OS data sharing group as the data store, as shown in Figure 79. Currently there are two ways of connecting to the DB2 z/OS data sharing group from Linux on zSeries. One is by using the sysplex aware JDBC Type 4 Driver as the DB2 Universal JDBC Driver in WebSphere Application Server. The second is by going through a DB2 Connect instance with sysplex awareness turned on. We had the opportunity to test both. The picture above depicts the JDBC Type 4 Driver flow.

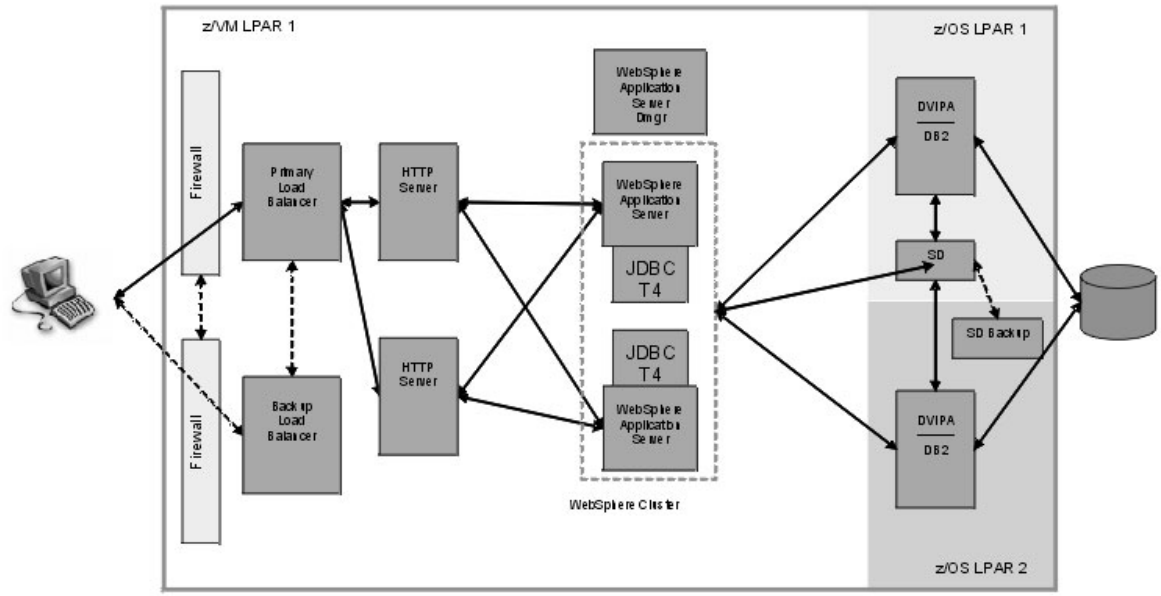


Figure 79. WebSphere with DB2 database on z/OS.

We configured the JDBC Type 4 Driver or DB2 Connect instance to connect to the sysplex distributor IP address first. The IP address is the location or group Dynamic VIPA (Virtual IP Address) that is associated with the DB2 data sharing group. The sysplex distributor is the strategic IBM solution for IP connection workload balancing in a z/OS Parallel Sysplex. By connecting to the Dynamic VIPA first, the workload is routed to the least busy DB2 data sharing member as determined by WLM (z/OS Workload Manager). After the initial connection, specific DB2 members' IP addresses are returned to the JDBC Type 4 Driver or DB2 Connect instance and all subsequent requests route to the members directly. For more details on the technologies talked about here please refer the High Availability Architectures For Linux on IBM System z document here:

`http://www.ibm.com/servers/eserver/zseries/library/whitepapers/pdf/HA_Architectures_for_Linux_on_System_z.pdf`

We'll now discuss how we configured the JDBC Type 4 Driver or DB2 Connect instance for sysplex awareness and testing high availability using these technologies.

Using JDBC Type 4 Driver

This section discusses enabling sysplex workload balancing on JDBC Type 4 Driver and running our test.

Enabling sysplex workload balancing on JDBC Type 4 Driver: Only driver levels 2.7 or later support sysplex awareness. DB2 V8.2 FP3, also known as DB2 V8.1 FP10 and above ships with the appropriate driver level. To setup sysplex awareness please refer to the topic "DB2 Universal JDBC Driver connection concentrator and Sysplex workload balancing" located at:

`http://publib.boulder.ibm.com/infocenter/db2luw/v8/topic/com.ibm.db2.udb.rn.doc/rn/r0012130.htm#wq575`

We'll just touch on a couple of areas that may appear vague from the guide. To enable sysplex workload balancing with JDBC Type 4 driver, you have to do two things. The first is to specify the sysplex parameters in the DB2JccConfiguration.properties file as noted in the aforementioned guide. The second, which we'll elaborate on, is to enable sysplex workload balancing on the data source in WebSphere Application Server.

Adding the directory path for DB2JccConfiguration.properties to the WebSphere Application Server DB2 Universal JDBC Driver classpath: From the WebSphere Application Server admin console, go to Resources -> JDBC Providers, and select your DB2 Universal JDBC Driver Provider under the appropriate scope. Under Class path, add the directory path to your DB2JccConfiguration.properties file, making sure to use a new line for the path. For example, ours looked like this (the last one being the directory path):

```
/opt/IBM/db2/V8.1/java/db2jcc.jar
/opt/IBM/db2/V8.1/java/db2jcc_license_cu.jar
/opt/IBM/db2/V8.1/java/db2jcc_license_cisuz.jar
/opt/IBM/db2/V8.1/java/db2jcc_javax.jar
/opt/IBM/db2/V8.1/
```

Setting the sysplex properties for the data source that your application uses to connect to the database server: From the WebSphere Application Server admin console, go to Resources -> JDBC Providers, and select your DB2 Universal JDBC Driver Provider under the appropriate scope. Select Data sources under Additional Properties on your right. Then select the data source that your application uses to connect to the database server. Select Custom properties under Addition Properties on your right. Then add the sysplex properties as mentioned in the aforementioned guide.

Running our test with multiple TCP/IP stacks on z/OS: We had two TCP/IP stacks (public and private) each running with its own DDF configuration. Our DB2 members on z/OS listened on both stacks. We only configured a private IP address on our WebSphere Application Server. Even though DB2 responded to requests from both stacks, a -DISPLAY DDF command on DB2 showed that it bound to only the public stack. With sysplex awareness enabled, upon the initial successful connection, the driver fetched the public IP addresses of the DB2 members and subsequent connections failed because the application running on WebSphere Application Server couldn't establish a connection to the public IP addresses. We saw trace messages in WebSphere Application Server's SystemOut.log as well as db2jcc dump log that indicated the application wouldn't establish a connection to the public stack.

We added a public IP address to our WebSphere Application Server and application transactions started going through to the DB2 members through the public stack.

Our test results: For ease of understanding, J80 is a z/OS system where one of our DB2 datasharing group members runs. Before we ran the test we checked the WLM weight of J80 and it had the highest weight of all the systems:

```
TOTALCONN: 0000000174  RDY: 001  WLM: 08  TSR: 085
```

We decided to cancel IRLM (The Internal Resource Lock Manager subsystem manages DB2 locks, each DB2 member has its own corresponding IRLM) to simulate a DB2 failure on J80 for this test, while a workload simulating 500 unique users was running. We had to disable ARM (z/OS Automate Restart Manager) because it would restart the failed DB2 member automatically when we cancelled IRLM thus making our test impossible to accomplish.

At the start of the test there were threads going to a number of DB2 members, including J80. When we cancelled IRLM by issuing F IRLMxx,ABEND, we didn't notice a drop in transaction rate or any error messages on the client side. On the WebSphere Application Server system, we noticed the following messages in SystemOut.log:

```
Caused by: javax.ejb.EJBException: TradeBean.getClosedOrders - error; nested exception is: javax.ejb.Tran
actionRolledbackLocalException:  ; nested exception is: javax.ejb.TransactionRolledbackLocalException:  ;
nested exception is: com.ibm.websphere.ce.cm.StaleConnectionException: A connection failed but has been r
-established. The hostname or IP address is "J90VIPA.pdl.pok.ibm.com" and the service name or port number
is 446 . Special registers may or may not be re-attempted (Reason code = 1  DB2ConnectionCorrelator: G90C
4BB.0168.BE5FFD337F4B
        ... 20 more
```

J90VIPA is the VIPA of another z/OS system running a different DB2 member in the datasharing group. When DB2 on J80 was brought down, failed connections were re-directed to another DB2 member and no failures were detected on the client side.

We waited 20 - 30 minutes with DB2 on J80 down, and still didn't notice any errors on the client side or any drop in throughput. By doing a netstat on the WebSphere Application Server system we saw that there were 137 total connections established with various DB2 members. We brought DB2 on J80 back, and didn't notice a significant change in throughput. After a few minutes, we started seeing transactions going to DB2 on J80. On the WebSphere Application Server end with netstat command, we saw 50 connections to J80, and a total of 152 to various DB2 members including the one on J80.

Using DB2 Connect

This section consists of:

Enabling sysplex workload balancing on DB2 Connect: Please refer to this white paper/book: *"IBM DB2 Connect: Quick Beginnings for DB2 Connect Enterprise Edition Version 8", GC09-4833-00*, chapter 15, for details on configuring your DCS database catalog entry for Sysplex support. We used DB2 Connect EE V8.2 FP3, also known as V8.1 FP10. To get both sysplex load balancing and connection concentrator support (the later enables balancing at the transaction boundary, rather than the coarser connection boundary), you need at least DB2 Connect V8.2, and it must be talking to z/OS DB2 V6.1 or higher. To just get sysplex load balancing but without the capability of balancing at transaction

boundaries, older levels of DB2 Connect EE are supported. Please refer to the DB2 Connect EE release notes for support information.

We'll touch up on one area that might appear vague in the guide. We'll also talk about WebSphere Application Server datasource configuration to go through DB2 Connect for DB2 z/OS communication.

Enabling sysplex in DCS catalog entry: The guide provides an example of configuring DB2 Connect for sysplex support. It provides sample db2 commands to enable the DCS catalog entry for sysplex support. We found that by executing the command from the Linux command prompt directly, we would hit syntax errors like the following.

```
db2inst1@litdbcon:~> db2 catalog dcs database DBLNXTR3 as USIBMT6PETDB2 parms ',,,,,sysplex'
SQL0104N  An unexpected token "," was found following "PARMS".  Expected
tokens may include:  "<character-string>".  SQLSTATE=42601
```

We found out that this is an issue with character entry and you can do one of the following to work around it:

1. You can use quotes around the catalog command, with a \ in front of any ' which would result in:

   ```
   db2 "catalog dcs database DBLNXTR3 as USIBMT6PETDB2 parms \',,,,,sysplex\'"
   ```

   Alternatively,

2. You can enter the db2 command environment by issuing a simple command: 'db2' That will put you in the environment and single quotes will work as normal. We used this method and were able to add DCS entry with the sysplex parameter:

   ```
   db2 => catalog dcs db DBLNXTR3 as USIBMT6PETDB2 parms ',,,,,sysplex'
   DB20000I  The CATALOG DCS DATABASE command completed successfully.
   DB21056W  Directory changes may not be effective until the directory cache is
   refreshed.
   db2 => list dcs directory

    Database Connection Services (DCS) Directory

    Number of entries in the directory = 1

   DCS 1 entry:

    Local database name                 = DBLNXTR3
    Target database name                = USIBMT6PETDB2
    Application requestor name          =
    DCS parameters                      = ,,,,,sysplex
    Comment                             =
    DCS directory release level         = 0x0100
   ```

Setting up WebSphere Application Server data source: We were going through DB2 Connect now which handled all the sysplex workload balancing. So we removed all the sysplex properties from the data source that were leftover from our JDBC Type 4 driver sysplex testing. We used the same driver to connect to DB2 Connect. We just had to make sure that the data source was going directly to our DB2 Connect system instead of directly to the sysplex distributor.

Another interesting aspect was that for the data source's Component-managed Authentication Alias, we had to use authentication data for DB2 z/OS, not DB2 Connect, even though we were going through DB2 Connect.

Running our DB2 Connect test: For ease of understanding, J80 is a z/OS system where one of our DB2 datasharing group members runs. Before we ran the test we checked the WLM weight of J80 and it had the highest weight of all the systems:

```
TOTALCONN: 0000000174  RDY: 001  WLM: 08  TSR: 085
```

We cancelled IRLM (IMS/VS resource lock manager) to simulate a DB2 failure for this test while a workload simulating 500 unique users was running. Like the JDBC Type 4 test, we had to disable ARM (z/OS Automate Restart Manager) because it would restart the failed DB2 member automatically when we cancelled IRLM thus making our test impossible to accomplish.

Test results were similar to that of the JDBC Type 4 test. At the start of the test there were threads going to a number of DB2 members, including DB2 on J80. When we cancelled IRLM on J80, we didn't notice a drop in transaction rate or any error messages on the client side.

We waited 20 - 30 minutes with the DB2 on J80 down, and still didn't notice any errors on the client side or any drop in throughput. We brought DB2 on J80 back, and didn't notice a significant change in throughput. After a few minutes, we started seeing transactions going to DB2 on J80.

Chapter 20. Migrating middleware

In the last report we talked about how we migrated our Linux Virtual Servers to the 2.6 kernel. This time around we revisited the migration of Tivoli Access Manager for e-business WebSEAL and found that the procedure for migration could've been done in a simpler, more efficient way.

We also migrated our WebSphere Application Servers from v5.1 to v6.0.2.5 because version 6 includes the HAManager capability for recovering in-flight transactions from a failed server (See our section on "Setting up and Testing High Availability Architecture") and because starting with version 6.0.2, WebSphere Application Server comes with a 64-bit version which aligns with IBM's strategic move to the 64-bit Linux kernel.

Migrating Tivoli Access Manager for e-business WebSEAL from 2.4 kernel to 2.6 kernel

We have updated the migration procedures for WebSEAL since the last test report. Please use the updated version when performing your WebSEAL migration from 2.4 kernel to 2.6 kernel as it contains the latest information. We migrated WebSEAL 5.1 on SLES 8 to WebSEAL 5.1.0.13 on SLES 9.

For ease of understanding, in the sample outputs, we use <old-littam02> to denote the old WebSEAL server on SLES 8, and <new-littam02> to denote the new WebSEAL server on SLES 9.

Backing up WebSEAL data

Before running the pdbackup command, we had to modify the WebSEAL backup file which contains the information of the WebSEAL instance configuration files:

```
/opt/pdweb/etc/amwebbackup.lst
```

By default the amwebbackup.lst file will use the default instance name of webseald-default. Since we created and used a unique webseal instance for our testing, we had to modify amwebbackup.lst with the correct webseal instance name – WebSEAL1. We made a global change to the amwebbackup.lst through vi, the word editor, with the following command:

```
%s/default/WebSeal1/g
```

Once the amwebbackup.lst file is updated, we backed up the WebSEAL and PD Runtime data using pdbackup:

```
<old-littam02>:~ # pdbackup -a backup -list \ /opt/PolicyDirector/etc/pdbackup.lst
<old-littam02>:~ # pdbackup -a backup -list \ /opt/pdweb/etc/amwebbackup.lst
```

We checked the output of the log file produced after running pdbackup, and verified the return code of 0 for a successful backup.

The backup process created archive files with PD Runtime and WebSEAL data in the format of *list_date.time.tar* such as the following:

```
/var/PolicyDirector/pdbackup/pdbackup.lst_21Feb2006.18_12.tar
/var/PolicyDirector/pdbackup/amwebbackup.lst_21Feb2006.18_14.tar
```

Applying FP13 to original system, verify functionality

We upgraded GSKit before applying Fixpack 13:

```
<old-littam02>:~/eTAM/FP13 # rpm -qa | grep gsk
gsk7bas-7.0-1.13

<old-littam02>:~/eTAM/FP13 # rpm -Uvh gsk7bas-7.0-3.9.s390.rpm
gsk7bas                 ##########################
```

We stopped PDWEB:

```
<old-littam02>:~/eTAM/FP13 # pdweb stop
```

We applied Tivoli WebSEAL Fixpack 13:

```
<old-littam02>:~/eTAM/FP13 # rpm -Uvh PDRTE-PD-5.1.0-13.s390.rpm
PDRTE-PD                ##############################
<old-littam02>:~/eTAM/FP13 # rpm -Uvh PDWebRTE-PD-5.1.0-13.s390.rpm
PDWebRTE-PD             ##############################
<old-littam02>:~/eTAM/FP13 # rpm -Uvh PDWeb-PD-5.1.0-13.s390.rpm
PDWeb-PD                ##############################
```

We restarted PDWEB:

```
<old-littam02>:~/eTAM/FP13 # pdweb start
Starting the: webseald-WebSeal1
<old-littam02>:~/eTAM/FP13 # pdweb status
webseald-WebSeal1 yes yes
```

We verified the version:

```
<old-littam02>:~/eTAM/FP13 # rpm -qa | grep PD
PDRTE-PD-5.1.0-13
PDWebRTE-PD-5.1.0-13
PDWeb-PD-5.1.0-13
<old-littam02>:~/eTAM/FP13 # pdversion
IBM Tivoli Access Manager Runtime 5.1.0.13
IBM Tivoli Access Manager Policy Server Not Installed
IBM Tivoli Access Manager Web Portal Manager Not Installed
IBM Tivoli Access Manager Application Developer Kit Not Installed
IBM Tivoli Access Manager Authorization Server Not Installed
IBM Tivoli Access Manager Java Runtime Environment Not Installed
IBM Tivoli Access Manager Policy Proxy Server Not Installed
IBM Tivoli Access Manager WebSEAL Server 5.1.0.13
```

We verified functionality by checking that we could still view the junction details created on the earlier version of WebSEAL:

```
pdadmin sec_master> s t WebSeal1-webseald-littam02 show /was
    Junction point: /was
    Type: SSL Proxy
    Junction hard limit: 0 - using global value
    Junction soft limit: 0 - using global value
    Active worker threads: 0
    Basic authentication mode: filter
    Forms based SSO: disabled
    Authentication HTTP header: insert - iv_user
    Remote Address HTTP header: do not insert
    Stateful junction: no
    Boolean Rule Header: no
    Scripting support: yes
    Preserve cookie names: no
    Delegation support: no
    Mutually authenticated using Basic Authentication: yes
        WebSEAL Username: wasadmin
        Password: lnx4ltic
    Insert WebSphere LTPA cookies: no
    Insert WebSEAL session cookies: no
    Request Encoding: UTF-8, URI Encoded
    Server 1:
        ID: f500cef8-c17d-11d9-8377-02000000001b
```

```
        Server State: running
        Proxy Hostname: litcp01.ltic.pok.ibm.com
        Proxy Port: 80
        Hostname: litwasclx.ltic.pok.ibm.com
        Port: 443
        Virtual hostname: litwasclx.ltic.pok.ibm.com
        Server DN:
        Query_contents URL: /cgi-bin/query_contents
        Query-contents: unknown
        Case insensitive URLs: no
        Allow Windows-style URLs: yes
        Total requests : 1
pdadmin sec_master> exit
```

Installing and migrating TAM WebSEAL 5.1.13 on the new system

We built a new 64bit 2.6 SUSE LINUX Enterprise Linux 9 System - littam02.ltic.pok.ibm.com. As mentioned earlier, we reference this new system as <new-littam02>. We used the same hostname since the WebSEAL configuration information is gathered from /etc/hosts during the pdconfig process.

We installed GSKit 7.0.3.9:

```
<new-littam02>:~/TAM # rpm -Uvh gsk7bas-7.0-3.9.s390.rpm
Preparing...                ########################### [100%]
1:gsk7bas                   ########################### [100%]
```

We installed the LDAP client:

```
<new-littam02>:~/TAM # rpm -ivh ldap-clientd-5.2-1.s390.rpm
Preparing...                ########################### [100%]
1:ldap-clientd              ########################### [100%]
```

We installed PDRTE, PDWebRTE, and PDWeb, v5.1:

```
<new-littam02>:/mnt/zSeries # rpm -Uvh PDRTE-PD-5.1.0-0.s390.rpm
Preparing...                ############################ [100%]
adding ivmgr user
1:PDRTE-PD                  ############################ [100%]
<new-littam02>:/mnt/zSeries # rpm -Uvh PDWebRTE-PD-5.1.0-0.s390.rpm
Preparing...                ############################ [100%]
1:PDWebRTE-PD               ############################ [100%]
<new-littam02>:/mnt/zSeries # rpm -Uvh PDWeb-PD-5.1.0-0.s390.rpm
Preparing...                ############################ [100%]
1:PDWeb-PD                  ############################ [100%]
```

We installed Tivoli WebSEAL Fixpack 13:

```
<new-littam02>:~/TAM # rpm -Uvh PDRTE-PD-5.1.0-13.s390.rpm
Preparing...                ############################ [100%]
1:PDRTE-PD                  ############################ [100%]
<new-littam02>:~/TAM # rpm -Uvh PDWebRTE-PD-5.1.0-13.s390.rpm
Preparing...                ############################ [100%]
1:PDWebRTE-PD               ############################ [100%]
<new-littam02>:~/TAM # rpm -Uvh PDWeb-PD-5.1.0-13.s390.rpm
Preparing...                ############################ [100%]
1:PDWeb-PD                  ############################ [100%]
```

We verified the version:

```
<new-littam02>:~/TAM # pdversion
IBM Tivoli Access Manager Runtime 5.1.0.13
IBM Tivoli Access Manager Policy Server Not Installed
IBM Tivoli Access Manager Web Portal Manager Not Installed
IBM Tivoli Access Manager Application Developer Kit Not Installed
IBM Tivoli Access Manager Authorization Server Not Installed
```

```
IBM Tivoli Access Manager Java Runtime Environment Not Installed
IBM Tivoli Access Manager Policy Proxy Server Not Installed
IBM Tivoli Access Manager WebSEAL Server 5.1.0.13
```

We transferred the LDAP key database to the new WebSEAL system to preserve SSL communication with LDAP Server:

```
<old-littam02>:/usr/ldap/etc # scp key_ldap.kdb <new>:/usr/ldap/etc/
Password:
key_ldap.kdb         100% |********************| 55080 00:00
```

We transferred the tar file backups of PD Runtime and WebSEAL from the original WebSEAL image to the new WebSEAL image:

```
<old-littam02>:/var/PolicyDirector/pdbackup # scp \
pdbackup.lst_21Feb2006.19_14.tar <new-littam02>:/var/PolicyDirector/pdbackup
Password:
pdbackup.lst_21Feb2006.18_12.tar              100%  200KB 200.0KB/s   00:01

<old-littam02>:/var/PolicyDirector/pdbackup # scp \
amwebbackup.lst_21Feb2006.19_17.tar <new
littam02>:/var/PolicyDirector/pdbackup
Password:
amwebbackup.lst_21Feb2006.18_14.tar           100% 8060KB   1.6MB/s   00:05
```

On the new WebSEAL image, we ran the pdbackup command with the restore option to restore the PD Runtime and WebSEAL configuration:

```
<new-littam02>:/opt/PolicyDirector/bin # ./pdbackup -a restore -file /root/ \ pdbackup.lst_21Feb2006.18_12.tar

<new-littam02>:/opt/PolicyDirector/bin # ./pdbackup -a restore -file /root/ \ amwebbackup.lst_21Feb2006.18_14.tar
```

We verified the return code in /tmp/msg__pdbackup.log completed with a return code of 0 for no errors.

We checked to see if WebSEAL is configured correctly:

```
<new-littam02>:~/PDWEB5.1.13 # pd_start status

Access Manager Servers

Server                    Enabled   Running

------------------------------------------

pdmgrd                    no        no
pdacld                    no        no
pdmgrproxyd               no        no
webseald-WebSeal1         yes       yes
```

We verified functionality by checking on the junction created on the old WebSEAL is now visible from the new WebSEAL server:

```
pdadmin sec_master> s t WebSeal1-webseald-littam02 show /was
    Junction point: /was
    Type: SSL Proxy
    Junction hard limit: 0 - using global value
    Junction soft limit: 0 - using global value
    Active worker threads: 0
    Basic authentication mode: filter
    Forms based SSO: disabled
    Authentication HTTP header: insert - iv_user
    Remote Address HTTP header: do not insert
    Stateful junction: no
    Boolean Rule Header: no
    Scripting support: yes
    Preserve cookie names: no
    Delegation support: no
```

```
Mutually authenticated using Basic Authentication: yes
    WebSEAL Username: wasadmin
    Password: lnx4ltic
Insert WebSphere LTPA cookies: no
Insert WebSEAL session cookies: no
Request Encoding: UTF-8, URI Encoded
Server 1:
    ID: f483a4bc-5239-11da-82bb-02000000001c
    Server State: running
    Proxy Hostname: litcp01.ltic.pok.ibm.com
    Proxy Port: 80
    Hostname: 192.168.71.98
    Port: 443
    Virtual hostname: 192.168.71.98
    Server DN:
    Query_contents URL: /cgi-bin/query_contents
    Query-contents: unknown
    Case insensitive URLs: no
    Allow Windows-style URLs: yes
    Total requests : 1065
```

Now that we verified the new WebSEAL server(SLES 9 64bit) is configured correctly and operational, we retired the original WebSEAL Server(SLES 8 31bit). We took a short outage to our production traffic and shutdown the original littam02 guest on VM and put the new WebSEAL littam02 into production. We used the original IP address for the new WebSEAL image. We didn't need to modify the hostname since both images shared the same name – **littam02.ltic.pok.ibm.com.**

We completed migrating and transitioning WebSEAL, we now have TAM v5.1.13 WebSEAL Server running in 31-bit compatibility mode on a 64bit 2.6 SuSE LINUX Enterprise Server 9.

Migrating WebSphere Application Server Network Deployment Cell from V5.1.1.x to V6.0.2.x

We migrated our WebSphere Application Server Network Deployment Cell from V5.1.1.4 to V6.0.2.1 in order to use the HAManager feature for the High Availability architectures testing (see Section "Implementing WebSphere Application Server HAManager"). The HAManager feature comes with base WebSphere Application Server V6 and above.

For this migration, we followed the migration instructions from the WebSphere Application Server V6 Info Center which can be accessed here:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp`

A really good presentation on migration can be found here:

`ftp://ftp.software.ibm.com/software/eod/was/6.0/Install_Migration/WASv6_Migration/playershell.swf`

The presentation goes over migration of a standalone WebSphere Application Server, migration of a Network Deployment cell, migration commands, migration of cell Web server plug-in module, mixed node environments limitations, and tools and application migration. It is a good starting point if you are thinking about migrating your WebSphere Application Server environment to V6.

We will go over some observations and road bumps we hit during our migration from a V5.1.1.4 Network Deployment cell to V6.0.2.1.

We migrated our cell in the following order:

1. WebSphere Application Server Network Deployment node
2. WebSphere Application Server server plug-in
3. WebSphere Application Server node 1, 2, and 3

We were first going to migrate to V6.0.0.1, but during the process of migrating the Network Deployment node, we hit errors during the WASPostUpgrade.sh process. We contacted support and was told to upgrade to the then latest level, V6.0.2.1, before attempting migration. Here are the recommendations from support and the procedures that we followed to migrate WebSphere Application Server Network Deployment node (we used the same procedures to migrate the nodes).

For best results in migration, it is strongly recommended you upgrade WebSphere V6.0.0.1 to the latest build, currently V6.0.2.1, before attempting migration. There are numerous updates to the migration code that may resolve the initial problem you are experiencing. We would suggest you remove the existing Dmgr profile in V6, install Refresh Pack 2 and Fix Pack 1 and then create the Dmgr profile again. Backup the configurations in WebSphere V5 and V6 and then try the migration of the Deployment Manager again. The next section has links to the InfoCenter documentation. The suggested steps to take are as follows:

1. At this point we had V6.0.0.1 installed so we just uninstalled it. The V5.1.1.4 server had to be stopped during the uninstall. While the Network Deployment manager was stopped for migration, the Application Servers were still running on the other nodes and serving our application.
2. We reinstalled WebSphere Application Server ND V6. We did not create a profile at this point. During the install we saw that V6 recognized that we already had another WebSphere Application Server running and that we might be migrating it later on, as seen in Figure 80 on page 317.

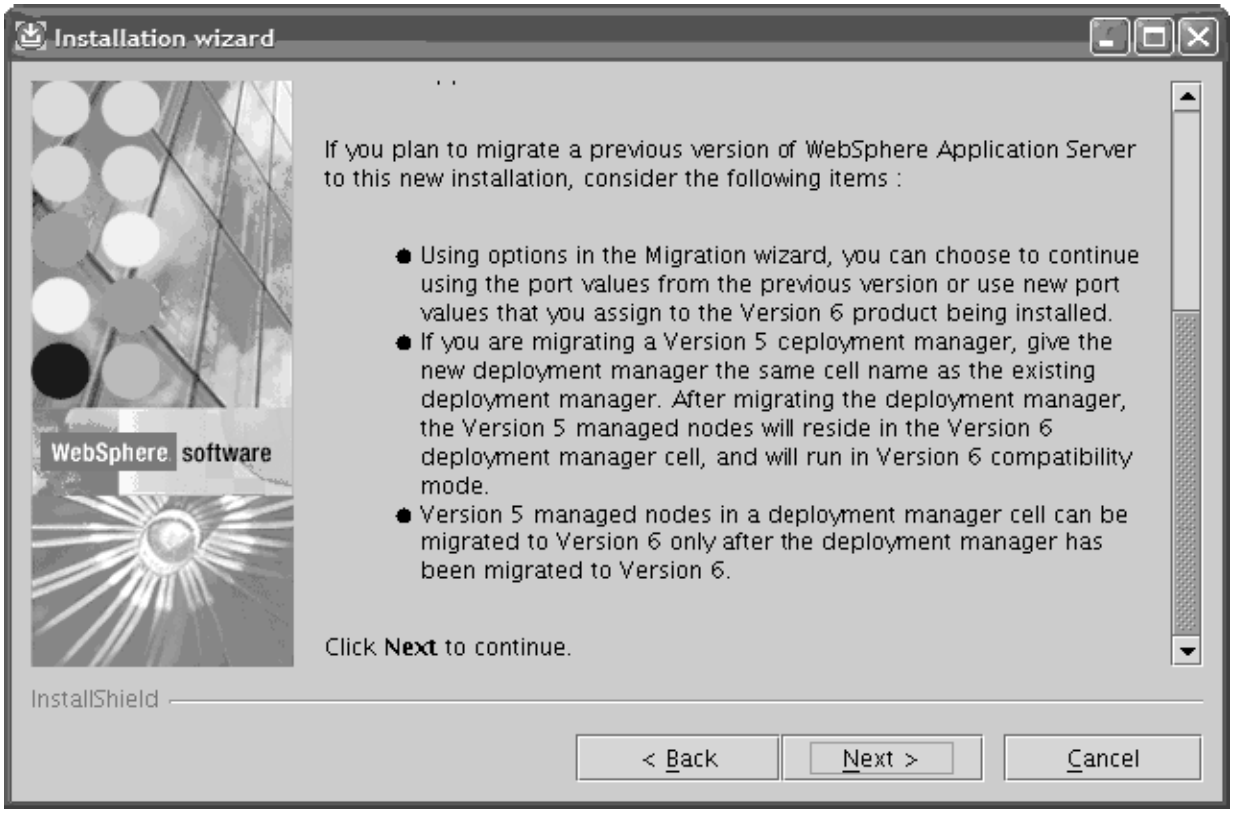


Figure 80. Installation wizard.

3. As actual root user, we downloaded Refresh Pack 2 into <WAS_HOME> directory: litwas04:/opt/IBM/WebSphere/AppServer/. Unpacked the file and it created the litwas04:/opt/IBM/WebSphere/AppServer/updateinstaller directory with several subfolders.
4. As actual root user, downloaded Fix Pack 1 into <WAS_HOME> directory: litwas04:/opt/IBM/WebSphere/AppServer/. Unpacked the file and it unpacked the files into the existing litwas04:/opt/IBM/WebSphere/AppServer/ updateinstaller directory and updated the updateinstaller code at the same time.
5. Updateinstaller now reflected the version of the Fix Pack and build date in the <WAS_HOME>/updateins5) Updateinstaller now reflected the version of the Fix Pack and build date in the <WAS_HOME>/updateinstaller/version.txt file. If for some reason you cannot go to the V6.0.2.1 level but must stay at V6.0.2, download the latest build of the updateinstaller as above and unpack it after you have unpacked Refresh Pack 2.taller/version.txt file. If for some reason you cannot go to the V6.0.2.1 level but must stay at V6.0.2, download the latest build of the updateinstaller as above and unpack it after you have unpacked Refresh Pack 2.
6. We verified that all Java processes were stopped and installed Refresh Pack 2 then Fix Pack 1. Instructions are in the links referenced below. We did not create a profile at this point.
7. We verified installation/upgrade is successful, then created Dmgr profile as instructed using the same naming structure as the node being migrated per the migration documentation. We used the profile creation wizard that comes with V6. Since V5 didn't have the concept of a profile, the profile name can be whatever you want. We chose Dmgr. However, the node, host and cell names

that you define on v6 must match the node being migrated. See Figure 81, Figure 82 on page 319, and Figure 83 on page 320.

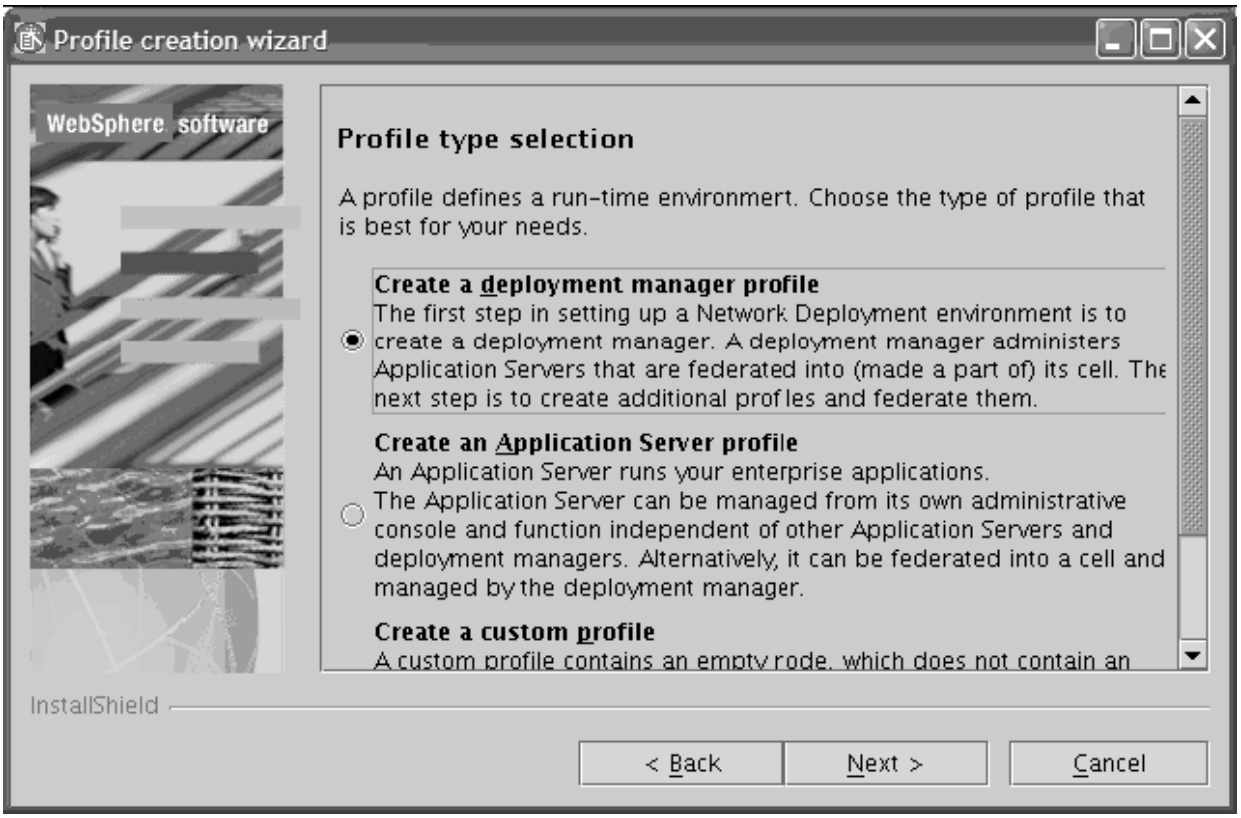


Figure 81. Profile type selection.

Figure 82. Profile name.

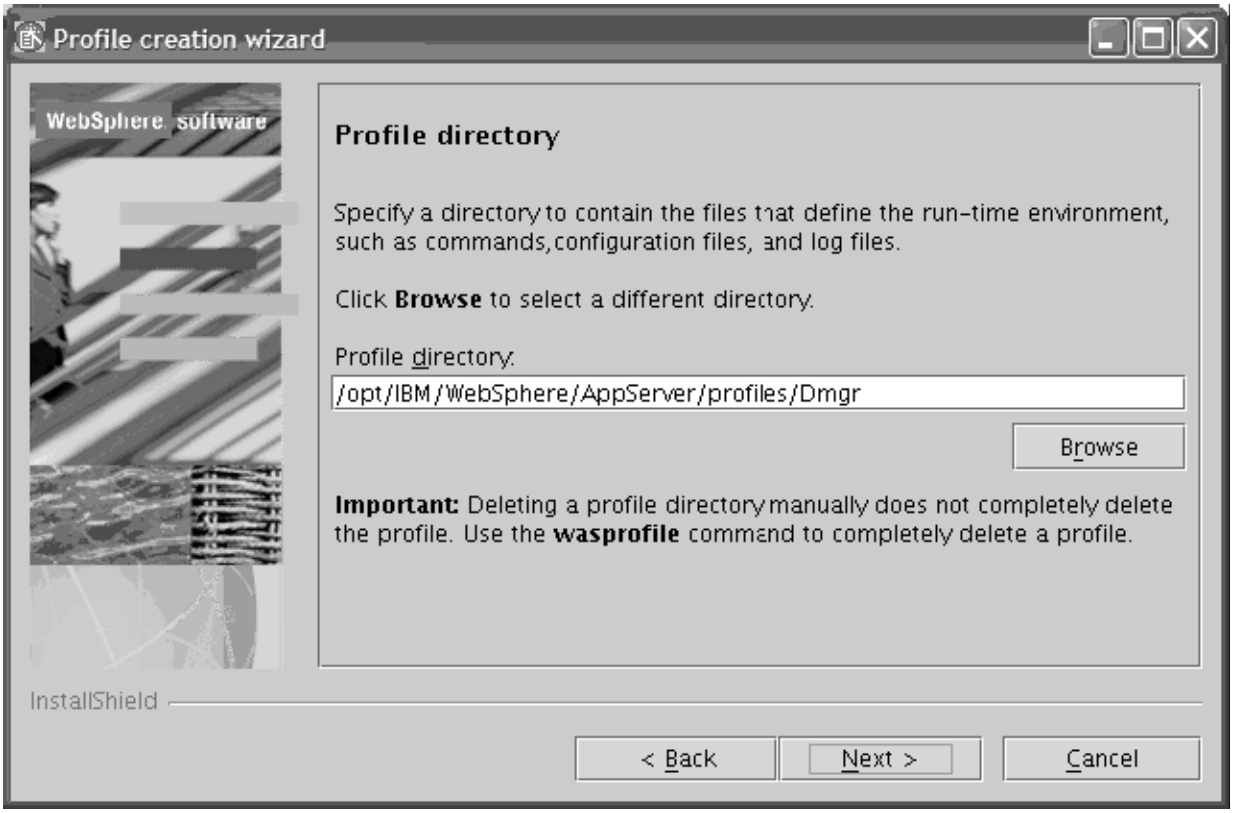


Figure 83. Profile directory.

8. We migrated the Dmgr using the Migration wizard as per the instructions provided in the info center. The nice thing about the wizard is that it will backup the current configurations and perform all the steps necessary to migrate your applications.
9. After the migration of the Network Deployment node, we brought it up to see that it still was able to manage all the V5.1.1.4 WebSphere Application Server nodes.
10. We followed the same procedure for migrating the Network Deployment node to migrate our WebSphere Application Server nodes, being careful about keeping the same node, host and cell names as the node being migrated. We migrated these one at a time while the rest were running and serving our application.
11. After all the nodes were migrated. We performed a final test to see that everything worked and our application was in good shape.

Helpful links during migration

The following links were very helpful during our migration:

- Deleting a profile:

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tins_removeprofile.html`
- IBM - Recommended Updates for WebSphere Application Server Base and Network Deployment editions:

  `http://www.ibm.com/support/docview.wss?rs=180&uid=swg27004980`
- IBM - 6.0.2: WebSphere Application Server V6.0 Refresh Pack 2 for Linux platforms:

http://www.ibm.com/support/docview.wss?rs=180&uid=swg24010068

- IBM - Installing V6.0.2 WebSphere Application Server on Linux systems:

  http://www.ibm.com/support/docview.wss?rs=180&context=SSEQTP&uid=swg21210058
- IBM - 6.0.2.1: WebSphere Application Server V6.0.2 Fix Pack 1 for Linux platforms:

  http://www.ibm.com/support/docview.wss?rs=180&uid=swg24010302
- IBM - Readme for IBM WebSphere Application Server V6.0.2.1:

  http://www.ibm.com/support/docview.wss?rs=180&uid=swg27006568
- IBM - Update Installer for WebSphere Application Server V6.0 releases:

  http://www.ibm.com/support/docview.wss?rs=180&uid=swg24008401
- Using the Profile creation wizard:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tins_instances.html
- Migrating from Network Deployment Version 5.x to a Version 6.0.x deployment manager:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tins_migratend50x.html
- WASPreUpgrade command:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/rins_WASPreUpgrade
- WASPostUpgrade command:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/rins_WASPostUpgrade.html
- Migrating a Version 5.x deployment manager to Version 6.0.x with the Migration wizard:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tins_helpmigw5dm.html

Chapter 21. Installing and configuring WebSphere Portal Server Cluster

WebSphere Portal Server cluster installation and configuration can take significant time, especially if you have many servers in the cluster and you have to install a full WPS on each of them. Wouldn't it be easier to install it once and save time by cloning the rest? Together with the zSeries New Technology Center we wrote a Redbook that documents the installation and cloning procedures which can be found at: www.redbooks.ibm.com/redpapers/abstracts/redp4175.html

Chapter 22. Linux and z/VM system programmer tips

Increasing the root filesystem size on production Linux system

The following is a simple procedure to increase the size of a root files system on CKD devices without LVM.

Problem

The Linux system was set up on 3390 Mod 3's and is nearly out of free space on the root file system. The system is NOT running with LVM so there is no way to increase the size of the root file system.

Solution

Transfer the system to a larger CKD device using tar with piping.

Assumptions

The following are the assumptions:

1. The system where the file system is expanded is SYSTEMA.
2. You have a second Linux system running the same level of Linux as the system you need to expand the file system on. We will refer to this as SYSTEMB in the example.

   Note: It is possible to run this procedure on a single Linux VM guest if you DEF 201 as 401 (SYSTEMA) and then IPL the SYSTEMB Linux system on a 201 pack. For simplicity we use two different Linux VM Guests in this example.
3. You have CKD devices larger the 3390 Mod 3's available.
4. SYSTEMA is shutdown.

Procedure

The following is the procedure:

1. Identity the DASD to be migrated.

   Current System disks:

   ```
   SYSTEMA  Device: 0201   File system:  /dev/dasda1   3338 Cylinders
   ```
2. Identify the Target DASD:

   ```
   SYSTEMA Device: 0301 No File System installed yet.
   ```
3. Bring up a Linux system on SYSTEMB.

   This system should be at the same level as the system you are migrating.
4. Attach or LINK both the Source and Target DASD to the transfer system.

   In this case I have linked the source volumes RR and the Target DASD R/W.

   To keep things simple I have come up with the following naming conventions:

   ```
   Source 201 disk  -   linked R/O as 401
   Target 301 disk  -    linked R/W as 301
   ```
5. Bring the volumes online on the transfer system:

   ```
   systemb:~ # echo 1 > /sys/bus/ccw/drivers/0.0.0301/online
   systemb:~ # echo 1 > /sys/bus/ccw/drivers/0.0.0401/online
   ```
6. Find out what /dev/dasdx the volumes are:

```
systemb:~ # proc /dev/dasd/devices

systemb:~ # cat /proc/dasd/devices
0.0.0200(FBA ) at ( 94:     0) is dasda
0.0.0201(ECKD) at ( 94:     4) is dasdb
0.0.0301(ECKD) at ( 94:     8) is dasdc
0.0.0401(ECKD) at ( 94:    12) is dasdd
```

Note: Truncated for readability.

7. DASDFMT, FDISK and install a filesystem on the target system:

```
systemb:~ # dasdfmt -b 4096 -f /dev/dasdc -y -p
systemb:~ # fdasd /dev/dasdc
systemb:~ # mkfs.ext3 -b 4096 /dev/dasdc1
```

8. Create mount points for the source and destination DASD.

   To make things easy, I created the mount points with the same name as the device number:

```
systemb:~ # mkdir /301
systemb:~ # mkdir /401
```

9. Mount the source and target filesystems:

```
systemb:~ # mount /dev/dasdc1 /301
systemb:~ # mount /dev/dasdd1 /401
```

10. Use Tar to move from the old volume to the new volumes.

    Make sure you issue this command from the source directory mount point. In this case we are in /401 when we issue the tar:

```
systemb:/401 # tar -cvpf - ./* | tar -xpf - -C /301

   Input Options:
    -c   Create
    -v Verbose - list all files processed
    -p Preserve Permission's/owner/group
    -f  To archive file  (-) dash in this case to feed the pipe

   Output Options:
    -x   Extract
    -p Preserve Permission's/owner/group
    -f  From archive file  (-) dash - the pipe
    -C The output directory
```

11. We now have copied all of the data from the Mod 3's copied to the Mod 9's. However the system will not boot. We need to run mkinitrd and zipl for the target directory /boot. We will do this with the chroot command from SYSTEMB:

```
systemb:~ # chroot /301 /bin/bash
systemb:~ # mkinitrd
systemb:~ # zipl
```

12. To prevent the system from hanging when you IPL - Make a copy of /etc/fstab to /etc/ffstab.orig, then remove all but the root file system from /etc/fstab. We will copy back the original /etc/fstab after we IPL the system.
13. At this point we are ready to shutdown SYSTEMB and IPL the SYSTEMA.
14. IPL the system and fix fstab.
    a. Make sure you have all of you volumes in /proc/dasd/devices that you expected. If not, bring online like we did in step 5.
    b. Once all of your DASD is back online and in the correct order - copy fstab.orig over fstab.
    c. Before we reboot - verify that the mount works correctly:

```
# mount -a
```

    d. If all the volume are online and in the correct place - reboot the system.

15. Your migration is now complete.

Chapter 23. Future Linux on zSeries projects

Following are some areas of future testing for the Linux on zSeries team.

Disaster Recovery

We will be doing a disaster recovery follow-on to the High Availability Architectures test that expands to include testing of other RAS/BR (Reliability, Availability, Serviceability/Business Resilience) characteristics through a variety of attacks against different elements of the infrastructure to assess the zLVS PET environment's overall robustness and identify strengths and weaknesses. We will create and test a "recovery matrix" that includes ways of injecting failures into various components in the environment, expected error messages, and what to expect in terms of recovery.

Where to find more information

During our testing, we used documentation from several sources, listed below. They contain all of the documents that we have cited throughout the course of this chapter.

- IBM TechDocs (flashes, white papers, etc.), available at www.ibm.com/support/techdocs/
- IBM WebSphere Application Server documentation, available at http://www.ibm.com/software/webservers/appserv/zos_os390/library/
- IBM WebSphere Studio Workload Simulator documentation, available at http://www.ibm.com/software/webservers/appserv/was/library/
- *Java Servlet Specification, v2.2*, available at http://java.sun.com/products/servlet/
- *Java 2 Platform Enterprise Edition Specification, v1.2*, available at http://java.sun.com/products/j2ee/
- *JavaServer Pages Specification, Version 1.1*, available at http://java.sun.com/products/jsp/
- *J2EE Connector Architecture*, available at http://java.sun.com/j2ee/connector/
- Tivoli Access Manager Info Center, available at:

  `http://publib.boulder.ibm.com/infocenter/tiv2help/index.jsp?toc=/com.ibm.itame.doc_5.1/toc.xml`
- Exploring Open Source Security for a Linux Server Environment available at

  `ftp://ftp.software.ibm.com/eserver/zseries/misc/literature/pdf/whitepapers/gm130636.pdf`
- DB2 Info Center located at

  `http://publib.boulder.ibm.com/infocenter/db2help/index.jsp?topic=/com.ibm.db2.udb.doc/ad/c0006975.htm`
- IBM WebSphere Application Server Network Deployment Edge Components Info Center located at

  `http://www-306.ibm.com/software/webservers/appserv/doc/v51/ec/infocenter/index.html`
- The High-Availability Linux project at:

  `http://linux-ha.org/`

Appendix A. Some of our parmlib members

This section describes how we have set up some of our parmlib members for z/OS. Table 18 summarizes our new and changed parmlib members for z/OS V1R5 and z/OS.e V1R5. Samples of some of our parmlib members are available on the Samples page of our Web site.

Table 18. Summary of our parmlib changes for z/OS V1R5 and z/OS.e V1R5

Member name	z/OS release	Change summary	Related to
IFAPRD*xx*	z/OS.e V1R5	No changes from z/OS.e V1R3. (See our December 2002 edition.)	z/OS.e, dynamic enablement
IEASYMPT	z/OS.e V1R5	No changes from z/OS.e V1R3. (See our December 2002 edition.)	z/OS.e, dynamic enablement
IEASYS*xx*	z/OS.e V1R5	No changes from z/OS.e V1R3. (See our December 2002 edition.)	z/OS.e
LOAD*xx* (Actually in SYS0.IPLPARM. See note below.)	z/OS V1R5	Changed the PARMLIB statement to use SYS1.PETR15.PARMLIB	concatenated parmlib
	z/OS.e V1R5	No changes from z/OS.e V1R3. (See our December 2002 edition.)	z/OS.e
LPALST*xx*	z/OS V1R5	Added: SYS1.SIATLPA	JES3
PROG*xx* (APF additions)	z/OS V1R5	Added: SYS1.SHASLINK SYS1.SHASMIG	JES2
PROG*yy* (LNKLST)	z/OS V1R5	Added to LNKLST*xx*: SYS1.SHASLINK SYS1.SHASMIG	JES2
		Added to LNKLST*xx*: SYS1.SIATLIB SYS1.SIATLINK SYS1.SIATMIG	JES3

Note: As of our OS/390 R6 testing, we changed LOAD*xx* to use a generic name, IEASYMPT, for our IEASYM*xx* member. We have successfully used the name IEASYMPT for our migrations through all subsequent releases of OS/390 and z/OS. Only the entries in SYS0.IPLPARM changed.

Appendix B. Some of our RMF reports

In this appendix we include some of our RMF reports, as indicated in “z/OS performance” on page 48.

RMF Monitor I post processor summary report

The following contains information from our *RMF Monitor I Post Processor Summary Report*. Some of the information we focus on in this report includes CP (CPU) busy percentages and I/O (DASD) rates.

```
1                                       R M F   S U M M A R Y   R E P O R T
                                                                                PAGE  001
            z/OS  V1R7              SYSTEM ID JA0            START 09/15/2005-11.45.00  INTERVAL 00.15.00
                                    RPT VERSION V1R7 RMF     END   09/15/2005-12.00.00  CYCLE 0.100 SECONDS
0
 NUMBER OF INTERVALS 1
-DATE   TIME     INT   CPU   DASD  DASD  TAPE   JOB   JOB   TSO   TSO   STC   STC  ASCH  ASCH  OMVS  OMVS SWAP DEMAND
 MM/DD HH.MM.SS MM.SS  BUSY  RESP  RATE  RATE   MAX   AVE   MAX   AVE   MAX   AVE   MAX   AVE   MAX   AVE RATE PAGING
009/15 11.45.00 15.00  32.3   1.2  2215   0.0     7     3    21    21   380   376     1     0    43    35 0.00   0.00
1                                       R M F   S U M M A R Y   R E P O R T
                                                                                PAGE  001
            z/OS  V1R7              SYSTEM ID JB0            START 09/15/2005-11.45.00  INTERVAL 00.15.00
                                    RPT VERSION V1R7 RMF     END   09/15/2005-12.00.00  CYCLE 0.100 SECONDS
0
 NUMBER OF INTERVALS 1
-DATE   TIME     INT   CPU   DASD  DASD  TAPE   JOB   JOB   TSO   TSO   STC   STC  ASCH  ASCH  OMVS  OMVS SWAP DEMAND
 MM/DD HH.MM.SS MM.SS  BUSY  RESP  RATE  RATE   MAX   AVE   MAX   AVE   MAX   AVE   MAX   AVE   MAX   AVE RATE PAGING
009/15 11.45.00 15.00  16.9   0.6  5243 549.0    13    10    74    72   646   643     1     0    27    23 0.00   0.00
1                                       R M F   S U M M A R Y   R E P O R T
                                                                                PAGE  001
            z/OS  V1R7              SYSTEM ID JC0            START 09/15/2005-11.45.00  INTERVAL 00.15.00
                                    RPT VERSION V1R7 RMF     END   09/15/2005-12.00.00  CYCLE 0.100 SECONDS
0
 NUMBER OF INTERVALS 1
-DATE   TIME     INT   CPU   DASD  DASD  TAPE   JOB   JOB   TSO   TSO   STC   STC  ASCH  ASCH  OMVS  OMVS SWAP DEMAND
 MM/DD HH.MM.SS MM.SS  BUSY  RESP  RATE  RATE   MAX   AVE   MAX   AVE   MAX   AVE   MAX   AVE   MAX   AVE RATE PAGING
009/15 11.45.00 15.00  12.3   1.5 194.5   0.0    12     9    27    27   385   379     1     0    38    29 0.00   0.00
1                                       R M F   S U M M A R Y   R E P O R T
                                                                                PAGE  001
            z/OS  V1R7              SYSTEM ID JE0            START 09/15/2005-11.45.00  INTERVAL 00.15.00
                                    RPT VERSION V1R7 RMF     END   09/15/2005-12.00.00  CYCLE 0.100 SECONDS
0
 NUMBER OF INTERVALS 1
-DATE   TIME     INT   CPU   DASD  DASD  TAPE   JOB   JOB   TSO   TSO   STC   STC  ASCH  ASCH  OMVS  OMVS SWAP DEMAND
 MM/DD HH.MM.SS MM.SS  BUSY  RESP  RATE  RATE   MAX   AVE   MAX   AVE   MAX   AVE   MAX   AVE   MAX   AVE RATE PAGING
009/15 11.45.00 15.00   8.4   1.1  1246   0.0     1     0     0     0   363   362     1     0    24    18 0.00   0.00
1                                       R M F   S U M M A R Y   R E P O R T
                                                                                PAGE  001
            z/OS  V1R7              SYSTEM ID JF0            START 09/15/2005-11.45.00  INTERVAL 00.15.00
                                    RPT VERSION V1R7 RMF     END   09/15/2005-12.00.00  CYCLE 0.100 SECONDS
0
 NUMBER OF INTERVALS 1
-DATE   TIME     INT   CPU   DASD  DASD  TAPE   JOB   JOB   TSO   TSO   STC   STC  ASCH  ASCH  OMVS  OMVS SWAP DEMAND
 MM/DD HH.MM.SS MM.SS  BUSY  RESP  RATE  RATE   MAX   AVE   MAX   AVE   MAX   AVE   MAX   AVE   MAX   AVE RATE PAGING
009/15 11.45.00 15.00   6.0   5.9 202.5   0.0     0     0     0     0   382   381     1     0    25    20 0.00   0.00
1                                       R M F   S U M M A R Y   R E P O R T
                                                                                PAGE  001
            z/OS  V1R7              SYSTEM ID JH0            START 09/15/2005-11.45.00  INTERVAL 00.15.00
                                    RPT VERSION V1R7 RMF     END   09/15/2005-12.00.00  CYCLE 0.100 SECONDS
0
 NUMBER OF INTERVALS 1
-DATE   TIME     INT   CPU   DASD  DASD  TAPE   JOB   JOB   TSO   TSO   STC   STC  ASCH  ASCH  OMVS  OMVS SWAP DEMAND
 MM/DD HH.MM.SS MM.SS  BUSY  RESP  RATE  RATE   MAX   AVE   MAX   AVE   MAX   AVE   MAX   AVE   MAX   AVE RATE PAGING
009/15 11.45.00 15.00  11.7   1.1 159.1   0.0     3     1     0     0   370   368     1     0    30    24 0.00   0.00
1                                       R M F   S U M M A R Y   R E P O R T
                                                                                PAGE  001
            z/OS  V1R7              SYSTEM ID J80            START 09/15/2005-11.45.00  INTERVAL 00.15.00
                                    RPT VERSION V1R7 RMF     END   09/15/2005-12.00.00  CYCLE 0.100 SECONDS
0
 NUMBER OF INTERVALS 1
-DATE   TIME     INT   CPU   DASD  DASD  TAPE   JOB   JOB   TSO   TSO   STC   STC  ASCH  ASCH  OMVS  OMVS SWAP DEMAND
 MM/DD HH.MM.SS MM.SS  BUSY  RESP  RATE  RATE   MAX   AVE   MAX   AVE   MAX   AVE   MAX   AVE   MAX   AVE RATE PAGING
009/15 11.45.00 15.00  86.1   3.9  4977   0.0   464   464   394   394   504   423     3     0   339   261 0.00   0.00
1                                       R M F   S U M M A R Y   R E P O R T
                                                                                PAGE  001
            z/OS  V1R7              SYSTEM ID J90            START 09/15/2005-11.45.00  INTERVAL 00.15.00
                                    RPT VERSION V1R7 RMF     END   09/15/2005-12.00.00  CYCLE 0.100 SECONDS
```

```
0
 NUMBER OF INTERVALS 1
-DATE   TIME     INT   CPU   DASD  DASD  TAPE   JOB   JOB   TSO   TSO   STC   STC  ASCH  ASCH  OMVS  OMVS SWAP DEMAND
 MM/DD HH.MM.SS MM.SS  BUSY  RESP  RATE  RATE   MAX   AVE   MAX   AVE   MAX   AVE   MAX   AVE   MAX   AVE RATE PAGING
009/15 11.45.00 15.00  13.9   1.3 446.2   0.0     3     2    12    12   377   370     1     0    29    21 0.00   0.00
1                                        R M F   S U M M A R Y   R E P O R T                                    PAGE  001
             z/OS  V1R7              SYSTEM ID Z0              START 09/15/2005-11.45.00  INTERVAL 00.15.00
                                     RPT VERSION V1R7 RMF      END   09/15/2005-12.00.01  CYCLE 0.100 SECONDS
0
 NUMBER OF INTERVALS 1
-DATE   TIME     INT   CPU   DASD  DASD  TAPE   JOB   JOB   TSO   TSO   STC   STC  ASCH  ASCH  OMVS  OMVS SWAP DEMAND
 MM/DD HH.MM.SS MM.SS  BUSY  RESP  RATE  RATE   MAX   AVE   MAX   AVE   MAX   AVE   MAX   AVE   MAX   AVE RATE PAGING
009/15 11.45.00 15.00  14.1   3.5  84.6   0.0     1     0     1     1   380   377     0     0    36    32 0.00   0.00
1                                        R M F   S U M M A R Y   R E P O R T                                    PAGE  001
             z/OS  V1R7              SYSTEM ID Z1              START 09/15/2005-11.45.00  INTERVAL 00.15.00
                                     RPT VERSION V1R7 RMF      END   09/15/2005-12.00.00  CYCLE 0.100 SECONDS
0
 NUMBER OF INTERVALS 1
-DATE   TIME     INT   CPU   DASD  DASD  TAPE   JOB   JOB   TSO   TSO   STC   STC  ASCH  ASCH  OMVS  OMVS SWAP DEMAND
 MM/DD HH.MM.SS MM.SS  BUSY  RESP  RATE  RATE   MAX   AVE   MAX   AVE   MAX   AVE   MAX   AVE   MAX   AVE RATE PAGING
009/15 11.45.00 15.00  23.3   3.4  1417   0.0     0     0     4     4   373   370     1     0    17    12 0.00   0.02
1                                        R M F   S U M M A R Y   R E P O R T                                    PAGE  001
             z/OS  V1R7              SYSTEM ID Z2              START 09/15/2005-11.45.00  INTERVAL 00.14.59
                                     RPT VERSION V1R7 RMF      END   09/15/2005-12.00.00  CYCLE 0.100 SECONDS
0
 NUMBER OF INTERVALS 1
-DATE   TIME     INT   CPU   DASD  DASD  TAPE   JOB   JOB   TSO   TSO   STC   STC  ASCH  ASCH  OMVS  OMVS SWAP DEMAND
 MM/DD HH.MM.SS MM.SS  BUSY  RESP  RATE  RATE   MAX   AVE   MAX   AVE   MAX   AVE   MAX   AVE   MAX   AVE RATE PAGING
009/15 11.45.00 14.59   3.0   1.6  62.9   0.0    61    60     1     1   365   364     0     0     8     8 0.00   0.00
1                                        R M F   S U M M A R Y   R E P O R T                                    PAGE  001
             z/OS  V1R7              SYSTEM ID Z3              START 09/15/2005-11.45.00  INTERVAL 00.15.00
                                     RPT VERSION V1R7 RMF      END   09/15/2005-12.00.00  CYCLE 0.100 SECONDS
0
 NUMBER OF INTERVALS 1
-DATE   TIME     INT   CPU   DASD  DASD  TAPE   JOB   JOB   TSO   TSO   STC   STC  ASCH  ASCH  OMVS  OMVS SWAP DEMAND
 MM/DD HH.MM.SS MM.SS  BUSY  RESP  RATE  RATE   MAX   AVE   MAX   AVE   MAX   AVE   MAX   AVE   MAX   AVE RATE PAGING
009/15 11.45.00 15.00  23.5   2.0 301.2   0.0   122   121     2     2   396   391     1     0    17     9 0.00   0.00
1                                        R M F   S U M M A R Y   R E P O R T                                    PAGE  001
             z/OS  V1R7              SYSTEM ID TPN             START 09/15/2005-11.45.00  INTERVAL 00.15.00
                                     RPT VERSION V1R7 RMF      END   09/15/2005-12.00.00  CYCLE 0.100 SECONDS
0
 NUMBER OF INTERVALS 1
-DATE   TIME     INT   CPU   DASD  DASD  TAPE   JOB   JOB   TSO   TSO   STC   STC  ASCH  ASCH  OMVS  OMVS SWAP DEMAND
 MM/DD HH.MM.SS MM.SS  BUSY  RESP  RATE  RATE   MAX   AVE   MAX   AVE   MAX   AVE   MAX   AVE   MAX   AVE RATE PAGING
009/15 11.45.00 15.00  43.4   2.7 154.2   0.0     5     4     5     5   355   354     0     0     4     4 0.00   0.09
```

RMF Monitor III online sysplex summary report

The following contains information from the RMF *Monitor III Online Sysplex Summary Report*. This is a real-time report available if you are running WLM in goal mode. We highlighted some of our goals and actuals for various service classes and workloads. At the time this report was captured we were running 2784 CICS transactions/second.

```
WLM Samples: 240     Systems: 13 Date: mm/dd/yy Time: xx.xx.xx Range: 60    Sec

                      >>>>>>>>XXXXXXXXXXXXXXXXXX<<<<<<<<

Service Definition: WLMDEF01              Installed at: 08/23/05, 11.27.42
     Active Policy: WLMPOL01              Activated at: 08/23/05, 11.30.40

               ------- Goals versus Actuals --------  Trans --Avg. Resp. Time-
               Exec Vel  --- Response Time ---  Perf  Ended  WAIT EXECUT ACTUAL
Name     T  I  Goal Act  ---Goal--- --Actual--  Indx  Rate   Time   Time   Time

BATCH    W           64                                0.400 0.745  12.29  12.99
BATCHLOW S  4    10  90                          0.11  0.000 0.000  0.000  0.000
DISCR    S  D        54                                0.400 0.745  12.29  12.99
CICS     W          N/A                                 2784  0.000  0.033  0.031
CICS     S  2       N/A  0.600 80%        100%  0.50   1863 0.000  0.027  0.020
```

```
CICSCONV S  3       N/A  10.00 50%        100%  0.70  5.217 0.000  4.758  4.758
CICSDEFA S  3       N/A  1.000 90%        100%  0.50  513.5 0.000  0.046  0.045
CICSLONG S  3       N/A  10.00 50%        100%  0.50  0.017 0.000  0.147  0.147
CICSMISC S  3       N/A  1.000 90%        100%  0.50  402.2 0.000  0.002  0.002
CICSRGN  S  2    60  75                         0.80  0.000
ICSS     W          69                               50.55 0.000  0.113  0.113
FAST     S  2    50  83                         0.60  40.48 0.001  0.132  0.132
SLOW     S  4    50  50                         1.00  10.07 0.000  0.037  0.037
VEL30    S  2    30  68                         0.44  0.000
STC      W          66                               25.83 0.001  3.161  3.162
DB2HIGH  S  2    50  57                         0.87  0.000 0.000  0.000  0.000
DDF      S  5     5 0.0                          N/A  0.033 0.000  2.66M  2.66M
```

RMF workload activity report in WLM goal mode

The following illustrates a couple of sections from our RMF *Workload Activity Report* in goal mode. This report is based on a 15 minutes interval. Highlighted on the report you see 78.4% of our CICS transactions are completing in 0.5 seconds, and our CICS workload is processing 1139.54 transactions per second.

```
1                                    W O R K L O A D   A C T I V I T Y

1                                    W O R K L O A D   A C T I V I T Y                                        PAGE   6
        z/OS V1R7              SYSPLEX UTCPLXJ8          START 09/15/2005-11.45.00 INTERVAL 000.15.00   MODE = GOAL
                               RPT VERSION V1R7 RMF      END   09/15/2005-12.00.00

                                        POLICY ACTIVATION DATE/TIME 08/23/2005 11.30.40

   REPORT BY: POLICY=WLMPOL01   WORKLOAD=CICS       SERVICE CLASS=CICS       RESOURCE GROUP=*NONE      PERIOD=1 IMPORTANCE=2
                                                    CRITICAL     =NONE

   TRANSACTIONS    TRANS.-TIME  HHH.MM.SS.TTT
   AVG      0.00   ACTUAL                 30
   MPL      0.00   EXECUTION              39
   ENDED  770497   QUEUED                  0
   END/S  856.07   R/S AFFINITY            0
   #SWAPS      0   INELIGIBLE              0
   EXCTD  527308   CONVERSION              0
   AVG ENC  0.00   STD DEV               195
   REM ENC  0.00
   MS ENC   0.00

              RESP  ------------------------------- STATE SAMPLES BREAKDOWN (%) ------------------------------  ------STATE------
   SUB    P   TIME  --ACTIVE-- READY IDLE  ----------------------------WAITING FOR----------------------------  SWITCHED SAMPL(%)
   TYPE        (%)   SUB  APPL              I/O CONV LOCK MISC PROD LTCH                                        LOCAL SYSPL REMOT
   CICS  BTE    78.4 |  2.7   0.0   0.1  0.0   0.4 96.7  0.0  0.0  0.2  0.0                                      17.7  79.1   0.0
   CICS  EXE  21.9  52.4   0.0  11.5  0.1  29.4  0.0  0.0  1.3  5.2  0.0                                       0.4  51.6   0.0
   DB2   BTE   0.0   0.0   0.0   0.0  0.0   0.0  0.0  0.0  0.0  0.0  0.0                                       0.0   0.0   0.0
   DB2   EXE  18.8  31.2   0.0   0.0  0.0  32.7  0.0  8.9 25.8  0.1  1.2                                       0.0   0.0   0.0
   IMS   BTE   0.3   100   0.0   0.0  0.0   0.0  0.0  0.0  0.0  0.0  0.0                                       0.0   0.0   0.0
   IMS   EXE   3.0  96.8   0.0   0.0  0.0   0.0  0.0  3.2  0.0  0.0  0.0                                       0.0   0.0   0.0
   SMS   BTE   0.0   0.0   0.0   0.0  0.0   0.0  0.0  0.0  0.0  0.0  0.0                                       0.0   0.0   0.0
   SMS   EXE  21.2  46.8   0.0   0.0  0.0  37.8  0.0 15.4  0.0  0.0  0.0                                       0.0   0.0   0.0

   GOAL: RESPONSE TIME 000.00.00.600 FOR  80%

           RESPONSE TIME EX   PERF
   SYSTEM      ACTUAL%    VEL% INDX

   *ALL           100      N/A  0.5
   JA0            100      N/A  0.5
   JB0            100      N/A  0.5
   JC0            100      N/A  0.5
   JE0            100      N/A  0.5
   JF0            100      N/A  0.5
   J80            100      N/A  0.5
   J90            100      N/A  0.5
   Z1            99.4      N/A  0.5
   Z3            99.3      N/A  0.5
1                                    W O R K L O A D   A C T I V I T Y                                        PAGE   7
        z/OS V1R7              SYSPLEX UTCPLXJ8          START 09/15/2005-11.45.00 INTERVAL 000.15.00   MODE = GOAL
                               RPT VERSION V1R7 RMF      END   09/15/2005-12.00.00

                                        POLICY ACTIVATION DATE/TIME 08/23/2005 11.30.40

                                       ----------RESPONSE TIME DISTRIBUTION----------
```

```
      ----TIME----    --NUMBER OF TRANSACTIONS--    -------PERCENT-------  0    10   20   30   40   50   60   70   80   90   100
      HH.MM.SS.TTT    CUM TOTAL        IN BUCKET    CUM TOTAL   IN BUCKET  |....|....|....|....|....|....|....|....|....|....|....|
   <  00.00.00.300          762K             762K         98.8        98.8  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
   <= 00.00.00.360          764K            2495          99.2         0.3  >
   <= 00.00.00.420          766K            1640          99.4         0.2  >
   <= 00.00.00.480          767K            1008          99.5         0.1  >
   <= 00.00.00.540          767K             632           100         0.1  >
   <= 00.00.00.600          768K             465           100         0.1  >
   <= 00.00.00.660          768K             304           100         0.0  >
   <= 00.00.00.720          768K             280           100         0.0  >
   <= 00.00.00.780          769K             193           100         0.0  >
   <= 00.00.00.840          769K             183           100         0.0  >
   <= 00.00.00.900          769K             158           100         0.0  >
   <= 00.00.01.200          769K             557           100         0.1  >
   <= 00.00.02.400          770K             685           100         0.1  >
   >  00.00.02.400          770K             397           100         0.1  >
1                                        W O R K L O A D   A C T I V I T Y                                                    PAGE   8
        z/OS V1R7              SYSPLEX UTCPLXJ8           START 09/15/2005-11.45.00 INTERVAL 000.15.00   MODE = GOAL
                               RPT VERSION V1R7 RMF       END   09/15/2005-12.00.00

                                       POLICY ACTIVATION DATE/TIME 08/23/2005 11.30.40


   REPORT BY: POLICY=WLMPOL01   WORKLOAD=CICS
                                      cics workload

   TRANSACTIONS    TRANS.-TIME  HHH.MM.SS.TTT   --DASD I/O--   ---SERVICE----   --SERVICE TIMES--   PAGE-IN RATES    ----STORAGE----
   AVG     86.95   ACTUAL                124   SSCHRT  8878   IOC     62703K   TCB        4174.0   SINGLE     0.0   AVG     49200.5
   MPL     86.95   EXECUTION              95   RESP     1.4   CPU    775053K   SRB        3200.4   BLOCK      0.0   TOTAL   4277751
   ENDED 1025752   QUEUED                  0   CONN     0.6   MSO      9117M   RCT           0.0   SHARED     0.0   CENTRAL 4277751
   END/S  1139.67  ▌ R/S AFFINITY           0   DISC     0.3   SRB    577547K   IIT          46.6   HSP        0.0   EXPAND     0.00
   #SWAPS      2   INELIGIBLE              0   Q+PEND   0.4   TOT     10532M   HST           0.0   HSP MISS   0.0
   EXCTD  610047   CONVERSION              0   IOSQ     0.0   /SEC    11702K   IFA           7.9   EXP SNGL   0.0   SHARED   573.63
   AVG ENC  0.00   STD DEV            11.665                                  APPL% CP    823.1   EXP BLK    0.0
   REM ENC  0.00                                             ABSRPTN   135K   APPL% IFACP   0.1   EXP SHR    0.0
   MS ENC   0.00                                             TRX SERV  135K   APPL% IFA     0.9
```

Appendix C. Some of our Linux for zSeries samples, scripts and EXECs

Files on our FTP server

Following are some sample files for our FTP server.

lticIPsetup

```
#!/bin/bash
#
# simple script to change the ifcfg-eth0 ip address and hostname regardless on Linux  #
iplist=/etc/ip.list

if [ -e /etc/run_lticIPsetup ]; then

  # assuming first column in the iplist is the ip-address
  uniqueid=$(cat /proc/sysinfo | grep "VM00 Name:" | awk '{print $3}')

  oldhostname=ltic0000.pdl.pok.ibm.com
  oldip=192.168.70.170

  myhostname=${uniqueid}.pdl.pok.ibm.com
  myip=$(grep -i $uniqueid $iplist | awk '{print $1}')

  if [ $(grep -i -c "SUSE" /etc/issue) -gt 0 ]; then
    # we have suse system
    targetfile=$(grep -l $oldip /etc/sysconfig/network/*)
    hostfile=/etc/HOSTNAME
  elif [ $(grep -i -c "RED HAT" /etc/issue) -gt 0 ]; then
    # we have redhat system
    targetfile=$(grep -l $oldip /etc/sysconfig/network-scripts/*)
    hostfile=/etc/sysconfig/network
  else
    echo "system not recognized"
    exit
  fi

  # change network file permanently for future boot up
  cat $targetfile | sed "s/$oldip/$myip/" > /tmp/target-eth0
  cat $hostfile | sed "s/$oldhostname/$myhostname/" > /tmp/target-host
  mv /tmp/target-eth0 $targetfile
  mv /tmp/target-host $hostfile
  chmod 644 $targetfile
  chmod 644 $hostfile

  # change running ip now
  ifconfig eth0 $myip netmask 255.255.255.0 broadcast 192.168.70.255

route add default gw 192.168.70.1
  hostname $myhostname

  # remove so that this script doesn't get run more than once
  rm /etc/run_lticIPsetup
fi
```

ip.list

```
192.168.70.170  LTIC0000
192.168.70.171  LTIC0001
192.168.70.172  LTIC0002
192.168.70.173  LTIC0003
192.168.70.174  LTIC0004
192.168.70.175  LTIC0005
192.168.70.176  LTIC0006
```

```
192.168.70.177  LTIC0007
192.168.70.178  LTIC0008
192.168.70.179  LTIC0009
192.168.70.180  LTIC0010
```

Files on our USER 194 disk

Following are our samples for our USER 194 disk.

PROFILE EXEC

```
/* LINUX PROFILE EXEC */
USR = USERID()
CP SET PF12 RETRIEVE
CP SET MSG ON
CP SET EMSG ON
CP SET RUN ON
CP SET RETRIEVE MAX
CP TERM CHARDEL OFF
ACC 592 K

/* Check if there is a 200 swap disk, if not, create one from V-DISK */
'PIPE CMS Q V 200'
if RC <> 0 THEN 'SWAPGEN 200 2048000 diag'

/* Ipl the Linux system if Disconnected  */
'PIPE CP Q ' USR ' | STACK LIFO '
PARSE PULL USER DASH STATE
IF STATE = 'DSC' THEN 'IPL 201 CLEAR'
ELSE call 'WELCOME'
EXIT
```

WELCOME EXEC

```
/*****************************************************************/
/*    WELCOME EXEC                                               */
/*        Display a Menu for the user                            */
/*****************************************************************/
/* Ipl the Linux system if Disconnected  */
'PIPE CMS ID | STACK LIFO '
PARSE PULL USER AT SYS .

TOP:
CLRSCRN  /* CLEAR THE SCREEN  */
say ' Welcome to ' USER ' on ' sys
say ' You have the following options: '
say;
say ' 1) IPL Linux on the 201 disk '
say ' 2) IPL Linux on the a disk other then 201 '
say ' 3) Install a new linux system '
say ' 4) Copy a pre-built Linux system '
say ' 5) Display my Networking Information'
say ' 6) What can I do with a LTICxxxx system '
say ' 7) EXIT '
 say;
 Pull opt
 SELECT
    when opt = 1 then do
         'IPL 201 CLEAR'
    end

    when opt = 2 then do
         say 'What disk do you want to IPL'
         pull disk
         if strip(disk)='CMS' then 'IPL CMS'
         ELSE 'IPL ' disk ' CLEAR'
```

```
    end

    when opt = 3 then do
         call DISTRO
    end

   when opt = 4 then do
        call DISTCOPY
      SIGNAL TOP
   end

   when opt = 5 then do
       'IPDATA'
       say ; say 'Hit enter to continue'
       pull .
       SIGNAL TOP
   end

   when opt = 5 then do
       'IPDATA'
       say ; say 'Hit enter to continue'
       pull .
       SIGNAL TOP
   end

   when opt = 6 then do
       'HELP LTIC'
      SIGNAL TOP
   end
   otherwise nop;
end /* Select */
```

Files on our USER 195 disk

Following are our samples for our USER 195 disk.

DISKCOPY EXEC

```
/*****************************************************************/
/*  DISTCOPY - Copy the DISTRO from the Master pack to 201 disk  */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
'CLRSCRN'  /* Clear the screen */
'PIPE < DISTCOPY LIST * |spec 1-3 1 14-80 5 |  CONSOLE'
say;say;
say 'Enter the ID of the system you would like to copy'
say '       or Blank to exit'
pull id
if id = '' then exit
'PIPE < DISTCOPY LIST * | LOCATE 1-3 /'id'/ | STACK LIFO '
PARSE PULL new_id VOL DESC

'CLRSCRN'  /* Clear the screen */

"FLASHCOPY " VOL" 0 END 201 0 END"
IF RC = 0 then do
     say" The copy has completed"
    say ' Hit enter to return to the Main Menu'
    pull .
    EXIT 0
END

/* If we got here the flash copy failed so we will do a DDR copy */
'CLRSCRN'  /* Clear the screen */
```

```
SAY;say; SAY '                          HIT ENTER WHEN COPY COMPLETES '

    'makebuf'
    push '   '
    push 'YES'
    push 'YES'
    push 'COPY ALL'
    push 'OUTPUT 201 DASD'
    push 'INPUT 'VOL 'DASD'
    push 'SYSPRINT CONS'
    'DDR'
    'dropbuf'
    Say ' Copy has completed with Return Code: 'rc
    say ' Hit enter to return to the Main Menu'
    pull .
```

DISKCOPY LIST

```
ID           DISTRO

1   572A     SUSE   SLES 9 RC5           64 BIT  2.6.5-7.97-s390x
2   5720     SUSE   SLES 9 RC5           31 BIT  2.6.5-7.97-s390

3   572B     RHEL4 UPDATE 1 BETA         64 BIT  2.6.9-6.37.EL
4   5721     REDHAT RHEL 4  BETA 1       31 BIT  2.6.8-1.528.2.5

5   572C     REDHAT RHEL4 BETA1 REFRESH  64 BIT  2.6.8-1.528.2.5
6   5722     REDHAT RHEL4 BETA1 REFRESH  31 BIT  2.6.8-1.528.2.5

7   572D     REDHAT RHEL4 BETA 2         64 BIT  2.6.9-1.648_EL
8   5723     REDHAT RHEL4 BETA 2         31 BIT  2.6.9-1.648_EL

9   5724     REDHAT RHEL3 UPDATE 3       31 BIT  2.4.21-4.EL
```

DISTRO EXEC

```
/*****************************************************************/
/* DISTRO EXEC                                                   */
/*                                                               */
/* This exec will display the linux RamDisk systems that can be  */
/*  be loaded and then prompt the user for the system to install.*/
/*                                                               */
/*                                                               */
/*****************************************************************/

'CLRSCRN'  /* Clear the screen */
'PIPE < DISTRO LIST * |spec 1-3 1 14-80 5 |  CONSOLE'
say;say;
say 'Enter the ID of the system you would like to install'
say '       or Blank to exit'
pull id
if id = '' then exit
'PIPE < DISTRO LIST * | LOCATE 1-3 /'id'/ | STACK LIFO '
PARSE PULL new_id TAG DESC
'LOADRDR ' tag
```

DISKCOPY LIST

```
ID           DISTRO
1   26579731  SUSE   SLES 9 RC5          31 BIT  2.6.5-7.97-s390
2   26579764  SUSE   SLES 9 RC5          64 BIT  2.6.5-7.97-s390x

3   24211764  REDHAT RHEL3  update 3     64 BIT- 2.4.21-17.EL
4   24211731  REDHAT RHEL3  update 3     31 BIT  2.4.21-17.EL

5   241931    SUSE   SLES 8              31 BIT  2.4.19-3suse-SMP
6   SLES864   SUSE   SLES 8              64 BIT  2.4.19
```

```
7   RHEL4U1B  REDHAT RHEL4 UPDATE1 Beta  64 BIT 2.6.9-6.37.EL
8   RH4U131   REDHAT RHEL4 UPDATE1 Beta  31 BIT 2.6.9-1.37.EL

9   26571264  SUSE   SLES 9 RC1-update   64 BIT  2.6.5-7.127-s390x

A   RH4RC231  REDHAT RHEL4 RC2           31 BIT 2.6.9-1.906_EL
B   RH4RC264  REDHAT RHEL4 RC2           64 BIT 2.6.9-1.906_EL

C   RH4RC_31  REDHAT RHEL4 GA            31 BIT 2.6.?
D   RH4RC_64  REDHAT RHEL4 GA            64 BIT 2.6.?

E   S8SP3_31  SuSE SLES 8 SP 3           31 BIT 2.6.?
F   S8SP3_64  SuSE SLES 8 SP 3           64 BIT 2.6.?
```

IPDATA EXEC

```
/*****************************************************************/
/*  IPDATA EXEC - Display the users IP Information.              */
/*                                                               */
/*****************************************************************/
'PIPE CMS ID | STACK LIFO '
PARSE PULL USER AT SYS .
CLRSCRN  /* CLEAR THE SCREEN  */

'PIPE < IPDATA LIST * | LOCATE 1-8 /'USER'/| VAR OUTPUT '
 PARSE VAR OUTPUT USERID IP


say ' The IP Address for user: ' USER ' on ' sys 'is: 'IP

/* GET A LIST OF ALL THE VSWITCHES */
'PIPE CP Q VSWITCH | LOCATE /VSWITCH SYSTEM/ | STEM VSWITCH. '

x=1

DO VSWITCH.0
   switch_str=vswitch.x
   parse var switch_str . . switch .
   'PIPE CP Q VSWITCH DETAIL 'switch'| LOCATE /RDEV/ | var port'
   PARSE VAR PORT . name .
  'PIPE CP Q VSWITCH DETAIL 'switch' | LOCATE /'USER'/ | var detail '
  PARSE VAR detail . . . . dev .
  if dev <> '' then do
    say; say ' VSWITCH 'switch' DEVICE:'dev ' Portname:'name
  end
x=x+1
END /* do */
```

IPDATA LIST

```
LTIC0000     192.167.70.170
LTIC0001     192.167.70.171
LTIC0002     192.167.70.172
LTIC0003     192.167.70.173
LTIC0004     192.167.70.174
LTIC0005     192.167.70.175
LTIC0006     192.167.70.176
LTIC0007     192.167.70.177
LTIC0008     192.167.70.178
LTIC0009     192.167.70.179
LTIC0010     192.167.70.180
```

LOADRDR EXEC

```
/*****************************************************************/
/* LOADRDR EXEC                                                  */
/* This EXEC will load the required files into the reader        */
/* so that you can IPL the LINUX RAM DISK to build the DISTRO    */
/*                                                               */
/* Usage:  DISKLOAD linux_distro_id                              */
/*                                                               */
/*****************************************************************/
arg tag
if tag  = '' then do
    say 'USEAGE:   linux_distro_id '
    say;
say 'RUN the xxxx EXEC to get the linux_distro_id'
    EXIT
end

'close rdr'
'purge rdr all'
'spool punch * rdr'

/*  Need to add some code to make sure the files exist */

'PUNCH 'tag ' IMAGE D (NOH'
'PUNCH 'tag ' PARM D (NOH'
'PUNCH 'tag ' INITRD D (NOH'
'change rdr all keep nohold'
'ipl 00c clear'
```

Appendix D. Availability of our test reports

The following information describes the variety of ways in which you can obtain our test reports.

Our publication schedule is changing

Starting in 2003, our publication schedule changed somewhat from our traditional quarterly cycle as a result of the planned change in the development cycle for annual z/OS releases. Keep an eye on our Web site for announcements about the availability of new editions of our test report.

Availability on the Internet: You can view, download, and print the most current edition of our test report from our z/OS Integration Test Web site at:

www.ibm.com/servers/eserver/zseries/zos/integtst/

Our Web site also provides all of our previous year-end editions, each of which contains all of the information from that year's interim editions.

You can also find our test reports on the z/OS Internet Library Web site at:

www.ibm.com/servers/eserver/zseries/zos/bkserv/

Each edition is available in the following formats:

- IBM BookManager BOOK format

  On the Web, BookManager documents are served as HTML via IBM BookServer. You can use your Web browser (no plug-in or other applications are needed) to view, search, and print selected topics. You can also download individual BOOK files and access them locally using the IBM Softcopy Reader or IBM Library Reader™. You can get the Softcopy Reader or Library reader free of charge from the IBM Softcopy Web site at www.ibm.com/servers/eserver/zseries/softcopy/.
- Adobe Portable Document Format (PDF)

  PDF documents require the Adobe Acrobat Reader to view and print. Your Web browser can invoke the Acrobat Reader to work with PDF files online. You can also download PDF files and access them locally using the Acrobat Reader. You can get the Acrobat Reader free of charge from www.adobe.com/products/acrobat/readstep.html.

Softcopy availability: BookMaster BOOK and Adobe PDF versions of our test reports are included in the OS/390 and z/OS softcopy collections on CD-ROM and DVD. For more information about softcopy deliverables and tools, visit the IBM Softcopy Web site (see above for the Web site address).

A note about the currency of our softcopy editions

Because we produce our test reports twice a year, June and December, we cannot meet the production deadline for the softcopy collections that coincide with the product's GA release and the softcopy collection refresh date six months later. Therefore, there is normally a one-edition lag between the release of our latest test report edition and the softcopy collection in which it is included. That is, the test report that appears in any given softcopy collection is normally one edition behind the most current edition available on the Web.

Hardcopy availability: Our December 2001 edition was the last edition to be published in hardcopy. As of 2002, we no longer produce a hardcopy edition of our year-end test reports. You can still order a printed copy of a previous year-end edition (using the order numbers shown in Table 19 below) through your normal ordering process for IBM publications.

Available year-end editions: The following year-end editions of our test report are available:

Table 19. Available year-end editions of our test report

Title	Order number	This year...	And these releases...	Softcopy collection kits
zSeries Platform Test Report	SA22-7997-00	2004	z/OS V1R6	SK3T-4269-14
z/OS Parallel Sysplex Test Report	SA22-7663-09	2003	z/OS V1R4	SK3T-4269-11 SK3T-4271-11
z/OS Parallel Sysplex Test Report	SA22-7663-07	2002	z/OS V1R3 and V1R4	SK3T-4269-07 SK3T-4271-07
z/OS Parallel Sysplex Test Report	SA22-7663-03	2001	z/OS V1R1 and V1R2	SK3T-4269-03 SK3T-4270-04 SK3T-4271-03
OS/390 Parallel Sysplex Test Report	GC28-1963-19	2000	OS/390 V2R9 and V2R10	SK2T-6700-24 SK2T-6718-14
OS/390 Parallel Sysplex Test Report	GC28-1963-15	1999	OS/390 V2R7 and V2R8	SK2T-6700-17
OS/390 Parallel Sysplex Test Report	GC28-1963-11	1998	OS/390 V2R5 and V2R6	SK2T-6700-15 and -17
OS/390 Parallel Sysplex Test Report	GC28-1963-07	1997	OS/390 V1R3 and V2R4	SK2T-6700-11 and -13
OS/390 Parallel Sysplex Test Report	GC28-1963-03	1996	OS/390 V1R1 and V1R2	SK2T-6700-07
S/390 MVS Parallel Sysplex Test Report	GC28-1236-02	1995	MVS/ESA SP V5	none

Other related publications: From our Web site, you can also access other related publications, including our companion publication, *OS/390 Parallel Sysplex Recovery*, GA22-7286, as well as previous editions of *OS/390 e-Business Integration Test (eBIT) Report*.

Appendix E. Useful Web sites

We have cited the IBM books we used to do our testing as we refer to them in each topic in this test report. This chapter contains listings of some of the Web sites that we reference in this edition or previous editions of our test report.

IBM Web sites

Table 20 lists some of the IBM Web sites that we reference in this edition or previous editions of our test report:

Table 20. Some IBM Web sites that we reference

Web site name or topic	Web site address
IBM Terminology (includes the *Glossary of Computing Terms*)	www.ibm.com/ibm/terminology/
IBM CICS Transaction Gateway	www.ibm.com/software/ts/cics/library/
IBM HTTP Server library	http://www.ibm.com/software/webservers/httpservers/library/
IBMLink™	www.ibm.com/ibmlink/
IBM mainframe servers Internet library	www.ibm.com/servers/eserver/zseries/library/literature/
IBM Redbooks	www.ibm.com/redbooks/
IBM Systems Center Publications (IBM TechDocs — flashes, white papers, etc.)	www.ibm.com/support/techdocs/
Linux at IBM	www.ibm.com/linux/
Net.Data Library	http://www.ibm.com/servers/eserver/iseries/software/netdata/docs/index.html
OS/390 Internet library	www.ibm.com/servers/s390/os390/bkserv/
Parallel Sysplex	www.ibm.com/servers/eserver/zseries/pso/
Parallel Sysplex Customization Wizard	http://www.ibm.com/servers/eserver/zseries/pso/tools.html
System Automation for OS/390	www.ibm.com/servers/eserver/zseries/software/sa/
WebSphere Application Server	www.ibm.com/software/webservers/appserv/
WebSphere Application Server library	`www.ibm.com/software/webservers/appserv/zos_os390/library/`
WebSphere Studio library	http://www.ibm.com/developerworks/views/websphere/libraryview.jsp
WebSphere Studio Workload Simulator	www.ibm.com/software/awdtools/studioworkloadsimulator/library/
z/OS Consolidated Service Test	www.ibm.com/servers/eserver/zseries/zos/servicetst
z/OS downloads	www.ibm.com/servers/eserver/zseries/zos/downloads/
z/OS Integration Test (includes information from OS/390 Integration Test and e-business Integration Test (ebIT))	www.ibm.com/servers/eserver/zseries/zos/integtst/
z/OS Internet library	www.ibm.com/servers/eserver/zseries/zos/bkserv/
z/OS UNIX System Services	www.ibm.com/servers/eserver/zseries/zos/unix/
z/OS.e home page	www.ibm.com/servers/eserver/zseries/zose/

Table 20. Some IBM Web sites that we reference (continued)

Web site name or topic	Web site address
z/OS.e Internet library	www.ibm.com/servers/eserver/zseries/zose/bkserv/

Other Web sites

Table 21 lists some other non-IBM Web sites that we reference in this edition or previous editions of our test report:

Table 21. Other Web sites that we reference

Web site name or topic	Web site address
Cisco Systems	www.cisco.com/
Java Servlet Technology	java.sun.com/products/servlet/
Java 2 Platform, Enterprise Edition (J2EE)	java.sun.com/products/j2ee/
JavaServer Pages (JSP)	java.sun.com/products/jsp/
J2EE Connector Architecture	java.sun.com/j2ee/connector/
SUSE Linux	www.suse.com/

Appendix F. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using such products to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to *z/OS TSO/E Primer*, *z/OS TSO/E User's Guide*, and *z/OS ISPF User's Guide Vol I* for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

z/OS information

z/OS information is accessible using screen readers with the BookServer/Library Server versions of z/OS books in the Internet library at:

`www.ibm.com/servers/eserver/zseries/zos/bkserv/`

Notices

This information was developed for products and services offered in the USA.

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

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

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

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

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

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

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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

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

COPYRIGHT LICENSE:

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

these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

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

AIX
BatchPipes
BookManager
BookMaster
CICS
CICSPlex
DB2
DB2 Connect
DFS
DFSMS/MVS
DFSMShsm
DFSMSrmm
Enterprise Storage Server
ESCON
@server
FICON
IBM
ibm.com
IBMLink
IMS
Infoprint
Language Environment
Library Reader
MQSeries
MVS
MVS/ESA
Net.Data
NetView
OS/2
OS/390
Parallel Sysplex
PR/SM
Processor Resource/Systems Manager
QMF
RACF
RAMAC
Redbooks
RMF
RS/6000
S/390
SP
SupportPac
Sysplex Timer
TotalStorage
VisualAge
VSE/ESA
VTAM
WebSphere
z/OS
z/OS.e
z/VM
zSeries

The following terms are trademarks of other companies:

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

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

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

UNIX is a registered trademark of The Open Group in the United States and other countries.

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

Index

F

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

Z

Readers' Comments — We'd Like to Hear from You

z/OS
zSeries Platform Test Report for z/OS and Linux Virtual Servers
Version 1 Release 7

Publication No. SA22-7997-03

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Name

Address

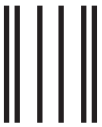
Company or Organization

Phone No.

Readers' Comments — We'd Like to Hear from You
SA22-7997-03

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department B6ZH, Mail Station P350
2455 South Road
Poughkeepsie, NY
12601-5400

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line

Printed in USA

SA22-7997-03