z/OS

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

Version 1 Release 8

SA22-7997-05

z/OS

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

Version 1 Release 8

SA22-7997-05

Note!
Before using this information and the products it supports, be sure to read the general information under “Notices” on page 257.

Sixth Edition, June 2007

This is a major revision of SA22-7997-04.

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(LBCRUZ)
Internet e-mail: lbcruz@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 xxi.

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
Loraine Arnold
Ozan Baran
Ryan Bartoe
Duane Beyer
Jeff Bixler
Muriel Bixler
Dave Buehl
Jon Burke
Alex Caraballo
Phil Chan
John Corry
Don Costello
Luis Cruz
Tony DiLorenzo
Eli Dow
Bob Fantom
Nancy Finn
Bobby Gardinor
Kieron Hinds
Gerry Hirons
Joan Kelley
Fred Lates
Al Lease
Frank LeFevre
Scott Loveland
George Markos
Sue Marcotte
Tammy McAllister
Azeem Mohammed
Elaine Murphy
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 7
- "Our Parallel Sysplex software configuration" on page 15
- "Our workloads" on page 18
- "Our networking configuration" on page 27

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 C, "Availability of our test reports," on page 251 for other ways to access our test reports.

We publish our report twice a year, every June and December. The contents of our test reports remain cumulative for any given 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 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, *z/OS V1R8.0 System z Parallel Sysplex Recovery*, GA22-7286. In this publication, we focus on describing:

- How to be prepared for potential problems in a Parallel Sysplex
- What the indicators are to let you know there is a problem
- What actions to take to recover

The recovery scenarios we describe are based on our own experiences in our particular test environment while running z/OS V1R8, DB2 V8, IMS V9, WebSphere Application Server V6.0, WebSphere MQ V6 and CICS TS V3R1, These scenarios do not represent a comprehensive list of all possible approaches and outcomes, but do represent the approaches we have tested and that work for us.

Note: The recovery book was written in the z/OS V1R8 time frame; however, many of the recovery concepts that we discuss still apply to later releases of 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	*z/OS Parallel Sysplex Overview*, SA22-7661SA22-7661
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 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 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*. Starting with z/OS V1R4, z/OS users can obtain the IBM Health Checker for z/OS from the z/OS Downloads page at 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:

- lbcruz@us.ibm.com for z/OS questions
- lefevre@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 SA22-7997-05
June 2007

This document contains information previously presented in SA22-7997-04.

New information

- Part 1, “System z Platform Evaluation Test,” on page 1
  - “Access to the Domain Name Server (DNS)” on page 36
  - “Using LDAP Server” on page 37
  - “Enterprise Key Manager Offering for Tape Encryption” on page 47
  - “Encryption Facility V1.2, setting up and testing the support for the OpenPGP standard” on page 59
  - Chapter 6, “Migrating to DB2 Version 9.1,” on page 81
  - “z/OS UNIX Directory List” on page 109
- Part 2, “Linux virtual servers,” on page 161
  - Chapter 12, “About our environment,” on page 163
  - Chapter 13, “zLVS PET Recovery,” on page 169
    - “Recovery process overview” on page 169
    - “Recovery from a resource failure” on page 170
    - “Recovery from a software server failure” on page 197
    - “Recovery from a z/VM, LPAR, Linux failure” on page 232
    - “Summary of Linux Recovery Test” on page 238
  - Chapter 14, “Future Linux on System z projects,” on page 243

Changed information

- “Our Parallel Sysplex hardware configuration” on page 7
- “Our Parallel Sysplex software configuration” on page 15
- “Our Integrated Cryptographic Service Facility (ICSF) configuration” on page 35
- “Overview of our LDAP configuration” on page 38
- Appendix B, “Some of our RMF reports,” on page 247
- Appendix C, “Availability of our test reports,” on page 251

Removed information: The following removed information can be found in our last report; the *zSeries Platform Test Report for z/OS and Linux Virtual Servers (SA22-7997-04) December 2006 edition*:

- Creating a split plex for production and test
- RACF Security Server mixed case password support

- Testing the IBM Encryption Facility for z/OS
- Migrating to z/OS V1R7
- Migrating to z/OS.e V1R7
- Migrating z/OS Images and a Coupling Facility to the z9 z/OS performance
- Migrating to CICS TS Version 3 Release 1
- Migrating to DB2 Version 8
- Migrating to IMS Version 9
- Implementing the IMS Common Service Layer and the Single Point of Control
- Using IBM Health Checker for z/OS
- z/OS UNIX enhancements in z/OS V1R7
- Using the hierarchical file system (HFS)
- Automount enhancement for HFS to zSeries file system (zFS) migration
- Using the zSeries file system (zFS)
- Displaying z/OS UNIX and zFS diagnostic information through message automation
- Removing additional diagnostic data collection from OMVS CTRACE LOCK processing
- Migrating from WebSphere MQ V5.3.1 to V6
- Using Websphere Message Broker
- Defining JMS and JDBC Resources for Trade6
- Providing authentication, course-grained security, and single sign-on for Web/EJB based applications running on WebSphere Application Server for z/OS
- Federated Single Sign-On with Tivoli Federated Identity Manager and WebSphere Application Server on z/OS
- **For Linux Virtual servers**
- Implementing High Availability Architectures
- Migrating middleware
- Installing and configuring WebSphere Portal Server Cluster
- Linux and z/VM system programmer tips
- Linux Utilities for System z
- Upgrading TAMe, TAMe WebSEAL, and TDS to v6

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 SA22-7997-04
December 2006

This document contains information previously presented in SA22-7997-03.

New information

- “Migrating to a Server Time Protocol Coordinated Timing Network (CTN)” on page 14
- Chapter 5, “Using the z9 Integrated Information Processor (zIIP),” on page 71

- “OMEGAMON XE for z/OS 3.1.0 zIIP SUPPORT” on page 77
- Chapter 7, “Implementing IMS JDBC Connector (formerly IMS Java),” on page 101
- Chapter 8, “Implementing IMS SOAP Gateway,” on page 105
- “z/OS UNIX enhancements in z/OS V1R8” on page 109
  - “Setting and changing the file format from the UNIX System Services shell” on page 110
  - “z/OS UNIX System Services: Displaying z/OS UNIX Latch Contention” on page 111
  - “Enhancements to the DISPLAY OMVS,F command” on page 115
  - “Preventing mounts during file system ownership shutdown” on page 116
  - “Distributed BRLM (Byte Range Lock Manager) with Lock Recovery Support” on page 117
- “Using the _UNIX03 z/OS UNIX Shell environment variable” on page 118
- “Implementing /etc/inittab in z/OS UNIX” on page 120
- “Moving to 64-bit Java and JDK 5” on page 122
- “BPXBATCH enhancements in z/OS V1R8” on page 124
- “BPXMTEXT support for zFS reason codes” on page 125
- “z/OS zFS enhancements in z/OS V1R8” on page 125
  - “Deny mounting of a zFS file system contained in a multi-file system aggregate when running in sysplex mode on z/OS V1R8” on page 126
  - “Stop zFS (modify omvs,stoppfs=zfs)” on page 127
- “Conflict with SDK for z/OS (java)” on page 36
- “Enabling WebSphere MQ Security” on page 135
- “Migrating to Websphere Message Broker Version 6” on page 137
- “EDSW – High Availability for WebSphere MQ-IMS bridge application” on page 140
- “Setting up eWLM for Application and System Monitoring” on page 150
- “Using SAF (RACF) on our TCPIP.PROFILE port reserves” on page 152
- “Setting up WebSphere Developer for zSeries (WDz) on PET Plex systems” on page 154
- Chapter 14, “Future Linux on System z projects,” on page 243

Changed information

- Our sysplex hardware configuration
- Our sysplex software configuration
- “Overview of our LDAP configuration” on page 38

Removed information: The removed information can be found in our June 2006 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 SA22-7997-03
June 2006

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

New information

- "Managing your z/OS queue managers using WebSphere MQ V6 Explorer" on page 130
- "Setting up WebSphere for eWLM monitoring of DB2 applications" on page 147
- Part 2, "Linux virtual servers," on page 161 including:
  - Chapter 12, "About our environment," on page 163
  -

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/

Summary of changes
for SA22-7997-02
December 2005

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

New information

- "Tivoli Workload Scheduler (TWS) EXIT 51 tip:" on page 20
- "z/OS UNIX System Services: Displaying z/OS UNIX Latch Contention" on page 111
- Appendix B, "Some of our RMF reports," on page 247

Changed information

- Our sysplex hardware configuration

Summary of changes
SA22-7997-01
June 2005

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

New information

- "Our Integrated Cryptographic Service Facility (ICSF) configuration" on page 35
- "About our z/OS V1R8 test environment running WebSphere Application Server" on page 143
- Chapter 14, "Future Linux on System z projects," on page 243
- Appendix B, "Some of our RMF reports," on page 247

Changed information

- Our sysplex hardware configuration
- Our networking configuration

Part 1. System z Platform Evaluation Test

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 run two Parallel Sysplexes, one with 9 members and the other with 3 members that consist of the following:

- Four central processor complexes (CPCs) running z/OS in 12 logical partitions (LPs).

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

  The z/OS images consist of the following:
  - Eight production z/OS systems
  - 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.)
- Five coupling facilities:
  - One failure-independent coupling facility that runs in a LP on a standalone CPC
  - Four non-failure-independent coupling facilities that run in LPs on three 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 9 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.

Parallel Sysplex environment

Just recently we removed the IBM System z9 BC Model 2096-S07 from our plex. This processor was running our z/OS.e V1R8 operating system. The last release to support z/OS.e is V1R8. Given this, and the fact that we will soon be migrating our entire plex over to z/OS V1R9, we decided not to continue to run z/OS.e in our environment (for the limited time we were still on V1R8).

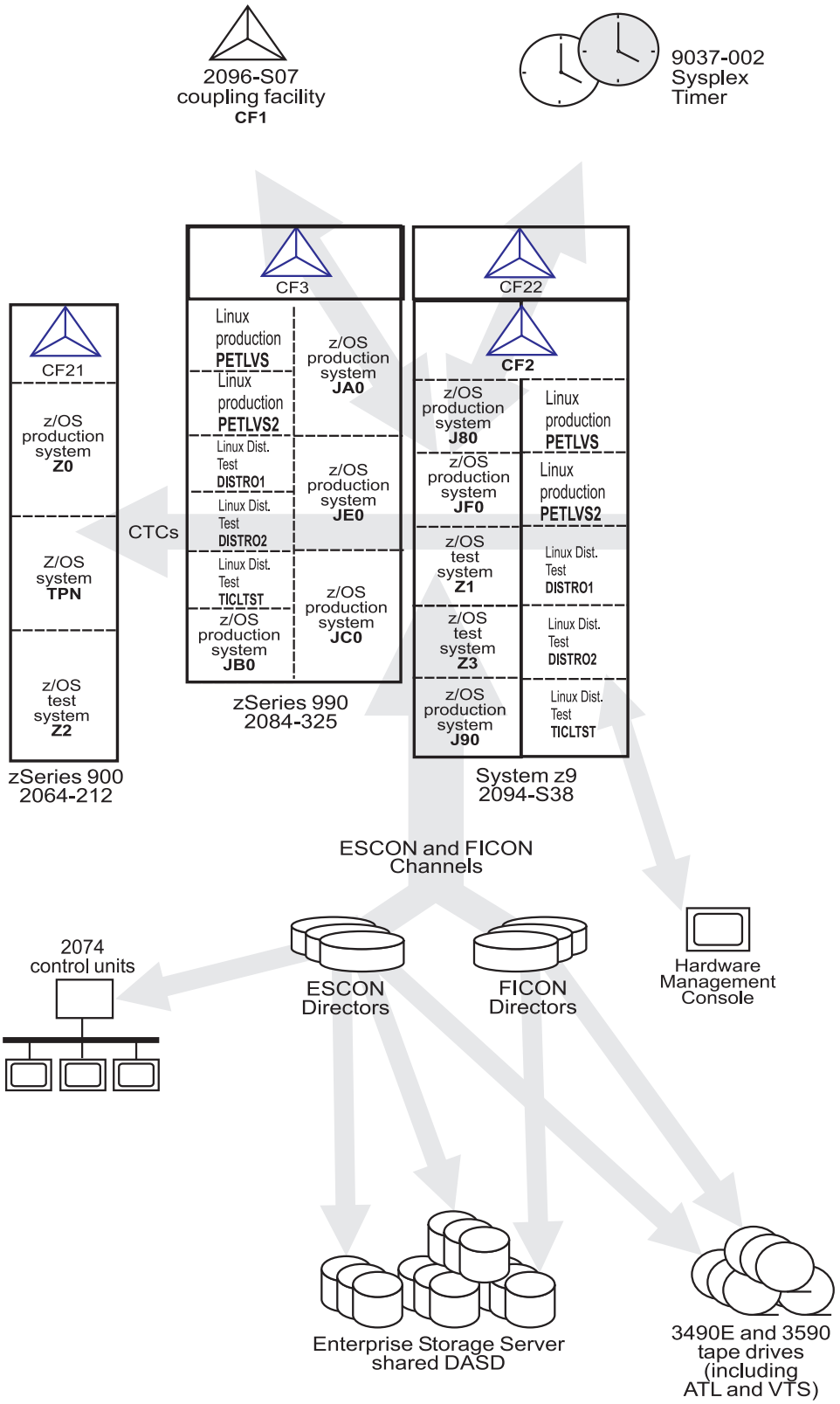


Figure 1. Our sysplex hardware configuration

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 9.

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	LCSS	System name, usage Virtual CPs (static, managed) Initial LPAR weight
IBM System @server System z9 EC (2094-S38)	1 CPC 38 CPs	LPAR 9 LPs	2176M	112640M	0-	**J80**, z/OS production system Plex 1 32 shared CPs 2 shared zIIPs 2 shared zAAPs
				155648M		**J90**, z/OS production system Plex 1 32 shared CPs 2 shared zIIPs 2 shared zAAPS
				15360M	0	**JF0**, z/OS production system Plex 1 16 shared CPs 2 shared zIIPs 2 shared zAAPS
				24576M	0	**Z1**, z/OS test system Plex 2 8 shared CPs 2 shared zIIPs 2 shared zAAPs
				24576M	0	**Z3**, z/OS test system Plex 2 8 shared CPs 2 shared zIIPs 2 shared zAAPs
				3072M	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

Table 2. Our mainframe servers (continued)

Server model (Machine type-model)	CPCs CPs	Mode LPs	HSA	Storage: Central	LCSS	System name, usage Virtual CPs (static, managed) Initial LPAR weight
IBM @server zSeries 900 Model 212 (2064-212)	1 CPC 16 CPs 4 ICF	LPAR mode 4 LPs (1 LP is a coupling facility)	256M	9216M		**Z0**, z/OS production system Plex 1 8 shared CPs
				6144M		**TPN**, z/OS system for TPNS Plex 1 12 shared CPs
				10752M	0	**Z2**, z/OS test system Plex 2 8 shared CPs,
IBM @server zSeries 990 Model 325 (2084-325)	1 CPC 32 CPs 2 IFL, 2 zAAP)	LPAR mode 20 LPs	1536MB	30720M	2	**JA0**, z/OS production system Plex 1 16 shared CPs, 2 shared zAAPs
				30720M	0	**JB0**, z/OS production system Plex 1 16 shared CPs, 2 shared zAAPs
				22528M	0	**JC0**, z/OS production system Plex 1 16 shared CPs, 2 shared zAAPs
				22528M	2	**JE0**, z/OS production system Plex 1 16 shared CPs, 2 shared zAAPs
				16384M	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

Coupling facility details

Table 3 provides information about the coupling facilities in our sysplex. Figure 1 on page 9 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
CF1 (Plex 1)	IBM System Z9 BC Model 2096-S07 stand-alone coupling facility 1 CPC with 4 CPs CFLEVEL=15 (CFCC Release 14.00, Service Level 04.05) Controlled by the HMC	14G
CF2 (Plex 1)	Coupling facility LP on a System z9 (2094-S38) 3 dedicated ICF CPs CFLEVEL=15 (CFCC Release 14.00, Service Level 00.17) Controlled by the HMC	14G
CF3 (Plex 1)	Coupling facility LP on a zSeries 990 Model 325 (2084-325) 3 dedicated ICF CPs CFLEVEL=14 (CFCC Release 14.00, Service Level 00.28) Controlled by the HMC	14G
CF21 (Plex 2)	Coupling facility LP on a zSeries 900 Model 212 (2064-212) 1 dedicated ICF CP CFLEVEL=13 (CFCC Release 13.00, Service Level 04.12) Controlled by the HMC	6G
CF22 (Plex 2)	Coupling facility LP on a System z9 (2094-S38) 1 dedicated ICF CP CFLEVEL=15 (CFCC Release 14.00, Service Level 04.05) 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		
		2096-S07	2094-S38	2084-325
		CF1	CF2	CF3
z/OS and CF Images				
2084-325 JA0, JE0, JC0, JB0, CF3		1 CBP 3 CFP	1 CBP 3 CFP	4 CBP 4 CFP
2064-212 Z0, TPN		6 CFP	2 CBP *	4 ICP
2094-S38 J80, JF0, CF2, J90		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

Table 5. Our coupling facility channel configuration on Plex 2. (continued)

Coupling Facility Channel Connections on Plex 2			
		2094-S38	2064-212
		CF22	CF21
z/OS and CF Images			
2064-212 Z2 CF21		2 CFP * 1 CBP *	2 ICP
2094-S38 Z1,Z3 CF22		2 ICP	2 CFP * 1 CBP *
		* = Same Links	

In addition to our coupling facility channel configuration listed in Table 4 on page 12 and Table 5 on page 12, we configured 1 ISC 3 link and 1 ICB 3 link between our 2084-325 CPC and our 2096-S07 CPC which will be used as Server Time Protocol (STP) timing-only links in our new STP environment. Please refer to "Migrating to a Server Time Protocol Coordinated Timing Network (CTN)" on page 14 for a further description of STP timing-only 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(R) (ESS, 2105-F20, 800) IBM System Storage (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)	3494 Model L10 with 16 Escon and Ficon attached 3590 tape drives and 8 3592 (Encryption capable) tape drives	All tape drives are accessible from all systems.
Virtual Tape Server (VTS)	3494 Model L10 with 32 virtual 3490E tape drives.	All tape drives are accessible from all systems.

Migrating to a Server Time Protocol Coordinated Timing Network (CTN)

We migrated to the Server Time Protocol (STP) since our last report. Details on this migration can be found on Our latest tips and experiences website. For more information, see the section ″Migrating to a Server Time Protocol Configuration″ at:

`www.ibm.com/servers/eserver/zseries/zos/integtst/tips.html`

where we discuss the z/OS Integration Test team's experiences associated with that migration.

We begin by first presenting a brief overview of STP and the respective terminology, and then follow it with a high level overview of the z/OS Integration Test lab environment.

We also present both the planning considerations and the actual migration steps taken by the team to deploy STP in their data center.

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)
- DB2 UDB for z/OS and OS/390 V9.1 (and its associated IRLM)
- WebSphere for z/OS V6.0.2
- WebSphere MQ for z/OS V6
- Websphere Message Broker V6

Up until recently, we ran z/OS.e in one partition on our System z9 BC 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. There are a couple of reasons, which when combined, made us decide to stop running this in our environment. First, we removed the server that was running z/OS.e V1R8, from our plex. The last release to support z/OS.e is V1R8. Given that and the fact that we will soon be migrating our entire plex over to z/OS V1R9, we decided not to continue to run z/OS.e in our environment (for the limited time we were still on V1R8).

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.

Overview of our software configuration

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

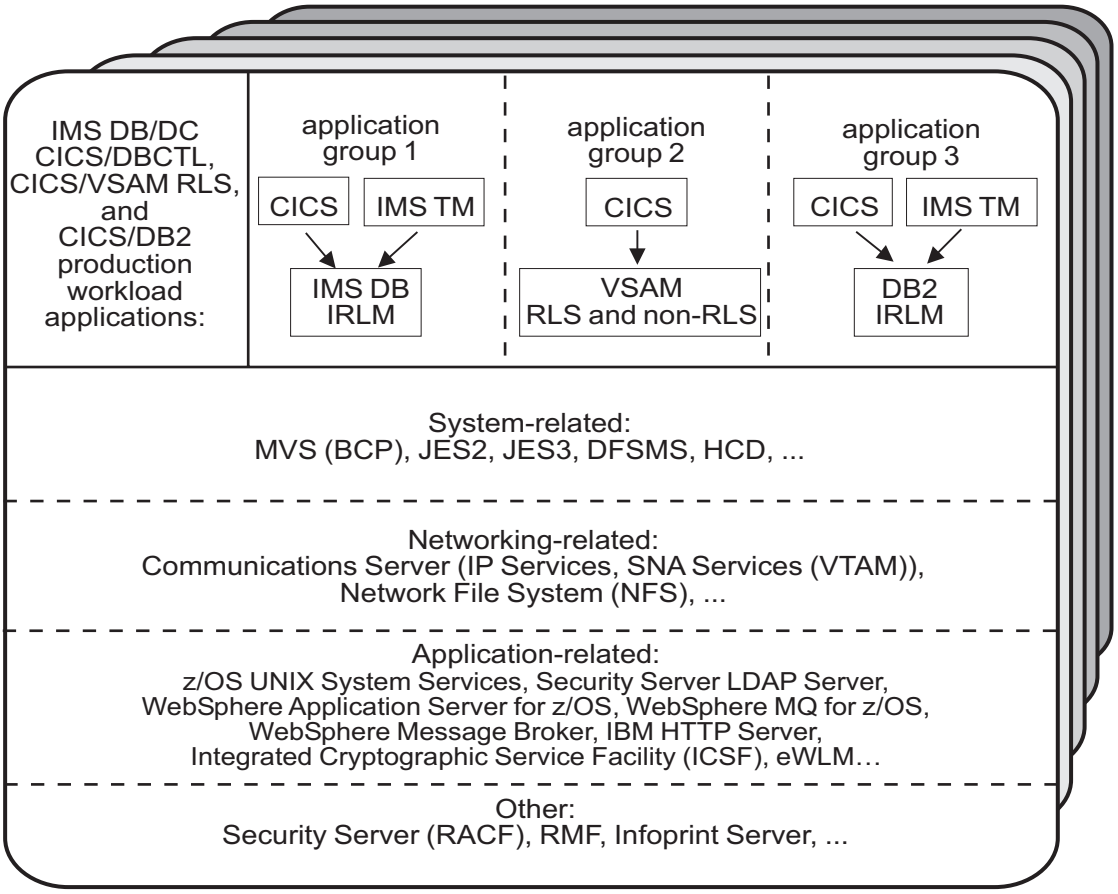


Figure 2. Our sysplex software configuration

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 2007 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 VTAM configuration

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

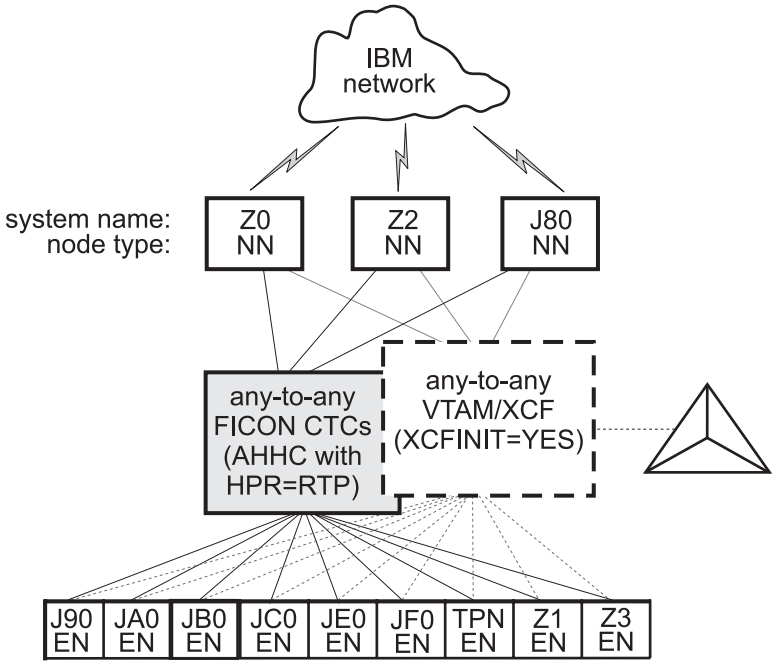


Figure 3. Our VTAM configuration

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 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
- IMS Bridge
- Distributed queueing with 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

Some of the 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: This workload tests the communication between z/OS queue managers using SSL TCPIP 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 EMHQ 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 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.

Chapter 2. About our networking environment

In this chapter we describe our networking environment, including a high-level overview of our configurations and workloads.

Our networking configuration

Figure 4 provides a logical view of our networking configuration.

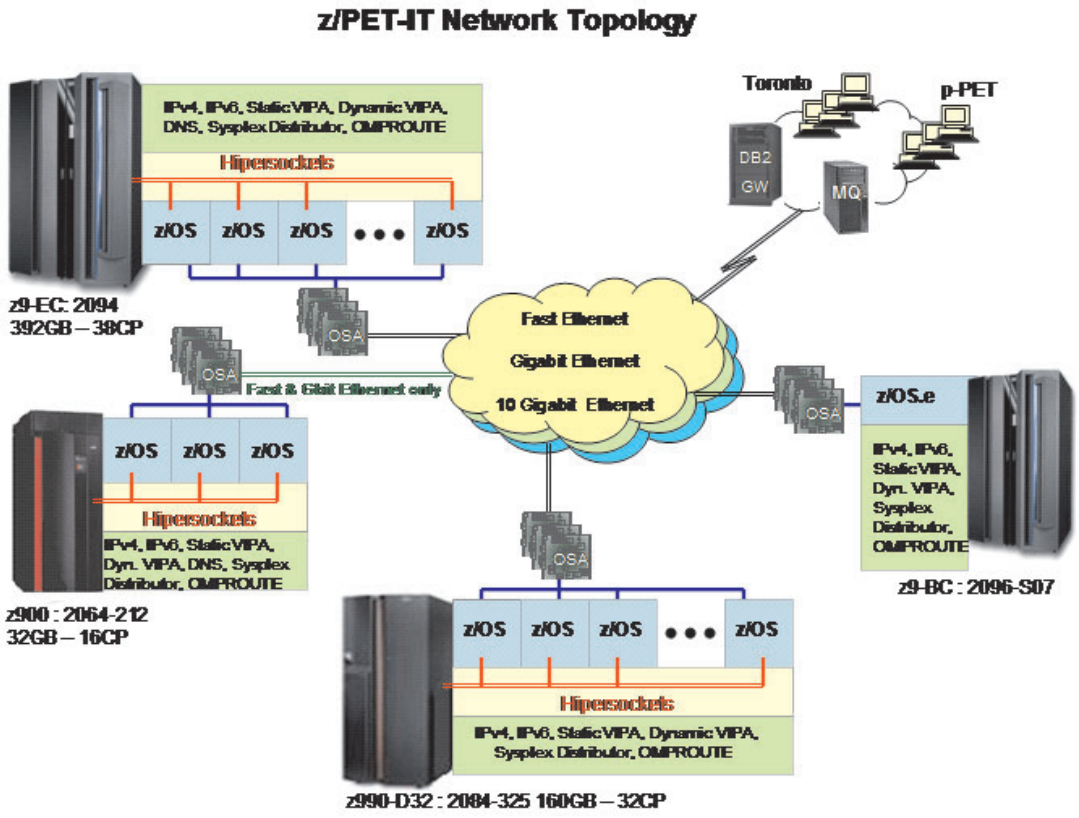


Figure 4. Our networking topology

Configuration overview

Our networking environment is entirely Ethernet. Currently we have Fast Ethernet, Gigabit Ethernet, and 10 Gigabit Ethernet, each running on separate networks. This setup provides a robust environment for our z/OS testing. Across these networks we run workloads that exercise many z/OS components and IBM products.

The following describes what is illustrated in Figure 4.

- We have OSA Fast Ethernet, OSA Gigabit Ethernet, and OSA 10 Gigabit Ethernet configured on three of our 4 CECs. Since our fourth CEC, the z900, does not support the 10 Gigabit OSA feature, we only have the OSA Fast Ethernet and OSA Gigabit Ethernet features configured.
- We use OMPROUTE on each z/OS image to provide dynamic OSPF routing support across our data center.
- We have a DNS setup using master and slave all on z/OS.

- We have dynamic XCF configured so that we can use hypersockets on the CEC's where there is more than one image for communications between those images.
- All of the networks are VLAN Tagged.
- We have a fully implemented IPv6 environment.
- We run many sysplex distributors for workload balancing with a variety of distribution methods using IPv4 and IPv6.

Our IPv6 environment configuration

We currently run a fully implemented IPv6 environment utilizing IPv6 OMPROUTE and DVIPA/Sysplex distributor, This is used to currently support MQSeries and DB2 V9.1 implementations.

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 both 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 TCP/IP stack that supports IPv4 and IPv6.

   We added the following two NETWORK statements to have a TCP/IP 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:

```
VIPADEFINE V6Z2FTP 2003:0DB3:1::2
VIPADISTRIBUTE SYSPLEXPORTS V6Z2FTP PORT 20 21
 DESTIP FEC0:0:0:1:0:168:49:37
```

Note: V6Z2FTP is the INTERFACE name for this VIPA.

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 separate 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.
```

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 allows 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 at *OS/390 Parallel Sysplex Test Report*, and the use of the LAN Server NFS in our December 2004 edition at *zSeries Platform Test Report*. All of our editions can be found at:

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

Networking workloads

For information about our networking 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 5 on page 31: 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 5 on page 31 : 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.

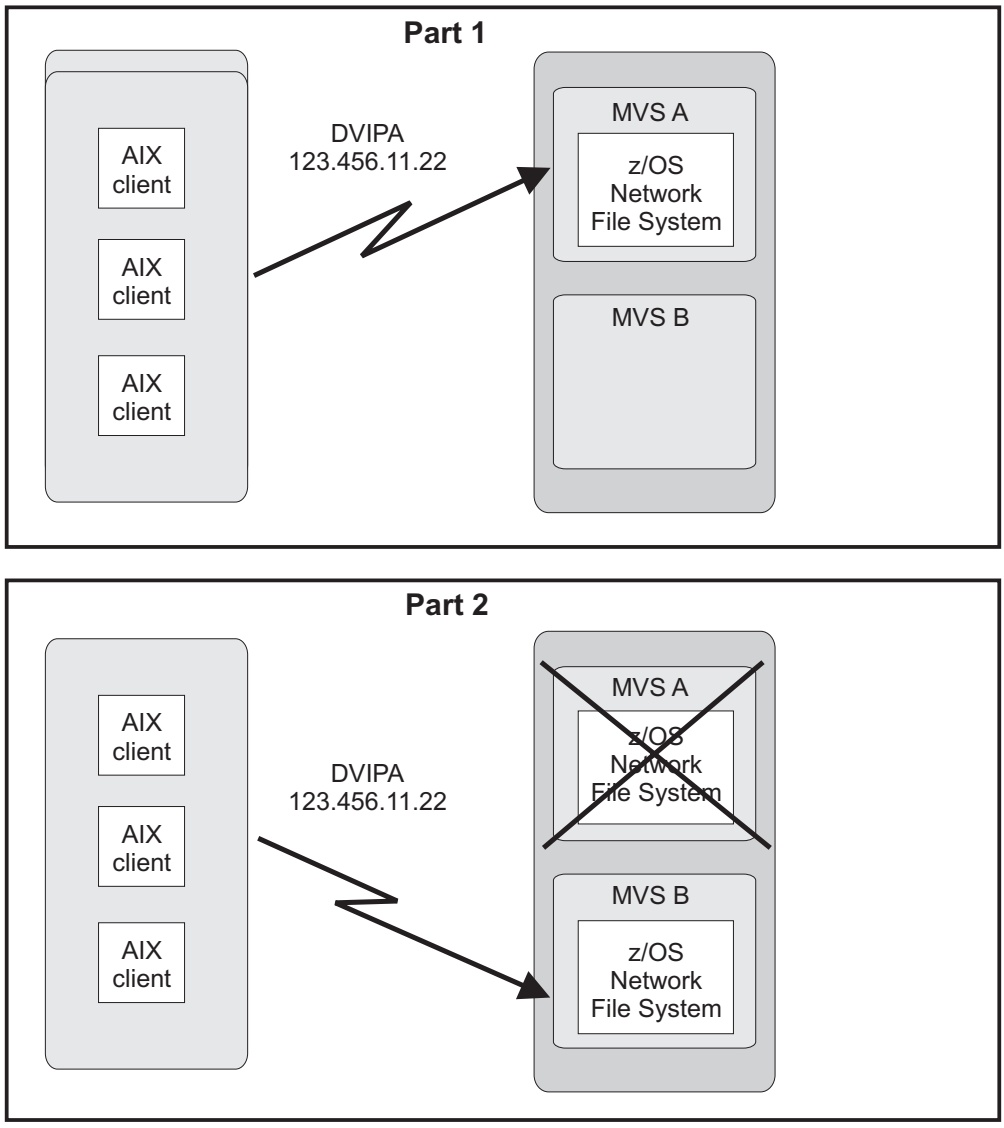


Figure 5. NFS configuration

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 3. About our security environment

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

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 HCR7731, on top of z/OS V1R8. 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 in our environment:

- Crypto Express2 Accelerator (CEX2A)
- Crypto Express2 Coprocessor (CEX2C)
- PCI Cryptographic Accelerator (PCICA)
- PCI X Cryptographic Coprocessor (PCIXCC)
- CP Assist for Cryptographic Functions (CPACF)
- CP Assist for Cryptographic Functions DES/TDES Enablement (CPACF, feature 3863)
- PCI Cryptographic Coprocessor (PCICC)
- Cryptographic Coprocessor Facility (CCF)

Since our goal is to run a customer-like environment, we have various workloads and jobs which take advantage of the products that interface with ICSF (which interfaces with the cryptographic hardware). These products include the following:

- SSL (through WebSphere Application Server, FTP, HTTP, LDAP and CICS)
- Enterprise Key Manager Offering for Tape Encryption
- Encryption Facility for z/OS V1 R1
- Encryption Facility for z/OS V1 R2

We also have an ICSF specific workload that runs daily which exercises the cryptographic services available through the ICSF Callable Services.

Note: For additional information on the Enterprise Key Manager Offering for Tape Encryption and the Encryption Facility for z/OS V1 R2, see "Enterprise Key Manager Offering for Tape Encryption" on page 47. For Encryption Facility

for z/OS V1R1 see the section titled Testing the IBM Encryption Facility for z/OS in our *December 2006 zSeries Platform Test Report*.

Using Kerberos (Network Authentication Service)

Conflict with SDK for z/OS (java)

During the running of our Network Authentication Service (NAS) workload we ran into a problem with the *kinit* command. Here is the message we were seeing after issuing the *kinit* command:

```
com.ibm.security.krb5.KrbException, status code: 0
       message: java.security.PrivilegedActionException: java.io.FileNotFoundException:
/etc/krb5/krb5.conf (EDC5129I No such file or directory.)
```

SDK for z/OS (java) ships many of the kerberos commands, including the *kinit* command, in its /<java directory>/bin directory. The problem ended up being that the .profile PATH variable had been updated to add the SDK for z/OS directory /<java_directory>/bin ahead of the NAS directory /usr/lpp/skrb/bin. This caused the SDK for z/OS *kinit* command to be executed instead of the NAS *kinit* command. Because the SDK for z/OS directory was needed in the .profile PATH variable, we moved the SDK for z/OS directory after the NAS directory in the PATH variable to resolve the problem. The PATH variable should be set to something like the following for proper NAS execution:

```
export PATH=$PATH:/usr/lpp/skrb/bin:/usr/lpp/java/J1.4/bin
```

This is the same type of conflict we reported with DCE in our *z/OS Parallel Sysplex Test Report*.

Caution needs to be taken when enabling Network Authentication Service with SDK for z/OS or DCE.

Access to the Domain Name Server (DNS)

During recent modifications to the locations of our Domain Name Servers it was determined that the Network Authorization Server (NAS) is not enabled to exploit the DNS on a sysplex basis. To align with the z/OS TCPIP stack's resolver the gethostbyaddr/gethostbyname API call needs to be used by NAS when determining the location of the DNS. A Marketing Requirement has been opened to address this issue.

Working with the GLOBALTCPIPDATA setting

Currently NAS accesses the resolver data directly. The default, for NAS, is to evaluate the */etc/resolv.conf* file. If the environment variable RESOLVER_CONFIG is used to specify the location of the resolver data then that will take precedence over the */etc/resolv.conf*.

Although our resolver is enabled to access the DNS on a sysplex basis, we did have */etc/resolv.conf* files defined. For this reason NAS was accessing these */etc/resolv.conf* files on each image within our sysplex where NAS was running. To reduce the need to make updates within each image's */etc/resolv.conf* file, we deployed the use of the RESOLVER_CONFIG environment variable. To determine where to point the RESOLVER_CONFIG environment variable you can take the following steps.

From the system console or SDSF enter the following command.

```
F RESOLVER,DISPLAY
```

The following is what is returned on our systems.

```
EZZ9298I DEFAULTTCPIPDATA - SYS1.TCPPARMS(RESSETUP)
EZZ9298I GLOBALTCPIPDATA - SYS1.TCPPARMS(GLOBAL)
EZZ9298I DEFAULTIPNODES - None
EZZ9298I GLOBALIPNODES - None
EZZ9304I NOCOMMONSEARCH
EZZ9293I DISPLAY COMMAND PROCESSED
```

The dataset and member associated with GLOBALTCPIPDATA is what we specified in our RESOLVER_CONFIG environment variable. This dataset member is what our TCPIP administrator updates instead of the */etc/resolv.conf* file. Using this approach will prevent the NAS administrator from having to be in communication with the TCPIP administrator when they make changes. However, if the dataset member changes that is associated with GLOBALTCPIPDATA than that would need to be communicated so that the RESOLVER_CONFIG variable can be updated.

It is recommended to place the RESOLVER_CONFIG environment variable in the */etc/profile*. Here is an example of how we specified our RESOLVER_CONFIG environment variable.

```
export RESOLVER_CONFIG="//'SYS1.TCPPARMS(GLOBAL)'"
```

Problems with the documentation

We did experience a few problems using *z/OS Integrated Security Services Network Authentication Service Administration*, SC24-5926.

Table 2-3. Environment variables for security runtime: The explanation for RESOLVER_CONFIG in this table states: 'This can be either the name of an HFS file (/hfs-path/hfs-file) or the name of a sequential MVS dataset (//'dataset-name').' However, the MVS dataset does not have to be a sequential dataset. A partitioned dataset will work as well.

To determine the correct format, when specifying the MVS dataset in the RESOLVER_CONFIG environment variable, *z/OS Communications Server: IP Configuration Guide*, SC31-8775 needed to be referenced.

Defining the RESOLVER_CONFIG: It was not clear where to define the RESOLVER_CONFIG environment variable either. In Chapter 2, Configuring Network Authentication Service, the section titled Security runtime environment variables, led us to believe that it should be set in the NAS envar file. Doing so does not allow for any clients, such as LDAP or EIM, that use NAS to pick up the RESOLVER_CONFIG value. However, further along in the same section it is indicated to set RESOLVER_CONFIG in either */etc/profile* or the users *.profile* file.

Using LDAP Server

The 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. For V1R8, there are two versions of the LDAP server available:

1. Integrated Security Services (ISS) Server
2. IBM Tivoli Directory Server (TDS)

This chapter contains the following sections:

• “z/OS IBM Tivoli Directory Server (IBM TDS)” on page 42

Overview of our LDAP configuration

We have a multiplatform LDAP configuration for both the Integrated Security Services (ISS) LDAP environment and the IBM Tivoli Directory Server (IBM TDS) environment. The following figures will illustrate both environments followed by a listing of exploiters of each environment.

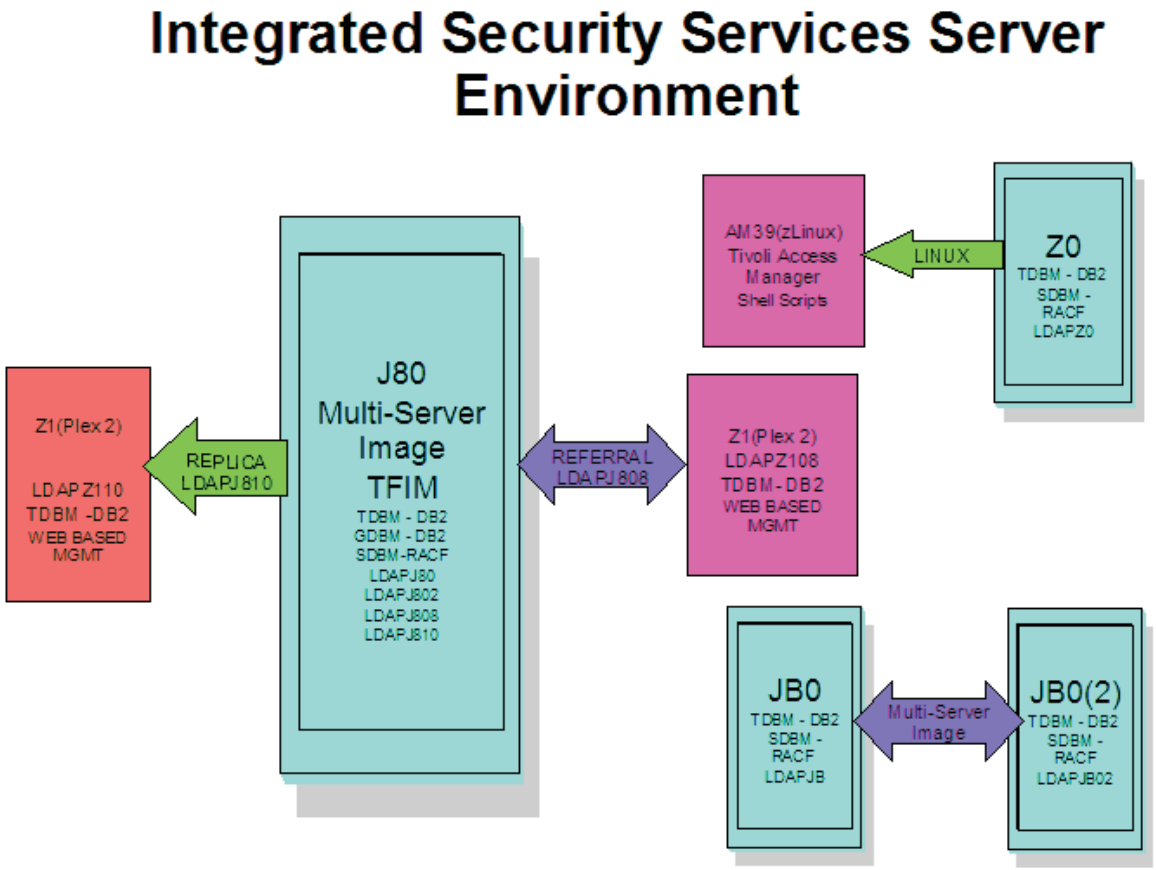


Figure 6. Integrated Security Services (ISS) LDAP environment

DB2 has a TDBM backend, connecting LDAP to the DB2 Database Directory. It also has a GDBM backend. The GDBM backend is used to store Change Log entries created as a result of RACF modifications. These environments are exploited using scripts run from Windows agents that allow stress to be placed on the z/OS LDAP Servers using DB2.

RACF has a SDBM backend, connecting to the RACF directory found on our plex.

Integrated Security Services (ISS) LDAP exploitation

LDAP Referral

This configuration is set up between a LDAP Server on Plex 2 (LDAPZ108) and a LDAP Server on Plex 1(LDAPJ808) both using TDBM backends. The Plex 2 LDAP Server (LDAPZ108) has a general referral found in its configuration file that points to a master LDAP server on z/OS (LDAPJ808).

This allows the user to run the *ldapsearch* command from the LDAP server on Plex 2 for an entry that is not found in that directory, but may be found in the LDAPJ808 master server's directory. The command will return all entries found that match from both directories.

Replication (Master/Slave)
We run our ISS replication transactions between LDAPJ810 and LDAPZ110. Replication functions quite like the Stress operation above but with one important difference. The master receives the new entry and its modifications and eventual deletion. However the slave, which has been initialized like the master, is checked twice, the first time after the entry is added to insure it has been replicated on the slave and also after the deletion to insure it has indeed been deleted from the slave through the replication process. The checking process is repeated until it is either found (during the add) or not found (during the deletion) or the server reaches a specified search count (which causes a failure).

Replication (Peer/Peer)
We run our ISS replication transactions between LDAPJ810 and LDAPZ110. Peer to peer replication functions similarly to master/slave except that each server takes turns at being the ″master″, that is having its entries manipulated by the program while the other server is checked for entry availability. When the program is run in a loop, the ″master″ and ″slave″ switch places on each new loop cycle.

Persistent Search
We run our ISS persistent search transactions between LDAPJ810 and LDAPZ110. The persistent search function detects the revisions that have been made to a server's entries and prints out the results, the detail depending on the display level setting. The program is initiated with the entry filter and operation monitor parameters set and it will listen to the designated server until a specified entry type operation is encountered for reporting. This repeats until the program is terminated. Of course for this function to operate, there must be some activity on the server being monitored. That is one use for the Stress function. An instance of it can be run to stimulate the desired server. Also, the persistent search could be directed against one of the replication servers if desired. For another workload scenario, several instances of the persistent search can be run, with each detecting a different change type (or combination thereof).

Tivoli Access Manager on zLinux
We have set up Tivoli Access Manager (TAM) on our zLinux SUSE 8 machine to enable cross platform testing between Linux and z/OS. TAM uses z/OS LDAP as a backend to store userid information that will either allow or deny user access to TAM. Testing is done using Shell scripts run on Linux that allow stress to be placed on the z/OS LDAP Server on Z0.

Tivoli Federated Identity Manager on zLinux
We have setup Tivoli Federated Identity Manager (TFIM) on our zLinux machine to enable cross platform testing between Linux and z/OS. TFIM uses z/OS LDAP as a backend to store userid information in a similar capacity to TAM. However, our TFIM setup requires the use of two LDAP Servers; LDAPJ80 and LDAPJ802. This environment is exploited using Shell scripts run from Windows agents that allow stress to be placed on the z/OS LDAP Servers on J80.

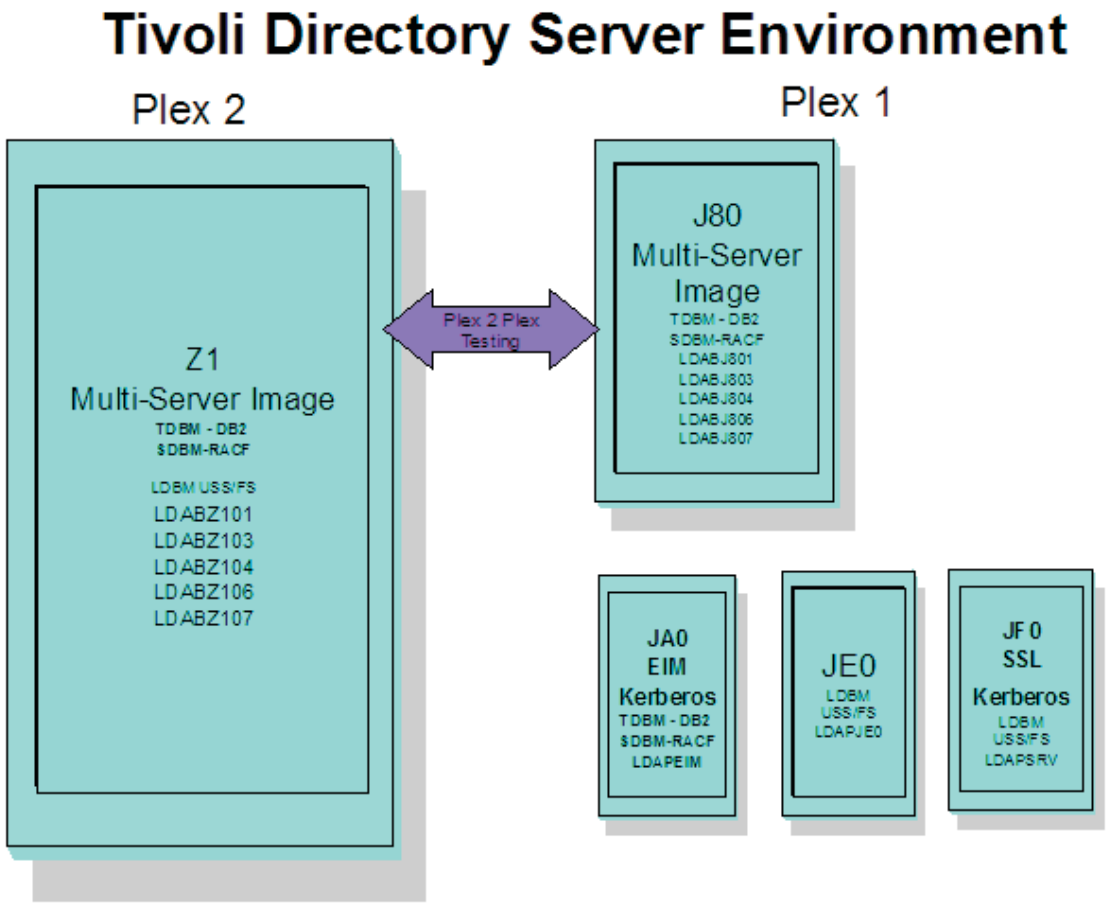


Figure 7. IBM Tivoli Directory Server (IBM TDS) environment

DB2 has a TDBM backend, connecting LDAP to the DB2 Database Directory. This environment is exploited using scripts run from Windows agents that allow stress to be placed on the z/OS LDAP Servers using DB2.

RACF has a SDBM backend, connecting to the RACF directory found on our plex.

Unix System Services file system
has an LDBM backend, connecting to a Unix System Services file system on our plex. This environment is exploited in two ways. The first is with tso http servers. The IBM HTTP Server powered by Domino running one of our z/OS images and Apache running on an xLinux box. Both of these http servers access the LDAPJE0 IBM TDS for authentication to access http resources. The second is to drive Kerberos transactions using shell scripts run from within our USS environment. This workload accesses the LDAPJF0 IBM TDS.

IBM Tivoli Directory Server (IBM TDS) exploitation

Kerberos
We currently have two LDAP servers on our plex that are setup for Kerberos transactions. They are LDAPSRV on JF0 and LDAPEIM on JA0.

EIM We currently have one LDAP server on our plex that is setup for EIM transactions. It is LDAPEIM on JA0.

LDAP Referral

This configuration is set up between a LDAP Servers on Plex 2 and a LDAP Servers on Plex 1 (LDABJ804/LDABJ806) using the TDBM and LDBM backends. The Plex 2 LDAP Servers (LDABZ104/LDABZ106) have a general referral found in its configuration file that points to master LDAP servers on z/OS (LDABJ804/LDABJ806). This allows the user to run the *ldapsearch* command from the LDAP servers on Plex 2 for an entry that is not found in that directory, but may be found in the master servers' directories. The command will return all entries found that match from both directories.

Replication (Master/Slave)

We run our IBM TDS master/slave replication transactions between LDABJ801 and LDABZ101. Replication functions quite like the Stress operation above but with one important difference. The master receives the new entry and its modifications and eventual deletion, but the slave, which has been initialized like the master, is checked twice, the first time after the entry is added to insure it has been replicated on the slave and also after the deletion to insure it has indeed been deleted from the slave through the replication process. The checking process is repeated until it is either found (during the add) or not found (during the deletion) or the server reaches a specified search count (which causes a failure).

Replication (Peer/Peer)

We run our IBM TDS peer/peer replication transactions between LDABJ803 and LDABZ103. Peer to peer replication functions similarly to master/slave except that each server takes turns at being the ″master″, that is having its entries manipulated by the program while the other server is checked for entry availability. When the program is run in a loop, the ″master″ and ″slave″ switch places on each new loop cycle.

Persistent Search

We run our IBM TDS persistent search transactions between LDABJ804 and LDABZ104. The persistent search function detects the revisions that have been made to a servers entries and prints out the results, the detail depending on the display level setting. The program is initiated with the entry filter and operation monitor parameters set and it will listen to the designated server until a specified entry type operation is encountered for reporting. This repeats until the program is killed. Of course for this function to operate there must be some activity on the server being monitored. That is one use for the Stress function. An instance of it can be run to stimulate the desired server. Also, the persistent search could be directed against one of the replication servers if desired. For another workload scenario, several instances of the persistent search can be run, with each detecting a different change type (or combination thereof).

z/OS Integrated Security Services LDAP Server

Incorrectly defined dir_misc table

The *ldapcnf* utility was incorrectly defining the dir_misc table in the DESCTS tablespace. It needed to be defined in the MISCTS tablespace. The spufi samples shipped in GLDHLQ.SGLDSAMP do not have this problem. If the ldapcnf utility was not used the dir_misc table should most likely be associated to the correct tablespace.

A customer reported this problem to z/OS LDAP development. The problem was reported as a performance issue when using the bulkload utility, *ldif2tdbm*. Working

with z/OS LDAP development to help determine the problem we found that some of our LDAP servers had the dir_misc table incorrectly defined as well.

Determining the tablespace of the dir_misc table: To determine the tablespace of the dir_misc table, submit the following select statement.

```
SELECT TSNAME FROM SYSIBM.SYSTABLES WHERE CREATOR='LDAPZ101' AND NAME='DIR_MISC';
```

Note: Change LDAPZ101 to the DB2 Userid for the LDAP server being checked.

If MISCTS is returned then the dir_misc table is in the correct tablespace. If not then it will need to be corrected.

Correcting the dir_misc table tablespace: To correct the tablespace of the dir_misc table reference APAR OA17432. Instructions are provided in this APAR to guide you through the required steps.

z/OS IBM Tivoli Directory Server (IBM TDS)

Migrating from an Integrated Security Services (ISS) LDAP Server

The *IBM Tivoli Directory Server Administration and Use for z/OS*, SC23-5191 was referenced while performing the migration. Chapter 10, Migrating to a z/OS IBM TDS LDAP, details the steps required.

Because IBM TDS is a new LDAP server for z/OS we have been involved with the alpha, beta and ESP programs. This allowed us to work very close with the LDAP development and publications team to help ensure the migration documentation was the best that it could be. We did experience problems, but because of our early involvement we were successful in getting those problems resolved. It is strongly recommended that the time be taken to fully read and understand Chapter 10, Migrating to a z/OS IBM TDS LDAP. Then map out your own unique migration plans based on the information in Chapter 10. This chapter also describes how to fallback to your previous ISS LDAP if required.

If the new LDBM backend will be deployed, we strongly encourage that a unique Unix System Services file system be created for the LDBM.

We did perform the following migrations:

- ISS LDAP Schema to IBM TDS Schema
- ISS LDAP TDBM to IBM TDS TDBM
- ISS LDAP TDBM to IBM TDS LDBM
- ISS LDAP GDBM to IBM TDS GDBM (DB2)
- ISS LDAP GDBM to IBM TDS GDBM (Unix System Services file system)

To migrate the server configuration and started task procedure, we used both the new *dsconfig* utility as well as making the manual updates. Which way you decide to migrate the server configuration and started task will depend on your own preferences. Within our own test group some like the *dsconfig* and others like the doing the updates manually.

Considerations for the dsconfig utility

The following are some considerations for the *dsconfig* utility.

ds.db2.profile concerns: There are two concerns with the *ds.db2.profile* for indicating the buffer pool.

Default value of BP0 being specified: Our DB2 team has indicated that this value should not be used by LDAP. A value of BP0 should be reserved for the DB2 system tables. Using a value of BP0 for LDAP may have a negative effect on performance.

Specify a buffer pool for both the tablespaces and indexes: Currently the ds.db2.profile only allows for the specification of the buffer pool for the tablespace. The buffer pool for the index then defaults to the same value. Having the tablespace and index buffer pool be the same may have a negative effect on performance. The IBM Tivoli Directory Server Administration and Use for z/OS - SC23-5191 Chapter 28 Performance tuning, references separating the tablespace and index buffer pool to improve performance.

To resolve this, the TDBSPUFI member, generated by the *dsconfig* utility, needs to be modified. For each CREATE INDEX entry add the following.

BUFFERPOOL BP*n*

Where *n* is a unique number different from the TableSpace buffer pool.

Here is an example for one index:

```
CREATE UNIQUE INDEX LDAPJ807.DIR_REGISTERX1
    ON LDAPJ807.DIR_REGISTER( ID, SRV )
    USING STOGROUP DBWGUSERSTG
    BUFFERPOOL BP7
    DEFER YES;
```

We have given a requirement to LDAP development to resolve both of these issues in a future release.

Recovering from a TCPIP Outage

If there is a TCPIP outage the ISS LDAP Server would fail as well. With IBM TDS the server can be configured to stay active and recover from a TCPIP outage. To enable IBM TDS to recover from a TCPIP outage the *tcpTerminate* directive in the IBM TDS configuration file must be updated to specify recover.

```
tcpTerminate recover
```

With the recover option specified, for tcpTerminate, the following type of messages will be seen in the IBM TDS servers JES log upon a TCPIP outage. If there is activity against the server these type messages can be expected:

```
061101 15:59:05.194570 GLD1068E Unable to accept connection: 122/00000000 - EDC5122I Input/output error.
061101 15:59:05.212887 GLD1068E Unable to accept connection: 122/00000000 - EDC5122I Input/output error.
061101 15:59:05.213192 GLD1069E Unable to receive data: 122/00000000 - EDC5122I Input/output error.
061101 15:59:05.218138 GLD1068E Unable to accept connection: 122/00000000 - EDC5122I Input/output error.
061101 15:59:05.218480 GLD1069E Unable to receive data: 122/00000000 - EDC5122I Input/output error.
061101 15:59:05.218634 GLD1068E Unable to accept connection: 122/00000000 - EDC5122I Input/output error.
061101 15:59:05.219480 GLD1068E Unable to accept connection: 122/00000000 - EDC5122I Input/output error.
061101 15:59:05.219655 GLD1068E Unable to accept connection: 122/00000000 - EDC5122I Input/output error.
```

The following messages will inform of the loss of TCPIP:

```
061101 15:59:13.000897 GLD1060I No longer listening for requests on 192.xxx.xxx.32 port 636.
061101 15:59:13.001016 GLD1060I No longer listening for requests on 192.xxx.xxx.32 port 389.
061101 15:59:13.001067 GLD1060I No longer listening for requests on 192.xxx.xxx.32 port 636.
061101 15:59:13.001123 GLD1060I No longer listening for requests on 192.xxx.xxx.32 port 389.
061101 15:59:13.001173 GLD1060I No longer listening for requests on 9.xxx.xxx.152 port 636.
061101 15:59:13.001252 GLD1060I No longer listening for requests on 9.xxx.xxx.152 port 389.
```

Once TCPIP is again operational these type messages will be displayed:

```
061101 16:09:13.002881 GLD1059I Listening for requests on 9.xxx.xxx.152 port 389.
061101 16:09:13.004396 GLD1211I Listening for requests on 9.xxx.xxx.152 secure port 636.
061101 16:09:13.006128 GLD1059I Listening for requests on 192.xxx.xxx.32 port 389.
```

```
061101 16:09:13.008058 GLD1211I Listening for requests on 192.xxx.xxx.32 secure port 636.
061101 16:09:13.009689 GLD1059I Listening for requests on 192.xxx.xxx.32 port 389.
061101 16:09:13.010957 GLD1211I Listening for requests on 192.xxx.xxx.32 secure port 636.
```

All of the same type messages as listed in the IBM TDS JES log will be written to the system console except for the GLD1060I messages. Once TCPIP is again operational, normal operations will be successful against the IBM TDS Server. No action will be required on the part of the IBM TDS administrator to get IBM TDS operational.

If the *tcpTerminate* directive specifies the value of terminate then IBM TDS will act the same as the ISS LDAP Server and will terminate with the TCPIP outage. Once TCPIP is again operational the IBM TDS administrator will have to also start IBM TDS. The exception would be if there is an automation program that would automatically restart IBM TDS.

Testing the LDBM backend

The new LDBM backend was successfully tested. The LDBM backend was configured as described in *IBM Tivoli Directory Server Administration and Use for z/OS*, SC23-5191. Once the LDBM backend was configured, we successfully loaded the schema followed by the loading of data. Data was loaded into LDBM using the *ldapadd* command. The full suites of LDAP Workloads were then configured to run against the new backend. The workloads emphasized user adds, deletes, and modifies against the LDBM backends. Change Log support for LDBM was also added to the IBM TDS server. This support was successfully tested by configuring an LDBM backend to be the storage point for change log entries. The setup was then tested by modifying several RACF user ids that would then create change log entries into the LDBM backend. The *ldapsearch* command was used to verify the contents of the database.

Testing Arm support

IBM TDS allows support for Automatic Restart Management (ARM). This support was tested by enabling the new *armName* option in the *ds.conf* file. We enabled this function in the *conf* file as follows:

```
ARMNAME LDAP1
```

This created the element name LDAP1_*systemname*, where *systemname* refers to the image that the ARM support will be enabled on. To verify the functionality we updated the *ds.conf* file and recycled the IBM TDS server with the changes. Arm support was verified by looking at the DSOUT dataset and seeing the following:

```
ARMNAME:GLDSRVR
```

Testing Sysplex support

Sysplex Support has been added to the IBM TDS server to allow improved workload performance in a Sysplex environment. For this support to be fully utilized, the TCP/IP Sysplex distributor needs to be appropriately configured. Once configured, the IBM TDS servers were then setup to share the same TDBM backend. Three IBM TDS servers were configured as such for this scenario. The *ds.conf* file was also modified for each server to include the serverSysplexGroup option in the configuration file. Refer to configuration options in IBM Tivoli Directory Server Administration and Use for z/OS documentation for serverSysplexGroup. Once the updates were made to the server it was then recycled and successfully brought up in Sysplex Support mode. To verify that everything functioned properly, the full suite of IBM TDS workloads were run against this setup. An LDAP server was also randomly dropped from the setup to verify the functionality of the Sysplex distributor. The transactions that were being sent to the removed server were properly routed to the functional servers.

Testing Interop functionality

We exercised IBM TDS with several other products. These next sections will describe that testing.

Testing with HTTP servers: The IBM HTTP Server (IHS) running on z/OS and an Apache Server running on Linux has been enabled to access IBM TDS for authentication. For both IHS and Apache two types of authentication were used. The first is where the userid and password are both in IBM TDS and used for authentication when prompted. The second is where the userid is in IBM TDS and is associated to a RACF userid. In this case when the prompt is given to enter the authentication data the userid listed in IBM TDS is entered but the associated RACF userid's password is entered. This password is listed in RACF and not in IBM TDS.

For IHS, the TCP and SSL Transport were enabled for communication between IHS and IBM TDS. For Apache, only the TCP Transport has been enabled for communication between Apache and IBM TDS.

We did find a compatibility problem between the IBM TDS server and the ISS LDAP server when communicating with IHS. It was resolved and the interface is working as expected.

Testing with Kerberos: The IBM TDS client and the EIM client were enabled to authenticate to the IBM TDS Server using Kerberos. A problem was found between the EIM client and IBM TDS. It has been resolved and the Kerberos authentication between the EIM client and IBM TDS server is now working as expected. No problems were identified using Kerberos authentication between the IBM TDS client and IBM TDS Server.

Testing with Enterprise Identity Mapping (EIM): We ran our suite of EIM tests and workloads with IBM TDS. We did find a DB2 Deadlock condition with one of our workloads. The solution was to change the DB2 locksize on the IBM TDS's dir_search table from ANY to ROW. This resolved the DB2 Deadlock problem.

Testing with Tivoli Access Manager on z/Linux: We configured the IBM TDS server TDBM and LDBM backend with Tivoli Access Manager 6.0 on z/Linux. We used a new database containing no previous TAM data and successfully configured TAM with IBM TDS. Once configuration was complete, we ran our full suite of TAM workloads that consisted of TAM user adds, deletes, and modifies. This placed ample stress on the IBM TDS server. All workloads ran successfully.

Testing with Tivoli Federated Identity Manager on z/Linux: We configured the IBM TDS server TDBM backend with Tivoli Federated Identity Manager 5.1 on z/Linux. We used a new database for this config also containing no previous TFIM data and successfully configured TFIM with IBM TDS. To exploit this setup we executed workloads that simulated Single Sign Functionality by attempting to log on to a designed sample website using various forms of ids. These ids were designed to be valid, invalid, and expired to test the full functionality of the setup. All workloads ran successfully.

Testing replication functionality

Replication was rigorously tested as part of the regression (ISS) and development (IBM TDS) code testing process. TDBM (DB/2 based) and LDBM (file based) backends were both setup in the following configuration:

```
CODE     LDAP server (Plex 1)  <-----> LDAP Server (Plex 2)
----     --------------------        --------------------
ISS          Master TDBM       <----->     Slave TDBM

ISS          Peer TDBM         <----->     Peer TDBM

IBM TDS      Master TDBM       <----->     Slave TDBM

IBM TDS      Peer TDBM         <----->     Peer TDBM

IBM TDS      Master LDBM       <----->     Slave LDBM

IBM TDS      Peer LDBM         <----->     Peer LDBM
```

Configuration: This section initially focuses on the Master/Slave TDBM setup with deviations covered later for the Peer servers and LDBM backend. The standard setup as described in *IBM Tivoli Directory Server Administration and Use for z/OS*, SC23-5191 was followed.

Initially the servers were setup and run using the ISS configuration utilities but were eventually migrated to operate with the IBM TDS code Rendition. More detail on the migration process follows.

The initial configuration of the servers did not include replication. This is done to insure the basic operation of the servers is correct before proceeding onto the replication process. The configuration files were modified to include the system specific information as per the parameter definitions. After the files were processed and the DB2 TDBSPUFI and RACF files were processed, the servers were started. Once the schema files were loaded, the servers were now ready to be prepared for replication.

First, each server was loaded with base information, consisting minimally of the base suffix entry, the LDAP Administrator entry, and a Replication Administrator entry (different from the LDAP administrator). The slave server (replica) process was shutdown and the configuration file modified to include the master server information (server name/port, master server DN, and its password). After including this information, the server was restarted. Replication will not occur until the replica object is added to the master server. This entry is created using the options desired as described in *IBM Tivoli Directory Server Administration and Use for z/OS*, SC23-5191. Once the entry is added, replication is enabled. Now it is a simple matter of adding information to the Master server which will be reproduced on the replica server.

Essentially this same process was used to create all the replication databases. Peer/Peer was configured slightly different. After the plain vanilla servers were started, and schemas and base entries added, they were both shutdown and the configuration files modified with peer server information. This included peer server DN and password, similar to the way the replica was configured previously. They were both restarted and a replica object was added to each server which effectively pointed to the other server. Replication then becomes active and the data can now be added to either server.

The same process was applied to a set of servers which were configured with LDBM backends. Once the servers are established, it is transparent to the replication setup process as to the backend implemented.

Testing

The following sections describe our testing of replication functionality.

General: A multipurpose 'C' program which utilized the LDAP API's was instituted to check various operational characteristics of the server code. Test specifics were controlled by inputting various parameters when calling the test program. Some of the functions tested were:

- Aliasing
- Referrals
- Persistent Search with a Stressing Driver
- Replication
- Performance Monitor

Replication testing will be discussed in more detail.

Replication testing: A similar replication testing process was applied to all the replication servers. The basic test consisted of synthesizing new entries from the existing entries. These new entries were processed by:

- adding them to one server
- checking for its existence on the other server
- modifying their attributes (adding one, changing one, and deleting one)
- checking that the changes were incorporated correctly
- deleting the entry
- checking for the deletion on the other server

Variations were made on this theme on each of the servers for additional test coverage. They are:

- With Peer/Peer servers, the servers took turns being updated directly by the program with the other being updated via the replication process.
- Also, an entry mode was provided which allowed each step above to be applied sequentially to one entry at a time (i.e. the new entry was created, modified and deleted before proceeding to the next) or in block mode (all entries added, all modified, and so on.)
- Some replication was accomplished with SSL connections.

Enterprise Key Manager Offering for Tape Encryption

z/OS Integration Test implemented and tested the new Tape Encryption offering which uses the enhanced 3592 Model E05 tape drive in conjunction with the new Encryption Key Manager software. Our goal was to create a high availability environment for the Encryption Keystore. We utilized shared HFS for the filesystems, shared PKDS in ICSF and Sysplex Distributor for workload distribution. All ten z/OS V1R8 systems in our zPET parallel sysplex had EKM's defined, with all of them listening on the same IP/port.

With this Encryption solution, data to be encrypted is sent to the drive in the clear (as it is today in a non-encrypted form) with encryption and decryption of the data taking place outboard in the tape drive. The Enterprise Key Manger (referred to as EKM for the remainder of this document), a Java™ based application, handles the key requests. This program has been designed to run on many different operating system platforms and is able to use different types of keystores. The four supported keystore types for z/OS are:

1. JCEKS
2. JCE4758KS
3. JCE4758RACFKS

4. JCERACFKS

In our testing on z/OS Integration Test, we configured EKM and tested both JCE4758RACFKS and JCE4758KS keystores. As for Java™ levels, there are two minimal releases that are supported. JDK 1.4.2. SR6 31bit is the level we used for our testing, however, JDK 5.0 SR3 is also supported. We also used a product which IBM recently acquired called JZOS. A new Java ™ class was created using JZOS so that the EKM application can run as a started task on z/OS systems with an operator command interface. The following steps are what we used to implement Tape Encryption on our environment and the experiences we encountered.

It is strongly recommended that you take a look at document *z/OS DFSMS Software Support for IBM System Storage TS1120 Tape Drive (3592), document number SC26-7514*. Specifically, section 1.2.3 "Installing the appropriate PTFs" which documents which enabling PTFs are needed for the various levels of z/OS.

The following are the steps we took in installing and testing the Enterprise Key Manager Offering for Tape Encryption:

Installing the enabling PTF

In order to take advantage of the Tape Encryption Solution on z/OS V1R8, we needed to install the PTFs for enabling APAR OA17562. Installation of the enabling PTF pulls in all of the other support PTFs. Please refer to this APAR when planning your install. Also there are RMM apars for compatibility and new function. Depending on your current RMM setup, you may need to stage the install of the apars for toleration aspects so we suggest you review this thoroughly before installing any apars.

It is strongly recommended that you take a look at document *z/OS DFSMS Software Support for IBM System Storage TS1120 Tape Drive (3592), document number SC26- 7514*. Specifically, section 1.2.3 "Installing the appropriate PTFs" which documents which enabling PTFs are needed for the various levels of z/OS.

Preparing the filesystems

We first created various zFS filesystems and mount points (directories) that are needed for JDK download, EKM logs and EKM configuration files. Since our sysplex takes advantage of shared HFS in Unix System Services, all the filesystems are shared. We used system names for directories within /ekmlogs and /ekmetc. We defined the JDK filesystem at 500cyls and mounted it at directory /javaekm. We then created a filesystem for the EKM error & debug logs at 500cyls and mounted it at /ekmlogs. The filesystem for /ekmetc only needs to be 2 cyls..

Downloading JDK, EKM and JZOS

The next step was to download using FTP the JDK 1.4.2 sr6 from IBM website: http://www.ibm.com/servers/eserver/zseries/software/java . The downloaded file was named UK17593.PAX.Z and the following command was used to unpax the file.

```
pax -ppx -rvzf UK17593.PAX.Z
```

You can verify that the unpax was successful by issuing a java version command. We also had to set a large region size for our TSO logon ID or we received the following error:

```
131:/javaekm/J1.4/bin $ java -version
Could not load dll : /javaekm/J1.4/bin/libjitc.so
: EDC5157I An internal error has occurred.
Warning: JIT compiler "jitc" not found. Will use interpreter.
java version "1.4.2"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2)
Classic VM (build 1.4.2, J2RE 1.4.2 IBM z/OS Persistent Reusable VM build cm142-
20060824 (SR6) (JIT disabled))
```

After we logged on with a size of 50000, the java –version command worked.

```
129:/javaekm/J1.4/bin $ java -version
java version "1.4.2"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2)
Classic VM (build 1.4.2, J2RE 1.4.2 IBM z/OS Persistent Reusable VM build cm142-
20060824 (SR6) (JIT enabled: jitc))
130:/javaekm/J1.4/bin $
```

If you get the following error:

```
JVMHP030: Unable to switch to IFA processor - issue "extattr +a libhpi.so"
```

It has to do with the way we executed the pax command. It did not keep all the file attributes upon inflating the JDK pax file. This error means that the JDK does not have authority to run on the zAAP processor. To fix it, issue from a UID(0) ID:

```
extattr +a $JAVA_HOME/bin/libhpi.so
```

Once the JDK is installed there are other customization steps you need to take in customizing Java. The EKM Jar file, IBMKeyManagementServer.jar, is shipped in the JDK download. While EKM is delivered in the JDK, the code is continually being enhanced so getting the latest GA code is advisable. This jar file is located in the JDK at /$JAVA_HOME/J1.4/lib/ext . The download site to obtain the latest version of the EKM code is:

```
http://www.ibm.com/support/docview.wss?rs=0&dc=D400&q1=ekm&uid=ssg1S4000504&loc=en_US*cs=utf-8ç=us&lang=en
```

Once you have downloaded the latest EKM code, copy this file to the JDK. We downloaded to a temporary filesystem and then copied to the JDK. An example of the copy command is:

```
cp /JA0/tmp/IBMKeymanagerServer.jar $JAVA_HOME/lib/ext
```

Another step to perform is to obtain the JZOS jar file which is needed for running EKM as a started task. The jar file is called jzosekm.jar. In future deliverables of JDK this file will be shipped. For this JDK maintenance level, UK17593, this jar file needs to be downloaded using FTP from the following website:

```
http://www.ibm.com/support/docview.wss?rs=0&dc=D400& q1=ekm&uid=ssg1S4000504&loc=en_US&cs=utf-8ç=us&lang=en
```

The JZOS EKM files need to be extracted. Enter the command:

```
pax -rf JZOSEKMFiles.pax.Z
```

then copy the .jar file accordingly

```
cp jzosekm.jar $JAVA_HOME/lib/ext
```

Creating the userid and defining the started task

A userid needs to be created and defined to use Unix System Services. This ID will be used by EKM when running as a started task. It also will be used when creating certificates in the keystore. We choose to use *ekmserv* and defined it to RACF using the adduser command with OMVS and TSO segments. We used a non-zero UID. Just remember that *ekmserv* needs to own the configuration, or debug and error files or permission errors will be encountered when starting EKM. Once the ID is defined we added a .profile in */u/ekmserv* which contained the following export statement.

```
export PATH=/javaekm/J1.4/bin:$PATH
```

You do not have to have a separate JDK for EKM and can utilize a JDK filesystem along with other java™ applications in your system. We just decided that we would have our EKM defined to use its own mounted JDK filesystem.

Next we needed to prepare for a started task. We chose to use EKM as the started task name which would use the RACF authority for *ekmserv*. The following 2 RACF commands will be needed to be entered by an administrator who has SPECIAL attribute. When a task named EKM* is started, then the *ekmserv* RACF userid will be assigned to that task and all access authorities existing for *ekmserv*. You will need to issue the refresh command on each system where EKM is to be started.

```
RDEFINE STARTED EKM*.* STDATA(USER(EKMSERV) GROUP(SYS1)
TRACE(YES))
SETROPTS RACLIST(STARTED) GENERIC(STARTED) REFRESH
```

Create a new member in your systems proclib for EKM. The following is an example of the started task JCL which we used.

```
//EKM PROC JAVACLS='com.ibm.jzosekm.EKMConsoleWrapper',
//   ARGS=,                            < Args to Java class
//   LIBRARY='JZOS.LOADLIB',           < STEPLIB FOR JVMLDM module
//   VERSION='14',                     < JVMLDM version: 14, 50, 56
//    LOGLVL='+T',                     < Debug LVL: +I(info) +T(trc)
//   REGSIZE='0M',                     < EXECUTION REGION SIZE
//    LEPARM=''
//EKM  EXEC PGM=JVMLDM&VERSION,REGION=&REGSIZE,
//     PARM='&LEPARM/&LOGLVL &JAVACLS &ARGS'
//STEPLIB  DD DSN=&LIBRARY,DISP=SHR
//SYSPRINT DD SYSOUT=*          < System stdout
//SYSOUT   DD SYSOUT=*          < System stderr
//STDOUT   DD SYSOUT=*          < Java System.out
//STDERR   DD SYSOUT=*          < Java System.err
//CEEDUMP  DD SYSOUT=*
//ABNLIGNR DD DUMMY
//*******************************************************************
//*
//*The following member contains the JVM environment script
```

```
//*
//*  xxx is for the system name
//*
//*"/ekmetc/xxx/KeyManagerConfig.properties.jce4758racfks"
//*
//*********************************************************************
//STDENV DD DSN=EKMSERV.ENCRYPT.CONFIG(&SYSNAME.),DISP=SHR
//
```

Preparing EKM for running as a started task using JZOS

On z/OS, to run EKM as a started task, you need to use JZOS, which is a Java Launcher tool that IBM recently acquired. To use this you will need to define a JZOS linklisted dataset and perform a few copies as shown below. We defined dataset JZOS.LOADLIB and added to the systems sys1.parmlib(lnklst00). This dataset is a PDSE and is (1,1) CYLs. We then issued the cp command below to copy the loadmod from the JDK filesystem to the PDSE dataset.

```
cp –X JVMLDM14 "//'JZOS.LOADLIB(JVMLDM14)'"
```

Creating the configuration shellscript

The next step is to create the PDS data named EKMSERV.ENCRYPT.CONFIG, which is specified in the started task JCL. We created a system named member in this dataset and then added the JZOS Java launcher script that will invoke the EKM.

The following is an example of contents we use in 'EKMSERV.ENCRYPT.CONFIG(JA0)':

```
# This is a shell script which configures
# any environment variables for the Java JVM.
# Variables must be exported to be seen by the launcher.
. /etc/profile
export JAVA_HOME="/javaekm/J1.4"
export PATH=/bin:"${JAVA_HOME}"/bin:
LIBPATH=/lib:/usr/lib:"${JAVA_HOME}"/bin
LIBPATH="$LIBPATH":"${JAVA_HOME}"/bin/classic
7
export LIBPATH="$LIBPATH":
# Customize your CLASSPATH here
CLASSPATH=${JAVA_HOME}/lib
CLASSPATH=/u/ekmserv:$CLASSPATH
export CLASSPATH="$CLASSPATH":
# Set JZOS specific options
export ZZZZ="/JA0/etc/KeyManagerConfig.properties.jce4758racfks"
export XXXX="com.ibm.keymanager.KMSAdminCmd"
export JZOS_MAIN_ARGS="$XXXX $ZZZZ"
# Configure JVM options (if any)
IJO="-Djava.protocol.handler.pkgs=com.ibm.crypto.hdwrCCA.provider"
export IBM_JAVA_OPTIONS="$IJO "
#export JAVA_DUMP_HEAP=false
#export JAVA_PROPAGATE=NO
#export IBM_JAVA_ZOS_TDUMP=NO
```

Copying unrestricted policy files

Another item that we had to do during the setup was to replace the US_export_policy.jar and local_policy.jar files in our $JAVA_HOME/lib/security directory with an unrestricted version of these files. These unrestricted policy files are required by EKM in order to serve AES keys. The preferred method to do this on z/OS is to copy the unrestricted policy files that are shipped in the z/OS Java SDK build under the jce demo directory. If you do not do this, you will find that you can encrypt a tape successfully but then find that you cannot decrypt the tape.

You only need to copy them to the lib/security directory as shown in this example:

```
/$JAVA_HOME/J1.4/demo/jce/policy-files $ cp unrestricted/* /javaekm/J1.4/lib/security
```

Creating the EKM configuration file

The next task is to create the configuration file that EKM needs. Since we use Shared HFS for filesystem sharing, we placed the files under the system name within /ekmetc. In this example is when we used the JCE4758RACFKS keystore. The configuration file is called KeyManagerConfig.properties.jce458racfks and it is located in directory /ekmetc/JA0 (JA0 is the system we are running on). We suggest reviewing *IBM Encryption Key Manager component for the Java platform "Introduction, Planning & User's Guide" (GA76-0418)* located at:

```
ftp://ftp.software.ibm.com/storage/Encryption/a7641803.pdf
http://www.ibm.com/support/docview.wss?rs=0&dc=D400&q1=ekm&uid=ssg1S4000504&loc
```

for specifics concerning each of the properties settings. One item to note is that you do not want to update the properties setting once the EKM program is running as changes made to the file will be lost when the EKM is shutdown. EKM writes back to this file upon shutdown and will update the settings that may have been made dynamically when it was running.

The following is an example of the contents of our EKM configuration file located at: /ekmetc/JA0/KeyManagerConfig.properties.jce4758racfks:

```
TransportListener.ssl.port = 1443
TransportListener.tcp.port = 3801
fips = Off
Admin.ssl.keystore.name = safkeyring://EKMSERV/KeyStore4758
config.keystore.provider = IBMJCE4758
config.keystore.password = password
TransportListener.ssl.clientauthentication = 0
TransportListener.ssl.ciphersuites = JSSE_ALL
Audit.handler.file.size = 10000
zOSCompatibility = true
drive.acceptUnknownDrives = true
TransportListener.ssl.truststore.name = safkeyring://EKMSERV/KeyStore4758
Audit.handler.file.directory = /ekmlogs/JA0/audit
TransportListener.ssl.protocols = SSL_TLS
config.keystore.file = safkeyring://EKMSERV/KeyStore4758
TransportListener.ssl.truststore.type = JCE4758RACFKS
debug.output = simple_file
TransportListener.ssl.keystore.name = safkeyring://EKMSERV/KeyStore4758
TransportListener.ssl.keystore.password = password
TransportListener.ssl.truststore.password = password
Audit.event.outcome.do = success,failure
Audit.eventQueue.max = 0
debug.output.file = /ekmlogs/JA0/debug
TransportListener.ssl.keystore.type = JCE4758RACFKS
Audit.handler.file.name = kms_audit.log
config.keystore.type = JCE4758RACFKS
requireHardwareProtectionForSymmetricKeys = true
Audit.event.types.backup = authentication,authorization,data
synchronization,runtime,audit management,authorization terminate,configuration
management ,resource management,none
drive.default.alias2 = Tape_Sol_Tst_Shr_Pvt_2048_Lbl_03
drive.default.alias1 = Tape_Sol_Tst_Shr_Pvt_1024_Lbl_02
Audit.event.outcome = success,failure
debug = all
Audit.event.types = all
config.drivetable.file.url = FILE:/JA0/etc/filedrive.table
Admin.ssl.truststore.name = safkeyring://EKMSERV/KeyStore4758
```

Creating RACF permissions for creating and managing Certificates

RACF permissions need to be created for creating and managing Certificates. If you already have these classes defined then skip down to the permit step:

```
RDEFINE FACILITY IRR.DIGTCERT.ADD UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.ADDRING UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.DELRING UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.CONNECT UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.REMOVE UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.LIST UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.ALTER UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.DELETE UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.GENCERT UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.GENREQ UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.EXPORT UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.EXPORTKEY UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.MAP UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.ALTMAP UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.DELMAP UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.LISTMAP UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.ROLLOVER UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.REKEY UACC(NONE)

PERMIT IRR.DIGTCERT.ADD CLASS(FACILITY) ID(EKMSERV)
ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.ALTER CLASS(FACILITY) ID(EKMSERV)
ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.DELETE CLASS(FACILITY) ID(EKMSERV)
ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(EKMSERV)
ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.ADDRING CLASS(FACILITY) ID(EKMSERV)
ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.DELRING CLASS(FACILITY) ID(EKMSERV)
ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(EKMSERV)
ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.CONNECT CLASS(FACILITY) ID(EKMSERV)
ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.REMOVE CLASS(FACILITY) ID(EKMSERV)
ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.MAP CLASS(FACILITY) ID(EKMSERV)
ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.LISTMAP CLASS(FACILITY) ID(EKMSERV)
ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.ALTMAP CLASS(FACILITY) ID(EKMSERV)
ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.DELMAP CLASS(FACILITY) ID(EKMSERV)
ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.REKEY CLASS(FACILITY) ID(EKMSERV)
ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.ROLLOVER CLASS(FACILITY) ID(EKMSERV)
ACCESS(CONTROL)
```

Setting up TCPIP for EKM

To ensure that key requests would always be obtainable when requested from either one of our systems or another in our test environment, we chose to configure TCPIP for higher availability by using Dynamic VIPA and Sysplex Distributor.

Configuring the EKM TCPIP Profile

The following is required in the TCPIP Profile for all images that are running EKM,

Ports 3801 and 1443 must be reserved in the PORT section

```
PORT
1443 TCP EKM ; Key Manager SSL
3801 TCP EKM ; Key Manager
```

If the port is already being used, the option SHAREPORT must be added to each of the applications using the port. We already had port 1443 in use for IMWEB. This is an example of the setting we used:

```
PORT
1443 TCP EKM  SHAREPORT               ; Key Manager
1443 TCP IMWEB SHAREPORT           ; Web server
```

EKM can also be optionally be started when TCPIP starts up by adding it to the AUTOLOG section. An example of the configuration file is as follows:

```
AUTOLOG
 EKM  ;Key Manager
ENDAUTOLOG
```

Configuring the SYSPLEX Distributor and Dynamic VIPA

What we configured in zPET is a Sysplex Distributor using distributed DVIPA's for higher availability.

In zPET we have a 10-way sysplex with EKM running on all of the images. The Sysplex Distributor distributes each key request to an image based on a configurable distribution method such as; WLM, ROUNDROBIN, and so on. In the case of a stack failure, it will move the Sysplex Distributor to another image where the requests will be handled. The 'backup' stack is configured in the tcpip profile. If one of the images where the key requests are being sent fails, the Sysplex Distributor will stop sending request to that image until it is back up.

Sysplex Distributor setup and Dynamic Vipa configuration is well documented in the following Communications Server manuals.

- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IP Configuration Guide*, SC31-8775

A few steps to get a Sysplex Distributor set up.

1. Dynamic XCF is needed. This is the path that is used for the requests thru the sysplex distributor.

   ```
   IPCONFIG DYNAMICXCF 192.168.49.30 255.255.255.0 3
   ```

   This needs to be done on all images where the Sysplex Distributor will be distributing to or will be a backup to the Sysplex Distributor.
2. Define the required DVIPA parameters in the VIPADYNAMIC section of the profile.

   VIPADYNAMIC

   Sysplex Distributor configuration.

   ```
   VIPADEFINE MOVEABLE IMMED 255.255.255.0 9.xx.xx.xxx
   VIPADISTRIBUTE DEFINE SYSPLEXP DISTM ROUNDROBIN 9.xx.xx.xxx PORT 3801 1443
   DESTIP ALL
   ENDVIPADYNAMIC
   ```

   This is saying that for all requests that come to DVIPA address 9.xx.xx.xxx, Port 3801 or Port 1443, send them to ALL images in the sysplex that are configured to accept requests (Dynamic XCF setup).

3. For the images that you will configure as backup, you'll need Dynamic XCF, EKM running and the following section which tells the sysplex what DVIPA's you are backing up.

```
VIPADYNAMIC
VIPABACKUP 10 9.xx.xx.xxx
ENDVIPADYNAMIC
```

Setting up shared keys in ICSF

We mainly used the JCE4758RACFKS keystore in our testing. The following steps were used to setup our ICSF environment using RACF. We had a set of predefined shared keys which were created by another test group. Here are the steps we took to setup ICSF to use the EKM with a JCERACFKS keystore.

For additional information, please see *IBM Encryption Key Manager component for the Java platform "Introduction, Planning & User's Guide" (GA76-0418)* at

`http://www.ibm.com/support/docview.wss?rs=0&dc=D400&q1=ekm&uid=ssg1S4000504&loc`

Note: Since our EKM runs under ID TAPEKMS, we issued all our RACF commands logged on to that ID. Specifying ID(TAPEKMS) on the RACF commands will also work.

1. We first configured ICSF to use a shared PKDS called SYS1.PKDSPLX2 across all systems in our sysplex. (See the *z/OS Cryptographic Services ICSF System Programmer's Guide*, SA22-7520 for information on setting up a shared PKDS).
2. We created KeyRing: RACDCERT ADDRING(KeyStore4758) ID(TAPEKMS)
3. We issued the following to get the cert/label into the PKDS and the RACF database:

```
RACDCERT ADD('EKMSERV.TAPESOL.TSTSHR.PVT1024.LBL01') PASSWORD('password') TRUST
WITHLABEL('tape_sol_tst_shr_pvt_1024_lbl_01') ICSF

RACDCERT ADD('EKMSERV.TAPESOL.TSTSHR.PVT1024.LBL02') PASSWORD('password') TRUST
WITHLABEL('tape_sol_tst_shr_pvt_1024_lbl_02') ICSF

RACDCERT ADD('EKMSERV.TAPESOL.TSTSHR.PVT2048.LBL03') PASSWORD('password') TRUST
WITHLABEL('tape_sol_tst_shr_pvt_2048_lbl_03') PCICC

RACDCERT ADD('EKMSERV.TAPESOL.TSTSHR.PVT2048.LBL04') PASSWORD('password') TRUST
WITHLABEL('tape_sol_tst_shr_pvt_2048_lbl_04') PCICC
```

   Because these certificates were created as self-signed, all of the above cmds generate the message: Certificate Authority not defined to RACF. Certificate added with TRUST status.
4. There is a Java restriction in that you need to have a CA in the keyring. This is planned to be fixed in a future release. So we added a DUMMY CA:

```
RACDCERT CERTAUTH GENCERT SUBJECTSDN(CN('DummyCertAuthCert')
C('US')) WITHLABEL('DummyCertAuthCert')  TRUST
RACDCERT CONNECT (RING(KeyStore4758) LABEL('DummyCertAuthCert')
CERTAUTH)
```

5. We CONNECTed the shared keys to the keyring:

```
RACDCERT ID(EKMSERV) CONNECT
(LABEL('tape_sol_tst_shr_pvt_1024_lbl_01') RING(KeyStore4758)
USAGE(PERSONAL))
RACDCERT ID(EKMSERV) CONNECT
(LABEL('tape_sol_tst_shr_pvt_1024_lbl_02') RING(KeyStore4758)
USAGE(PERSONAL))
RACDCERT ID(EKMSERV) CONNECT
(LABEL('tape_sol_tst_shr_pvt_2048_lbl_03') RING(KeyStore4758)
```

```
    USAGE(PERSONAL))
    RACDCERT ID(EKMSERV) CONNECT
    (LABEL('tape_sol_tst_shr_pvt_2048_lbl_04') RING(KeyStore4758)
    USAGE(PERSONAL))
```

Setting up the DFSMS Environment

We defined a data class to request the encryption. An encryption-capable TS1120 tape drive recorded in (enterprise encrypted format 2 (EEFMT2)) MEDIA5 format. We modified ACS routines to associate the tape output functions using encryption with a data class that requested encryption. We also used our data class to specify two new key labels and two new corresponding key encoding mechanisms. We upgraded 3592 Model E05 drive microcode to enable the drives to recognize and enable the EEFMT2 formatted cartridges to be reused.

Adding IOS support for the TS1120 tape drive

We added the new parameter, EKM, in the IECIOSxx parmlib member in support of the encryption-capable TS1120 tape drive. In order to use, In-band key management, we modified our IECIOSxx member to include the TCP/IP-related information needed to direct the IOS proxy to an appropriate key manager (primary and secondary).

For each encryption key manager, we specified the host name with the port as follows in the IECIOSxx parmlib member:

```
EKM PRIMARY=9.xx.xx.xxx:PORT
EKM SECONDARY=9.xx.xx.xxx:PORT
```

You can use the following EKM command to display the host names for the primary and secondary encryption key manager:

```
D IOS,EKM
```

We also used the SETIOS EKM command to dynamically change the settings for the encryption key manager, as follows:

```
SETIOS,EKM PRIMARY=9.xx.xx.xxx:PORT
SETIOS,EKM SECONDARY=9.xx.xx.xxx:PORT
```

The following sample JCL was used to encrypt the data on a 3992 Cartridge:

```
//ENCRYPT JOB
,,MSGLEVEL=(1,1),CLASS=A,MSGCLASS=R,NOTIFY=&SYSUID,
// REGION=0M
/*JOBPARM SYSAFF=*
//*
//STEP2 EXEC PGM=IEBGENER
//SYSUT1 DD DSN=PET.ENCRYPTION.TEST,DISP=SHR,
// UNIT=3390
//SYSUT2 DD UNIT=CRYPTAPE,DSN=PET.ENCY.TAPE,LABEL=(1,SL),
// DATACLAS=ENCRYPT,DISP=OLD,VOL=SER=XXXXXX
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//*
```

We used the following JCL to decrypt the data from Tape to DASD:

```
//DECRYPT JOB
,,MSGLEVEL=(1,1),CLASS=A,MSGCLASS=R,NOTIFY=&SYSUID,
// REGION=0M
/*JOBPARM SYSAFF=*
//*
//STEP2 EXEC PGM=IEBGENER
//SYSUT1 DD UNIT=CRYPTAPE,DSN=PET.ENCY.TAPE,LABEL=(1,SL),
// DATACLAS=ENCRYPT,DISP=OLD,VOL=SER=XXXXXX
```

```
//SYSUT2 DD
DSN=ENCRYPT.DUMP.DATA,DISP=(NEW,CATLG),VOL=SER=XXXXXX,
// UNIT=3390,SPACE=(TRK,(1000,0))
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//*
```

Running EKM and operational aspects

Once you have completed the install and customizations, the operational aspects are quite easy. Since it's defined as a started task, all that's needed is to issue the S EKM on each system that it was configured to run on. If you are using a ICSF hardware keystore, the ICSF task needs to be running before EKM is started or the server will not be able to connect to the keystore and will fail. Once the EKM is started, the JZOS code allows console interactions and also automatically issues the 'startekm' command.

Following is an example of the syslog messages when EKM is started:

```
IEF403I EKM - STARTED - TIME=14.49.27
BPXM023I (EKMSERV) 608
EKM console interaction is now available.
Starting the Encryption Key Manager 1.0-20061024
```

To submit commands to the EKM from the console:

```
F EKM,APPL='EKM command'
```

To stop the EKM properly:

```
P EKM
BPXM023I (EKMSERV) Loaded drive key store successfully
BPXM023I (EKMSERV) Starting the Encryption Key Manager 1.0-20061024
BPXM023I (EKMSERV) Processing Arguments
BPXM023I (EKMSERV) Processing
BPXM023I (EKMSERV) Server is started
BPXM023I (EKMSERV) Server is running. TCP port: 3801, SSL port: 1443
```

You will notice that the message indicates the code build level of EKM.

The following message indicates that the EKM server is connected to TCPIP and is monitoring the ports 3801 and 1443.

```
BPXM023I (EKMSERV) Server is running. TCP port: 3801, SSL port: 1443
```

Shutting down the EKM is quite easy. All you need to do is issue command P EKM and the server will stop cleanly. The following are the commands that can be issued from a MVS console to be directed to a EKM running on a specific system. If you forget the commands you can always issue the HELP command and the list will be provided.

```
F EKM,APPL='HELP'
```

The following list results from issuing the help command:

```
EKMAdmin usage:

adddrive -drivename <name> Y-rec1 <alias>¨ Y-rec2 <alias>¨

deldrive -drivename <name>
equivalent command is rmdrive or deletedrive or removedrive

exit or quit

export -drivetab|-config -url <url name>
```

```
help
equivalent command is ?

import -merge|-rewrite -drivetab|-config -url <url name>

listcerts Y-alias <alias>¨ Y-verbose|-v¨

listconfig

listdrives Y-drivename <drive_name> Y-verbose|-v¨¨

logout
equivalent command is logoff
Only useful when LoginModule is enabled

modconfig -set -property <name> -value <value> | -unset -property <name>
equivalent command is modifyconfig

moddrive -drivename <name> Y-rec1 alias¨ Y-rec2 alias¨
equivalent command is modifydrive

refresh

startekm

status

stopekm

version

sync -all|-config|-drivetab -ipaddr <ip address:ssl port> Y-merge|-rewrite¨
```

Notes:

1. Dashes and spaces must be contained in quotes for either to show up as part of an argument value.
2. Since we used Shared HFS function and shared directories for debug and error logs, we will **NOT** be using the **sync -all** or **sync -config** commands. This function is designed to 'SYNC' all the settings from one EKM to another. You do not want to use this function if you are using shared HFS libraries as this will change the log settings on synchronized systems to use the same directories. Only the **sync -drivetable** command should be used for this type of configuration.

Errors we encountered with the Enterprise Key Manager

The following are errors that we encountered during our testing. There are many other errors you may encounter besides the ones we list and you can find them in the *IBM Encryption Key Manager component for the Java platform "Introduction, Planning & User's Guide" (GA76-0418)* which can be viewed at the following link:

`http://www.ibm.com/support/docview.wss?rs=0&dc=D400&q1=ekm&uid=ssg1S4000504&loc`

1. If you plan on using a HARDWARE provider for your keystore, such as 4758, there is another customization step you need to perform or you will find you can encrypt a tape but then not be able to decrypt it. The error you will see in the audit log will be:

   ```
   ***Error: Information not available : Information not available for protected
   private keys ErrorCode=0xEE0F
   ```

   The problem is resolved by copying the un-restricted security files that are shipped in your JDK build as shown in the example:

   ```
   /javaekm/J1.4/lib/security:> cp javaekm/J1.4/demo/jce/policy-files/unrestricted/*
   ```

2. In a Business 2 Business scenario, when adding the certificate which contains only a public key, it is very important to use the option USAGE(CERTAUTH) on the RACF CONNECT command. If this is not specified, the EKM will not start as it believes there should also be a private key contained in the certificate. The error we received was:

   ```
   java.io.IOException: The private key of key_label is not a software or icsf key.
   Error creating key entry because private key is not available
   ```

3. If in the scenario above, where the shared keys are stored in the hardware, and the cards are not available on that image, EKM will not start. The error you will receive is the following:

   ```
    Oct 31, 2006 8:52:06 AM ThreadYmain,5,main¨ KeyStoreLoader
   LoadKeyStore ALL:
   java.security.PrivilegedActionException: java.io.IOException: Hardware error from call
   CSNDPKI returnCode 12 reasonCode 11060
   ```

4. If in the scenario above, where the shared keys are stored in the hardware, and ICSF is not started, you will receive the following error:

   ```
   Hardware error from call CSNDDSV returnCode 12reasonCode 0.
   ```

Encryption Facility V1.2, setting up and testing the support for the OpenPGP standard

z/OS Integration Test recently implemented and tested the new release of the Encryption Facility, V1.2, OpenPGP support. (In a prior test effort, we did the same for Encryption Facility V1.1. You can read about our experience testing the prior release in our December 2006 test report.) With the OpenPGP support added in V1.2, user's can encrypt and decrypt z/OS-type data for use with OpenPGP-compliant systems as well as OpenPGP-compliant messages and files.

Our testing consisted of installing the software needed to run this support, regression testing the System z format originally available in V1.1 along with testing the new OpenPGP format.

Note: We used the *Encryption Facility for z/OS Using Encryption Facility for OpenPGP Version 1 Release 2, document number SA23-2230* throughout the setup and testing of this product.

Our setup consisted of the following steps:

1. "Installing software updates"
2. "Setting up the Encryption Facility for OpenPGP" on page 60
3. "Downloading Java" on page 60
4. "Setting up JZOS" on page 60
5. "Copying Java unrestricted policy files" on page 61
6. "Creating the Encryption Facility userid" on page 61
7. "Setting up the RACF Keyring and certificates" on page 61
8. "Configuring Encryption Facility OpenPGP support" on page 61
9. "Running the OpenPGP Encrypt and Decrypt jobs" on page 63

Installing software updates

In order to test the new OpenPGP support available with Encryption Facility V1.2, we installed the following:

- Encryption Facility V1.2, FMID HCF7740

- ICSF APAR OA19177 – We already had ICSF HCR7731 installed so we didn't need to install the product, just the support for OpenPGP
- IBM 31-bit SDK for z/OS, Java 2 Technology Edition, Version 5 at Service Refresh level SDK5 SR4
- IBM JZOS Batch Toolkit for z/OS (JZOS) – This comes with the Java level above.

Setting up the Encryption Facility for OpenPGP

Once the new code was installed, we used the Encryption Facility V1.2 Program Directory to set up the directories that were needed for the OpenPGP support. We created the directory *encryptionfacility* off of */var* and */etc*. Encryption Facility for OpenPGP uses a configuration file called ibmef.config. There is a sample ibmef.config in */usr/lpp/encryptionfacility*. Per the program directory, we copied this sample into */etc/encryptionfacility*. Note that you do not need to specify command options in this file, as you can specify command options on the command line itself. If the *–homedir* option is not specified, the ibmef.config file is assumed to be in the */etc/encryptionfacility* directory.

Downloading Java

The next step was to download using FTP the JDK 1.4.2 sr6 from IBM website:

```
http://www.ibm.com/servers/eserver/zseries/software/java
```

The downloaded file was named UK21775.PAX.Z. From the */java/efjava* directory, we issued the following command to unpax the file.

```
pax -rf UK21775.PAX.Z
```

At first, this pax command failed with the following error message:

```
pax: FSUMF073 J5.0/bin/motif21/libmawt.so: user not authorized to restore
extended attribute "l".
```

We found that the user id issuing the pax command must have read authority to the following Facility Classes:

```
BPX.FILEATTR.PROGCTL
BPX.FILEATTR.SHARELIB
BPX.FILEATTR.APF
```

Once our user id had the correct authorities, we were able to install java successfully!

Setting up JZOS

Since we planned to launch our Encryption Facility OpenPGP work from batch, we used IBM JZOS Batch Toolkit for z/OS (JZOS) which comes with the java level we installed above. We used the java readme at:

```
http://www.ibm.com/servers/eserver/zseries/software/java/j5sdk31readme.html
```

We followed the directions under Optional Install of the JZOS Batch Launcher to install the latest JZOS code.

From the */java/efjava/J5.0/mvstools/samples/jcl* directory we copied:

```
JVMJCL50 into SYS1.SAMPLIB(JVMJCL50)
JVMPRC50 into SYS1.PROCLIB(JVMPRC50)
```

From the */java/efjava/J5.0/mvstoolspv* directory we copied:

```
JVMLDM50 into JZOS.LOADLIB
```

For JVMJCL50 we added appropriate job card information to the job. We also added:

```
export JAVA_HOME=/java/efjava/J5.0
```

For JVMPRC50, since we use JZOS.LOADLIB, without a high level qualifier, we removed the <HLQ> from the STEPLIB for JVMLDM module statement. We also uncommented the STEPLIB DD DSN=&LIBRARY,DISP=SHR statement.

To test to make sure JZOS was setup correctly, we submitted JVMJCL50 and it ran successfully, Hello World was output to STDOUT.

Copying Java unrestricted policy files

Another item that we had to do during the setup was to replace the US_export_policy.jar and local_policy.jar files in our */java/efjava/J5.0/lib/security* directory with an unrestricted version of these files. These unrestricted policy files are required by Encryption Facility V1.2 in order for Java to use large key sizes. The unrestricted policy files that are needed are shipped in the z/OS Java SDK build under the jce demo directory under the same names as above (*US_export_policy.jar* and *local_policy.jar*). We copied the files contained in the jce demo directory into our */java/efjava/J5.0/lib/security* directory (using the same names).

For additional information concerning these unrestricted files, see:

```
http://www.ibm.com/servers/eserver/zseries/software/java/j5jcecca.html#online
```

Creating the Encryption Facility userid

We created a userid called EFV12 which we used when running our encrypt/decrypt batch jobs. This userid was also used when creating certificates through RACF.

Setting up the RACF Keyring and certificates

Since we decided to use the RACF keyring with keys stored in ICSF, we needed to create the keyring and the appropriate certificates. Following are the steps we took:

```
Create KeyRing: RACDCERT ADDRING(efv12key) ID(EFV12)
Create CA self signed certificate:  RACDCERT CERTAUTH GENCERT
   SUBJECTSDN(CN('EFV12 CertAuth') C('US')) WITHLABEL('EFV12
   CertAuth') ICSF TRUST
Generate EFV12 certificate: RACDCERT ID(EFV12) GENCERT
   SUBJECTSDN(CN('EFV12 Cert') O('IBM')) WITHLABEL('EFV12Cert') ICSF
   SIGNWITH(CERTAUTH LABEL('EFV12 CertAuth')) TRUST
Connect EFV12 certificate to keyring: RACDCERT ID(EFV12) CONNECT
   (LABEL('EFV12Cert') RING(efv12key) DEFAULT USAGE(PERSONAL))
Connect CA certificate to keyring: RACDCERT ID(EFV12) CONNECT (LABEL('EFV12
    CertAuth') RING(efv12key) CERTAUTH)
```

Note: Specifying ICSF when creating the CA and the EFV12 certificate stores the private key in the ICSF PKDS.

Configuring Encryption Facility OpenPGP support

Encryption Facility V1.2 OpenPGP ships the following three members in SYS1.SAMPLIB:

1. CSDJZSVM – Procedure for executing the JZOS Java Batch Launcher
2. CSDSMPEN – Shell script used to configure Java JVM environment variables
3. CSDSMJCL – Batch job to launch the encrypt / decrypt work using the Encryption Facility V1.2 OpenPGP support.

CSDJZSVM is the procedure that is used for executing the Java Batch Launcher as it relates to the OpenPGP support. We placed this member in SYS1.PROCLIB and updated it to our environment. Our JZOS library is called JZOS.LOADLIB, so our LIBRARY= statement in this proc looks as follows:

```
LIBRARY='JZOS.LOADLIB'
```

CSDSMPEN is the shell script used to configure the Java JVM environment variables to be used for our OpenPGP testing. We placed this member in our EFV12.OPENPGP.JCL dataset. We made the following change in order to point to the level of Java we installed above:

```
  export JAVA_HOME=<JAVA_HOME> to export JAVA_HOME=/java/efjava/J5.0
```

CSDSMJCL is the sample job you can use to issue the encrypt/decrypt work using the Encryption Facility OpenPGP code. In order to use this sample, we had to do a few things. First we added jobcard information. We then updated the PROCLIB statement to point to SYS1.PROCLIB. For each job, JAVA1, JAVA2 and JAVA3, you need to ensue that the PROC= statement points to the Java Batch Launcher procedure. In our case, this is in SYS1.PROCLIB and is the same name as what is shipped in the sample, CSDJZSVM, so we did not have to make a change here. We did however, have to make a change to the STDENV dataset in each of the jobs so it pointed to the member containing the Java environment variables. For example, our JAVA1 job statements look as follows:

```
//JAVA1 EXEC PROC=CSDJZSVM,VERSION='50'
//STDENV DD DSN=EFV12.OPENPGP.JCL(CSDSMPEN),DISP=SHR
```

Further down in the sample, under the MAINARGS DD statement is where you would put your command options if you choose not to specify them in the ibmef.config file. We chose to use a combination of both.

Our view on this is that the values we place in the ibmef.config file are those which we would not change very often. So we placed the following values in the ibmef.config file:

```
KEY_RING_FILENAME /var/encryptionfacility/ibmpkring.ikr – This value
   defines the name of the key ring which stores the OpenPGP certificates.
USE_ASYNC_IO true – This value, along with the following two were
   set to take advantage of the asynchronous processing that is available
   when encrypting larger files
USE_ASYNC_COMPRESS true
USE_ASYNC_CIPHER true
JAVA_KEY_STORE_TYPE JCECCARACFKS – We chose this keystore type as
   we are using a RACF keyring with our keys stored in ICSF.
JAVA_KEY_STORE_NAME efv12key – This is the name of the RACF keyring
KEYSTORE_PASSWORD efv12 – This value is the password used to access
   the keystore. Initially we did not have this value set when we attempted
   our initial encrypt. As a result, we received the following messages:
  CSD002A Enter keystore password for efv12key:
  CSD0050I Command processing ended abnormally:
  .CSD0043I Input not recognized.
   Since RACF keyrings do not have passwords, this value can be set to anything.
KEY_PASSWORD efv12 – This value is the password given when a new key
   is generated in the key store. When we attempted a decrypt without this value specified,
   we received the following messages:
  CSD001A Enter keystore password for efv12servercert:
  CSD0050I Command processing ended abnormally:
  .CSD0043I Input not recognized.
   Looking in the Encryption Facility for z/OS: Using Encryption Facility for
   OpenPGP guide, Table 1. Key store repositories, when using JCECCARACFKS, the keystore
   password can be anything. The key password MUST match the keystore password.
JCE_PROVIDER_LIST com.ibm.crypto.hdwrCCA.provider.IBMJCECCA – This value prefixes
   the JCE provider list found in the java.security file. Since we are using a RACF keyring
   with keys stored in ICSF, we needed to specify this. When this value was not specified,
```

```
   the encrypt failed and I received message CSD0050I
   Command processing ended abnormally: . JCECCARACFKS not found.
RACF_KEYRING_USERID EFV12 – This value is the userid that the keyring is owned by.
```

Some of the other errors we hit along the way:

- When we created the EFV12 user id, we did not give it the proper authority to access the certificates through RACF. As a result, when we attempted an encrypt, we received the following messages:

  ```
  CSD0050I Command processing ended abnormally:
  .R_datalib (IRRSDL00) error: not RACF authorized to use the requested service (8, 8, 8).
  ICH408I USER(EFV12)  GROUP(SYS1) NAME(TESTER)
    IRR.DIGTCERT.LISTRING CL(FACILITY)
    INSUFFICIENT ACCESS AUTHORITY
    ACCESS INTENT(READ)   ACCESS ALLOWED (NONE)
  ```

- We issued the following RACF commands, from a userid with the proper authority, to give userid EFV12 proper authority to RACF commands:

  ```
  PERMIT IRR.DIGCERT.LISTRING CLASS(FACILITY) ID(EFV12) ACCESS(READ)
  PERMIT IRR.DIGCERT.LIST CLASS(FACILITY) ID(EFV12) ACCESS(READ)
  SETR RACLIST(FACILITY) REFRESH
  ```

- When we attempted an encrypt after fixing the problem above, we encountered the following error message:

  ```
  CSD0050I Command processing ended abnormally:
  ./var/encryptionfacility/ibmpkring.ikr (EDC5111I Permission denied.)
  ```

  From an ID with proper authority, we issued a chmod 775 to the encryptionfacility directory off of */var*.

- Initially when we typed in the KEY_PASSWORD value we typed it in incorrectly which resulted in it not being the same as the KEYSTORE_PASSWORD. When we issued the decrypt job, it failed with the following error:

  ```
  CSD0050I Command processing ended abnormally:
  .Given final block not properly padded
  ```

  Realizing our error, we changed it to the correct value and the decrypt ran fine.

Running the OpenPGP Encrypt and Decrypt jobs

Following is an example of the jcl used to encrypt a dataset:

```
//EFV12ENC JOB 'JOB INFORMATION',MSGLEVEL=(1,1),
//        MSGCLASS=H,CLASS=A,NOTIFY=&SYSUID,
//        USER=EFV12
//PROCLIB JCLLIB ORDER=SYS1.PROCLIB
//*
//*********************************************************************
//* -homedir is the location of the ibmef.config file
//* -o value is the output file
//* -rA value is the certificate to be used for the encryption
//* -e value is the dataset to be encrypted
//*********************************************************************
//*
//JAVA1 EXEC PROC=CSDJZSVM,VERSION='50'
//STDENV DD DSN=EFV12.OPENPGP.JCL(CSDSMPEN),DISP=SHR
//*
//DDDEF DD DSN=&SYSUID..EFPGP.ENC.OUT,
//        DISP=(NEW,KEEP),
//        DCB=(RECFM=VB,LRECL=32756,BLKSIZE=32760),
//        UNIT=SYSALLDA,
//        SPACE=(CYL,(300,1))
//*
//MAINARGS  DD *
-homedir /etc/encryptionfacility/
```

```
-o '//EFV12.EFPGP.ENC.OUT'
-rA 'EFV12Cert'
-e '//SYSTEM.SYSMDUMP'
/*
```

Following is an example of the jcl used to decrypt the data that was encrypted above:

```
//EFV12DEC JOB 'JOB INFORMATION',MSGLEVEL=(1,1),
//        MSGCLASS=H,CLASS=A,NOTIFY=&SYSUID,
//        USER=EFV12
//PROCLIB JCLLIB ORDER=SYS1.PROCLIB
//*
//*********************************************************************
//* -homedir is the location of the ibmef.config file
//* -o value is the output file
//* -rA value is the certificate to be used for the encryption
//* -d value is the dataset to be decrypted
//*********************************************************************
//*
//JAVA1 EXEC PROC=CSDJZSVM,VERSION='50'
//STDENV DD DSN=EFV12.OPENPGP.JCL(CSDSMPEN),DISP=SHR
//*
//DDDEF DD DSN=&SYSUID..EFPGP.DEC.OUT,
//        DISP=(NEW,KEEP),
//        DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160),
//        UNIT=SYSALLDA,
//        SPACE=(CYL,(157,1))
//*
//MAINARGS  DD *
-homedir /etc/encryptionfacility/
-o '//EFV12.EFPGP.DEC.OUT'
-rA 'EFV12Cert'
-d '//EFV12.EFPGP.ENC.OUT'
/*
```

Chapter 4. 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 V1R8"
- "Migrating to z/OS.e V1R8" on page 67

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.

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 our...
z/OS.e V1R7	*zSeries Platform Test Report for z/OS and Linux Virtual Servers December 2005 edition*
z/OS V1R7	*zSeries Platform Test Report for z/OS and Linux Virtual Servers December 2005 edition*
z/OS.e V1R6	*zSeries Platform Test Report for z/OS and Linux Virtual Servers September 2004 edition*
z/OS V1R6	*zSeries Platform Test Report for z/OS and Linux Virtual Servers September 2004 edition*

Migrating to z/OS V1R8

This section describes our migration experiences with z/OS V1R8.

z/OS V1R8 base migration experiences

In this section we described our experiences with our base migration to z/OS V1R8, 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 V1R8

The following is an overview of our z/OS V1R8 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 66 shows the high-level process we followed to migrate the members of our sysplex from z/OS V1R7 to z/OS V1R8.

Table 9. Our high-level migration process for z/OS V1R8

Stage	Description
Updating parmlib for z/OS V1R8	We created SYS1.PETR18.PARMLIB to contain all the parmlib members that changed for z/OS V1R8 and we used our LOAD*xx* member for migrating our systems one at the time. (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 V1R8 image	We brought up z/OS V1R8 on our Z1 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 V1R8 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 V1R8 images	We continued to bring up additional z/OS V1R8 images across our sysplex, as follows: • We brought up z/OS V1R8 on our Z2 test system and ran with it for a week. • Next we migrated our last test system, Z3, and ran for a week. • Next, we migrated some of our production systems, JA0, JE0 and TPN, and we ran with it for a couple of days. • At this point, we took two of our production V1R8 images; JA0 and JE0 back down to V1R7. 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 four additional production systems, Z0, JH0, J90 and J80, and ran for a week. • Next we migrated the rest of our production systems, JF0, JB0 and JC0 to V1R8.

More about our migration activities for z/OS V1R8

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 V1R7 and z/OS V1R8
- z/OS V1R7 and z/OS.e V1R8
- z/OS V1R7 JES2 and z/OS V1R8 JES2
- z/OS V1R7 JES3 and z/OS V1R8 JES3.

Using concatenated parmlib: We continue to use concatenated parmlib support to add or update parmlib members for z/OS V1R8. Appendix A, "Some of our parmlib members," on page 245 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 V1R8 in one place and makes it easier to migrate multiple systems. Rather than change many parmlib members each time we migrate another system to V1R8, we just add the PARMLIB statements at the appropriate places in SYS0.IPLPARM(LOAD*xx*) to allow that system to use SYS1.PETR18.PARMLIB.

Recompiling REXX EXECs for automation: We recompiled our SA OS/390 REXX EXECs when we migrated to z/OS V1R8. We discuss the need to recompile these REXX EXECs in our our December 1997 edition.

Migrating to z/OS.e V1R8

This section describes our migration experiences with z/OS.e V1R8.

z/OS.e V1R8 base migration experiences

This section describes our experiences with migrating one system image (JH0) from z/OS.e V1R7 to z/OS.e V1R8. Here we only cover our experiences with our base migration to z/OS.e V1R8, including our high-level migration process and other migration activities and considerations.

Our high-level migration process for z/OS.e V1R8

The following is an overview of our z/OS.e V1R8 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 V1R8 and z/OS.e V1R8.

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 z9 BC 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 z9 BC Software Pricing Configuration Technical Paper at www.ibm.com/servers/eserver/zseries/library/techpapers/pdf/gm130121.pdf.

Table 10 on page 68 shows the high-level process we followed to migrate our z/OS.e V1R7 system to z/OS.e V1R8.

Table 10. Our high-level migration process for z/OS.e V1R8

Stage	Description
Obtaining licenses for z/OS.e	You need a license for the appropriate number of engines on the z9 BC 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 z890 or z9 BC, if you intend to install it there). We use an internal process to do this; however, you must use the official process stated in *z9 BC Software Pricing Configuration Technical Paper*.
Updating the z890 or z9 BC 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 V1R8	z/OS.e requires the `LICENSE=Z/OSE` statement in the IEASYS*xx* parmlib member. We used the same SYS1.PETR18.PARMLIB data set that we created for z/OS V1R8. We then have separate IEASYS*xx* and IFAPRD*xx* members in SYS1.PARMLIB that we tailored specifically for z/OS.e.
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 V1R8.
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.PETR18.PARMLIB, we carried the change along for V1R8.
IPLing the z/OS.e V1R8 image	We brought up z/OS.e V1R8 on our JH0 production system.

More about our migration activities for z/OS.e V1R8

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

About our z9 BC LPAR environment: z/OS.e must run in LPAR mode on a System z9 BC 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 z9 BC 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 z9 BC server.

Note: Don't let the fact that z/OS.e only runs on a z890 or z9 BC server confuse you. These are fully functional zSeries servers and, in addition to z/OS.e, they support all of the same zSeries operating systems as a z9 EC 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 V1R8. We use the same SYS1.PETR18.PARMLIB data set as we do for our z/OS V1R8 systems.

Below are examples of our parmlib customizations to accommodate z/OS.e V1R8. Appendix A, “Some of our parmlib members,” on page 245 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   z9 BCname
LPARNAME  ZOSEJH0
PARMLIB  SYS1.PETR18.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.PETR18.PARMLIB to point to our new IFAPRD02 parmlib member and to reflect the z/OS.e LPAR name, as follows:

```
⋮
SYSDEF HWNAME(z9 BCname)
       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 V1R8. 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 V1R8

Our testing of z/OS.e V1R8 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

Chapter 5. Using the z9 Integrated Information Processor (zIIP)

IBM extended its mainframes data serving capabilities, delivering a new roadmap for the future of data serving and information on demand, previewing new DB2 function, and introducing a new specialty engine directed toward data serving workloads.

The new specialty engine, the IBM System z9 Integrated Information Processor (IBM zIIP), is now available on the System z9 Enterprise Class (EC) and System z9 Business Class (BC) servers.

A zIIP is similar in concept to the zSeries Application Assist Processor (zAAP). Like zAAPs; but unlike CPs, ICFs and IFLs, zIIPs can do nothing on their own; they can not perform an IPL and can not run an operating system. zIIPs must operate along with general purpose CPs within logical partitions running z/OS or z/OS.e, however they are designed to operate asynchronously with the general purpose CPs to execute selective workloads such as:

- ERP or CRM application serving - For applications, running on z/OS, UNIX, Intel, or Linux on System z that access DB2 for z/OS V8 on a System z9, through DRDA over a TCP/IP connection, DB2 gives z/OS the necessary information to have portions of these SQL requests directed to the zIIP.
- Data Warehousing applications – Requests that utilize DB2 for z/OS V8 for long running parallel queries, including complex star schema parallel queries, may have portions of these SQL requests directed to the zIIP when DB2 gives z/OS the necessary information. These queries are typical in data warehousing implementations. The addition of select long running parallel queries may provide more opportunity for DB2 customers to optimize their environment for Data Warehousing while leveraging the unique qualities of service provided by System z9 and DB2.
- Some DB2 for z/OS V8 utilities – A portion of DB2 utility functions used to maintain index maintenance structures (LOAD, REORG, and REBUILD INDEX) that typically run during batch, can be redirected to zIIPs.

This chapter describes what we did to configure and to prepare to exercise and test the zIIP feature on our z9 systems.

Prerequisites for zIIP

The following are prerequisites for zIIP usage:

- z/OS V1R6 with JBB77S9 applied
- z/OS V1R7 with JBB772S applied
- z/OS V1R8
- DB2 V8 with the appropriate maintenance.

More detailed information about all the software and hardware prerequisites can be found in the following PSP buckets:

- Hardware 2094 and 2096 devices buckets.
- z/OS BCP zIIP bucket
- zIIP functional PSP Bucket

Also please contact your local hardware and software representatives for any additional requirements.

Configuring the zIIPs

We configured two zIIPs on all our z/OS images on our System z9 EC and we configured one zIIP on our System z9 BC. When you configure your z/OS logical partitions you simply specify how many logical zIIPs you want to define for each partition, just as you do for the number of standard CPs and zAAPs. When you IPL the system, z/OS determines how many zIIPs are configured and manages an additional dispatcher queue for zIIP-eligible work.

We did the following to configure our zIIPs:

1. Updated the image profile for all our System z9 EC partitions to define two zIIPs to each partition
2. Updated our System z9 BC partition to define one zIIP. Figure 8 shows an example of the image profile for our J80 z/OS image with 2 zIIPs defined:

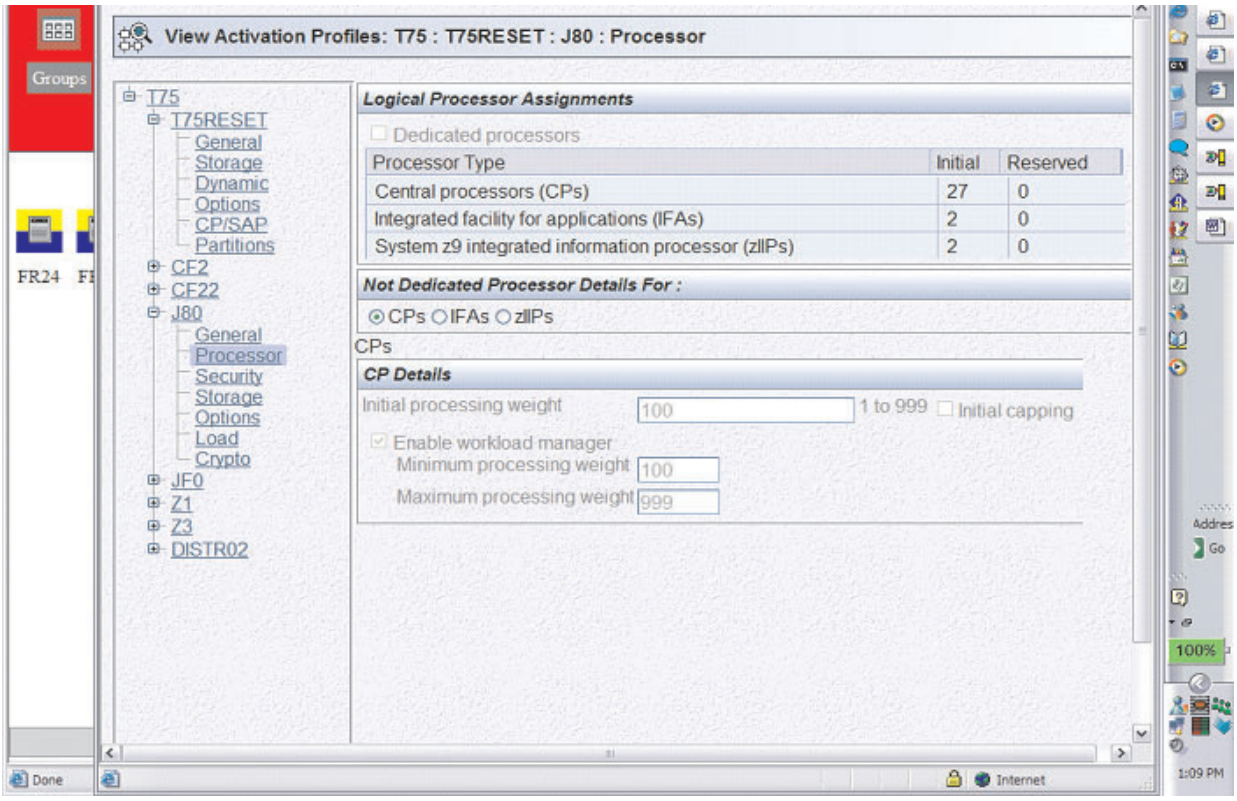


Figure 8. Image profile for our J80 z/OS image with 2 zIIPs defined

3. Deactivated, activated and IPL'd the z/OS partitions to bring the zIIPs online.

   You can use the D M=CPU command to display the status of the zIIPs. The zIIPs appear as an integrated information processor in response to the D M=CPU command.

   Response example for the D M=CPU command on system JH0:

```
-JH0D M=CPU
 IEE174I 13.14.39 DISPLAY M 372
 PROCESSOR STATUS
```

```
 ID  CPU                 SERIAL
 00  +                    01FE2D2096
 01  +                    01FE2D2096
 02  +                    01FE2D2096
 03  +                    01FE2D2096
 04  +A                   01FE2D2096
 05  +I                   01FE2D2096

 CPC ND = 002096.S07.IBM.02.00000002FE2D
 CPC SI = 2096.Z04.IBM.02.000000000002FE2D
 CPC ID = 00
 CPC NAME = K25
 LP NAME = ZOSEJH0    LP ID =  1
 CSS ID  = 0
 MIF ID  = 2
+ ONLINE    - OFFLINE    . DOES NOT EXIST    W WLM-MANAGED
N NOT AVAILABLE

A        APPLICATION ASSIST PROCESSOR (zAAP)
I        INTEGRATED INFORMATION PROCESSOR (zIIP)
CPC ND  CENTRAL PROCESSING COMPLEX NODE DESCRIPTOR
CPC SI  SYSTEM INFORMATION FROM STSI INSTRUCTION
CPC ID  CENTRAL PROCESSING COMPLEX IDENTIFIER
CPC NAME CENTRAL PROCESSING COMPLEX NAME
LP NAME  LOGICAL PARTITION NAME
LP ID    LOGICAL PARTITION IDENTIFIER
CSS ID   CHANNEL SUBSYSTEM IDENTIFIER
MIF ID   MULTIPLE IMAGE FACILITY IMAGE IDENTIFIER
```

Response example for the D M=CPU command on system J80:

```
-D M=CPU
 IEE174I 07.47.11 DISPLAY M 895
 PROCESSOR STATUS
 ID  CPU                 SERIAL
 00  +                    07299E2094
 01  +                    07299E2094
 02  +                    07299E2094
 03  +                    07299E2094
 04  +                    07299E2094
 05  +                    07299E2094
 06  +                    07299E2094
 07  +                    07299E2094
 08  +                    07299E2094
 09  +                    07299E2094
 0A  +                    07299E2094
 0B  +                    07299E2094
 0C  +                    07299E2094
 0D  +                    07299E2094
 0E  +                    07299E2094
 0F  +                    07299E2094
 10  +                    07299E2094
 11  +                    07299E2094
 12  +                    07299E2094
 13  +                    07299E2094
 14  +                    07299E2094
 15  +                    07299E2094
 16  +                    07299E2094
 17  +                    07299E2094
 18  +                    07299E2094
 19  +                    07299E2094
 1A  +                    07299E2094
 1B  +A                   07299E2094
 1C  +A                   07299E2094
 1D  +I                   07299E2094
 1E  +I                   07299E2094

 CPC ND = 002094.S38.IBM.02.0000000C299E
```

```
 CPC SI = 2094.729.IBM.02.00000000000C299E
CPC ID = 00
 CPC NAME = T75
 LP NAME = J80        LP ID =  7
 CSS ID  = 0
 MIF ID  = 7

 + ONLINE    - OFFLINE    . DOES NOT EXIST    W WLM-MANAGED
 N NOT AVAILABLE

 A        APPLICATION ASSIST PROCESSOR (zAAP)
 I        INTEGRATED INFORMATION PROCESSOR (zIIP)
 CPC ND  CENTRAL PROCESSING COMPLEX NODE DESCRIPTOR
 CPC SI  SYSTEM INFORMATION FROM STSI INSTRUCTION
 CPC ID  CENTRAL PROCESSING COMPLEX IDENTIFIER
 CPC NAME CENTRAL PROCESSING COMPLEX NAME
 LP NAME  LOGICAL PARTITION NAME
 LP ID    LOGICAL PARTITION IDENTIFIER
 CSS ID   CHANNEL SUBSYSTEM IDENTIFIER
 MIF ID   MULTIPLE IMAGE FACILITY IMAGE IDENTIFIER
```

Monitoring zIIP utilization:

There is support in RMF to provide information about zIIP utilization. This information is useful to determine if and when you need to add zIIP capacity. For more details about RMF support for zIIPs and new fields on this report, please see *z/OS RMF Report Analysis*, SC33-7991.

Here is an example of our RMF Monitor III, CPC Report that displays the use of zIIP processors (in **bold**) on our System z9 EC images:

```
            HARDCOPY     RMF V1R7   CPC Capacity                  Line 1 of 30
 Command ===>
0Samples: 119     System: J80   Date: 05/24/06  Time: 10.22.00  Range: 120   Sec
0Partition:   J80        2094 Model 729
 CPC Capacity:    1524   Weight % of Max: 10.0       4h MSU Average:   114
 Image Capacity:  1419   WLM Capping %:   ****       4h MSU Maximum:   322
0Partition  --- MSU ---  Cap  Proc    Logical Util %   - Physical Util % -
              Def   Act  Def   Num    Effect   Total   LPAR  Effect  Total
0*CP                                                    5.0    77.9   82.9
 DISTR01        0     1  NO    2.0       0.5     0.6    0.0     0.0    0.0
 DISTR02        0     0  NO    5.0       0.0     0.0    0.0     0.0    0.0
 JF0            0   366  NO   14.0      48.9    49.7    0.4    23.6   24.0
 J80            0   750  NO   23.0      60.6    62.1    1.2    48.1   49.2
 Z1             0    14  NO    8.0       3.2     3.4    0.0     0.9    0.9
 Z3             0    82  NO    8.0      19.2    19.5    0.1     5.3    5.4
 PHYSICAL                                               3.3            3.3

*AAP                                                    2.3    97.6   99.9
JF0                     NO    2.0      32.8    33.0    0.2    32.8   33.0
J80                     NO    2.0      32.7    33.0    0.3    32.7   33.0
Z1                      NO    2.0       0.1     0.3    0.1     0.1    0.3
Z3                      NO    2.0      32.0    32.1    0.1    32.0   32.1
PHYSICAL                                               1.6            1.6

*IFL                                                    0.2     0.0    0.2
PETLVS                  NO    1.0       0.0     0.0    0.0     0.0    0.0
PETLVS2                 NO    1.0       0.0     0.0    0.0     0.0    0.0
PHYSICAL                                               0.2            0.2

*ICF                                                    0.0    99.5   99.5
CF2                           3.0      99.6    99.6    0.0    74.7   74.7
CF22                          1.0      99.4    99.4    0.0    24.9   24.9
PHYSICAL                                               0.0            0.0

*IIP                                                   4.0    59.8   63.7
```

```
JF0                    NO    2.0      17.2     17.6     0.4      17.2     17.6
J80                    NO    2.0      42.6     43.3     0.8      42.6     43.3
PHYSICAL                                                2.8               2.8
```

SMF type 70.1, 72.3, 79.1 and 79.2 records contain new fields with zIIP measurements. There are also new fields in SMF type 30 records to indicate the amount of time spent in zIIP work as well the amount of time spent executing zIIP eligible work on standard processors. *z/OS MVS System Management Facilities (SMF)*, SA22-7630 can give you details on the new fields.

SDSF also provides information about system zIIP utilization as well as enclave zIIP utilization. New columns on the DA display and the Enclave display have been added to provide this information. For more details about these new fields for SDSF please see *z/OS SDSF Operation and Customization*, SA22-7670.

Here is one example for the SDSF enclave display that shows zIIP utilization on our z9 EC systems:

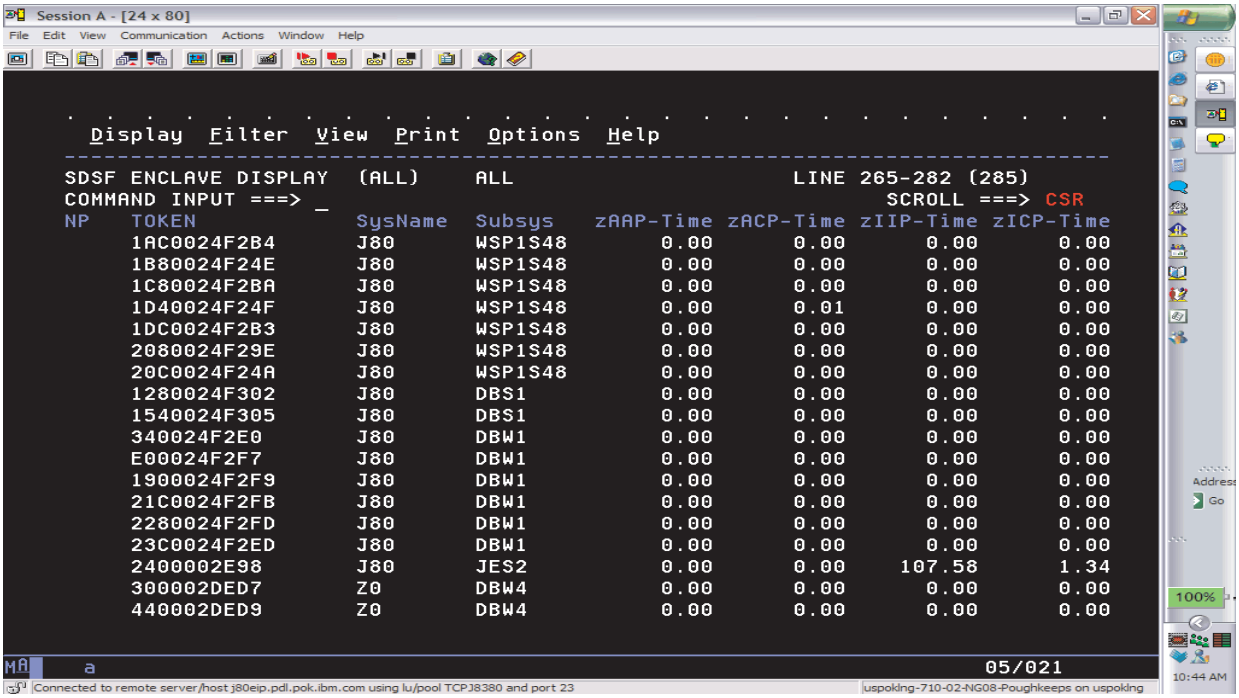


Figure 9. SDSF display showing zIIP utilization

Workloads that exercise the zIIP processors

The System z9 Integrated Information Processor (zIIP) is designed so that specific types of DB2 programs or utilities can negotiate with z/OS to have a portion of their enclave Service Request Block (SRB) work redirected from the general purpose Central Processor (CP) over to the zIIP, thereby freeing the CP for other tasks.

Those types of work which do not utilize SRBs, such as stored procedures and user-defined functions, are not eligible to offload work to the zIIP.

Currently, there are basically three situations or scenarios that may benefit from having a portion of their SQL requests redirected to the zIIP; they include the following:

1. Applications running on z/OS, UNIX, Intel, or Linux on System z that access DB2 via DRDA over a TCP/IP connection.

   To test offloading portions of SQL requests using DRDA access over a TCP/IP connection to zIIP, we employed the use of the IBM Trade Performance Benchmark Sample for WebSphere Application Server V6.0 (or simply the Trade 6 workload), which may be obtained from the following website:

   `https://www.software.ibm.com/webapp/iwm/web/preLogin.do?source=trade6`

   Logon (or register if you are a new user), download tradeInstall.zip (1.7MB), and refer to the Trade Technology document (tradeTech.pdf) located in the install package for general information regarding Trade 6.

   During our testing, we were able to drive substantial zIIP utilization using the Trade 6 workload and were able to monitor it via RMF Monitor III.

2. Requests that utilize DB2 for long running complex parallel queries, such as star schema parallel queries.

   For this particular scenario, we made use of the following star join query which was executed after having enabled star schema parallelism:

```
SELECT COUNT(*) FROM
              ADMF001.TBFACT1 F,
              ADMF001.TBDIMN01 D1,
              ADMF001.TBDIMN02 D2,
              ADMF001.TBDIMN03 D3,
              ADMF001.TBDIMN04 D4,
              ADMF001.TBDIMN05 D5
  WHERE
    F.TIME_CLOSED_KEY  = D1.TIME_CLOSED_KEY   AND
    F.TOD_KEY          = D2.TOD_KEY           AND
    F.RECEIVED_VIA_KEY = D3.RECEIVED_VIA_KEY  AND
    F.CASE_KEY         = D4.CASE_KEY          AND
    F.CUSTOMER_KEY     = D5.CUSTOMER_KEY      AND
F.TIME_CLOSED_KEY  = 182;
```

   We noted some activity being redirected to the zIIP, but not a great deal. Note that even though star schema parallelism has been enabled and a zIIP is available for use, the DB2 Optimizer can decide that the optimal path is not to use star join, thus bypassing the zIIP. The optimizer's focus is not whether the query can take advantage of zIIP offload or not, but rather choosing the lowest cost access path.

3. Some DB2 utilities used in the maintenance of index structures that are normally executed in batch, such as the LOAD, REORG, and REBUILD INDEX utilities.

   Testing the offloading of portions of the DB2 LOAD, REORG, and REBUILD INDEX utilities to zIIP entailed the creation of a batch workload comprised of three jobs, each of which performs a task specific to zIIP testing:

   LOAD Reloads tables

   RBLDINDX
   Rebuilds indexes

   REORG
   Reorgs tables

   The jobs are currently chained together with LOAD executing first; LOAD then calls RBLDINDX, which in turn calls REORG. If desired, for continuous operation REORG can be set to call LOAD again. The three jobs together take about a half hour to complete. Of the three scenarios mentioned, this particular one redirected more work to the zIIP than the star schema parallel queries but less than the Trade 6 workload utilizing DRDA over TCP/IP connections.

OMEGAMON XE for z/OS 3.1.0 zIIP SUPPORT

We recently installed OMEGAMON XE for z/OS 3.1.0 into PET. To learn more about OMEGAMON XE for z/OS 3.1.0 go to:

```
http://www-306.ibm.com/software/tivoli/products/omegamon-xe-zos/
```

If you already have OMEGAMON XE for z/OS 3.1.0 installed you will need the following support to enable the zIIP support:

```
OB550:  UA27609 (APAR OA15898)
M2550:  UA27610 (APAR OA15899)
M5310:  UA27611 (APAR OA15900)


OP360 TEP: 3.1.0-TIV-KM5-IF0001
ITM6.1 TEP 3.1.0-TIV-KM5-ITM-IF0001
```

We were the first Plex with zIIPs to actually verify and use the OMEGAMON XE for z/OS 3.1.0 zIIP support. To access the OMEGAMON Classic support for zIIP, select 'C CPU' from the OMEGAMON MAIN MENU. See Figure 10. zIIP is represented by **IIP**.

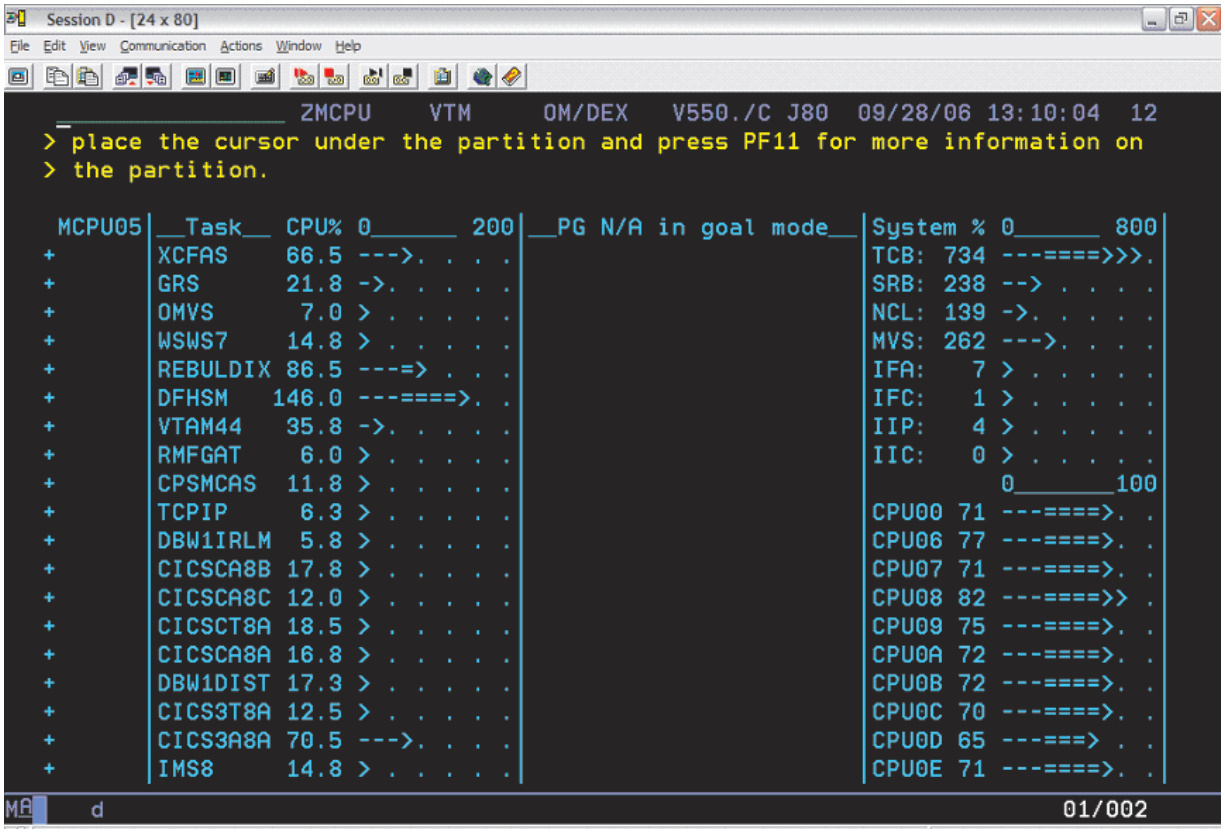


Figure 10. OMEGAMON ZMCPU screen

From your TEP server, the OMEGAMON XE for z/OS zIIP can be found in the predefined workspace 'System CPU Utilization'. Figure 11 on page 78 and Figure 12 on page 79 show the TEP OMEGAMON XE for z/OS 'System CPU Utilization' workspace:

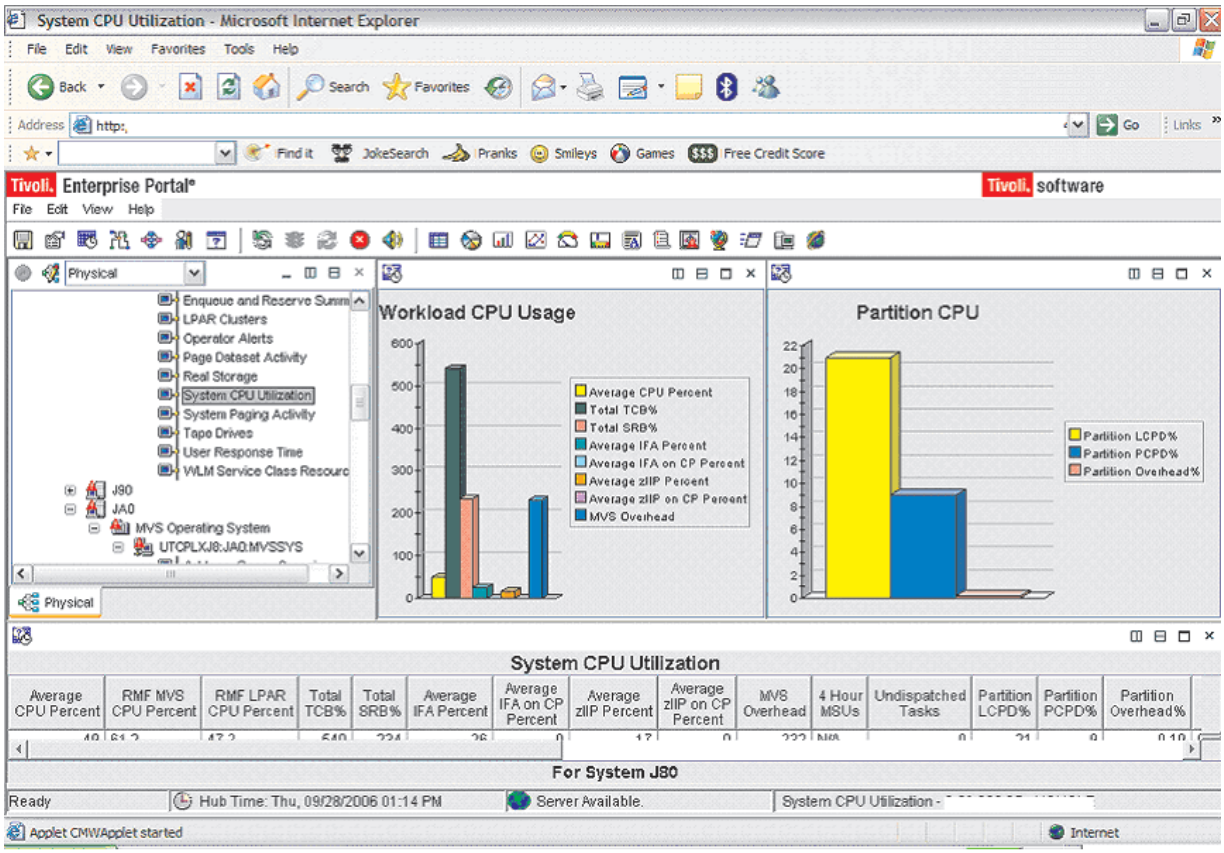


Figure 11. OMEGAMON System CPU Utilization 1

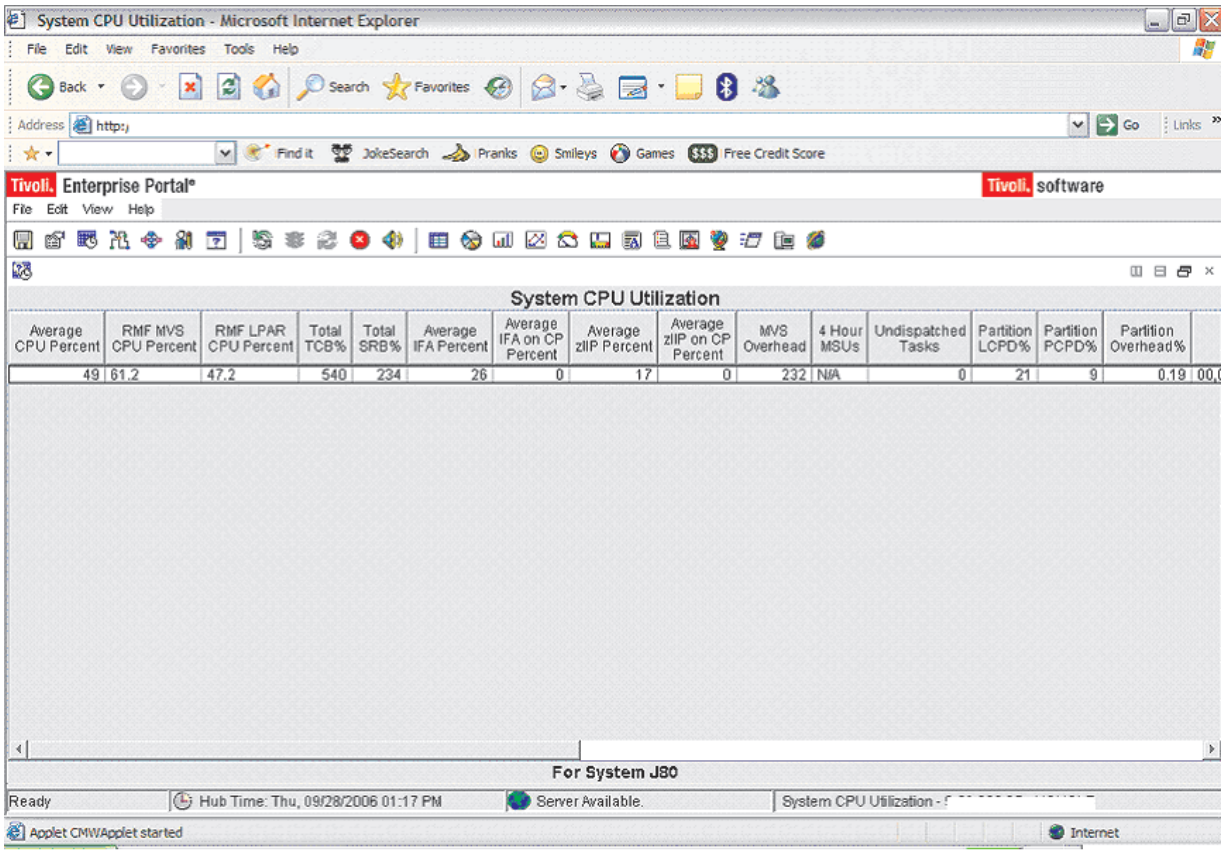


Figure 12. OMEGAMON System CPU Utilization 2

From your TEP server, the OMEGAMON XE for z/OS zIIP support can also be found in the predefined workspace 'Address Space Overview' as seen in Figure 13 on page 80.

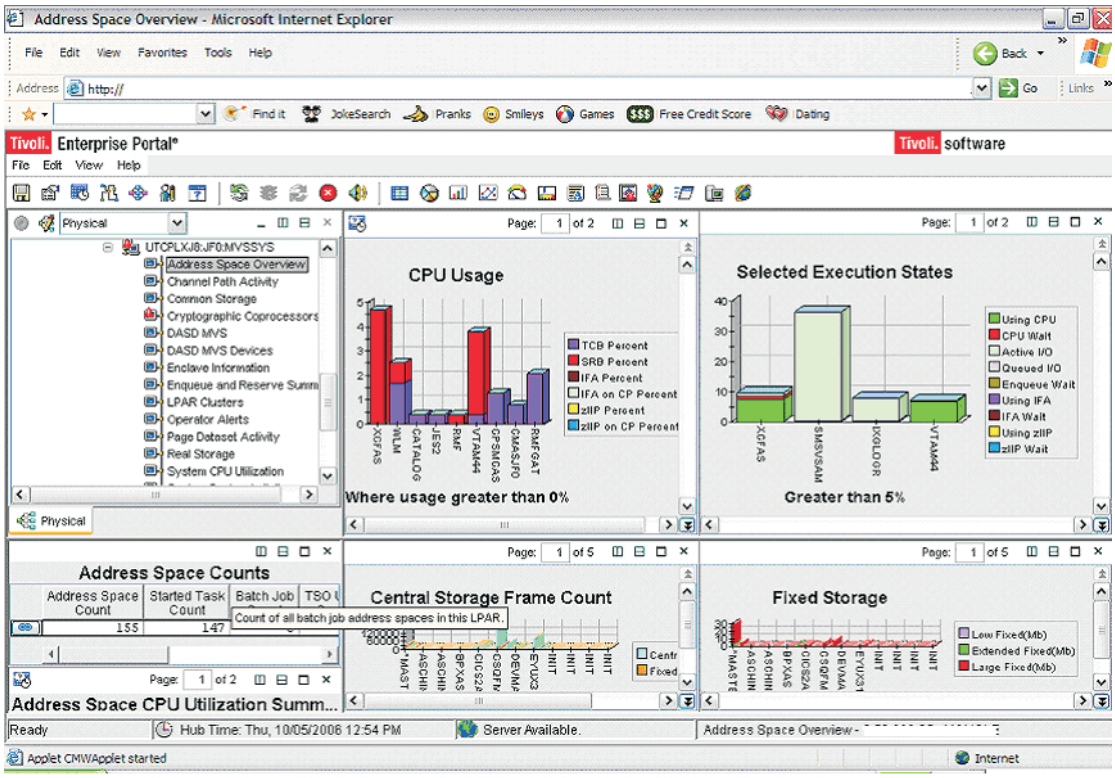


Figure 13. OMEGAMON Address Space Overview

Chapter 6. Migrating to DB2 Version 9.1

This chapter addresses the processes and experiences encountered during the migration of the Integration Test production 3 way DB2® data sharing group DBSG from DB2 Version 8 to Version 9.1 (composed of members DBS1, DBS2, DBS3).

We used the *DB2 Installation Guide, (GC18-9846-00)* 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*.

Migrating DB2 on z/OS requires common known administration skills on zSeries(z/OS) platform. This chapter is organized in the following sections:

Migration considerations

Before you migrate to DB2 Version 9, note the following points:

- Migrations to DB2 Version 9 are only supported from subsystems currently running DB2 Version 8; 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 9 subsystem after it is in compatibility mode. For this step, only the following selected Version 8 IVP jobs can be executed:
  1. Version 8 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 8 phase 3 IVP applications
     a. ISPF-CAF applications, with the exception of DSNTEJ3C and DSNTEJ3P.

  Note: If you want to run these IVPs as part of the verification of DB2 Version 9 compatibility mode, they must first be run under Version 8 in their entirety

before you start the Version 9 migration process and must remain available for use after you complete the migration to Version 9 compatibility mode.

- Examining ″Migration Considerations″ of the *DB2 Installation Guide, (GC18-9846-00)*, the following items are of particular interest:
  - Global temporary tables require a 32K buffer pool.
  - Declared global temporary tables and static scrollable cursor result tables require a table space with a 32-KB page size because 8-KB and 16-KB page sizes are not supported for table spaces that are created in the work file database.
  - Declared global temporary tables need a 32-KB table space in the work file database.
  - There are changes to the format of the BSDS. To support up to 10000 data sets per copy for archive logs and 93 data sets per copy for active logs, the BSDS must be converted using job DSNTIJUZ
  - The work file database is the only temporary database. The TEMP database is no longer used by DB2
  - If the application uses GROUP ATTACH, then the GROUP ATTACH process is randomized so that all members running on the same z/OS image have an equal chance of getting attach.
  - Changes in the BIND PACKAGES and BIND PLAN defaults changed from CURRENTDATA YES to NO.
- Functions that are no longer supported:
  - Java stored procedures no longer run in resettable JVMs.
  - DB2-established stored procedure address spaces are no longer supported. Stored procedures must be moved to a WLM environment.
  - JBDC/SQLJ Driver for OS/390 and z/OS is no longer supported. All procedures need to be modified to work with the IBM DB2 Driver for JDBC and SQLJ.
  - Simple table spaces are no longer supported. The default is segmented.
- During the migration of the first member of a data sharing group to DB2 Version 9, 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 9 libraries exclusively.
- The Administrative Task Scheduler (ATS) as currently implemented in DB2 for z/OS is the first piece of tooling infrastructure for our next-gen Web-and Eclipse based tooling.

  The subsystem parameter ADMTPROC, in macro DSN6SPRM, saves the start procedure name of the Admin Scheduler that is associated with the DB2 subsystem. ADMTPROC cannot be updated online. Whenever DB2 starts up, it starts the Admin Scheduler that is specified in ADMTPROC, if it is not up yet. In addition, every time DB2 starts or stops, it posts an event to the Admin Scheduler so that the Admin Scheduler can execute tasks that depend on these events.

  Reference material:
  - *DB2 Installation Guide, (GC18-9846-00)*
  - *DB2 Version 9.1 for z/OS Administration Guide (SC18-9840-00)*
  - *DB2 Version 9.1 for z/OS Application Programming and SQL Guide (SC18-9841-00)*

Premigration activities

Before migrating to DB2 Version 9, application of the fallback SPE to all members of the Version 8 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 9 concatenations before invoking the DB2 installation CLIST:

- DB2.DB2910.SDSNSPFM was concatenated to ISPMLIB.
- DB2.DB2910.SDSNSPFP was concatenated to ISPPLIB.
- DB2.DB2910.SDSNSPFS was concatenated to ISPSLIB.
- DB2.DB2910.SDSNSPFT was not concatenated to ISPTLIB, as DB2 online help was not installed.

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.DB2910.SDSNCLST(DSNTINST)'
```

We filled in the first panel DSNTIPA1 as shown in Figure 14.

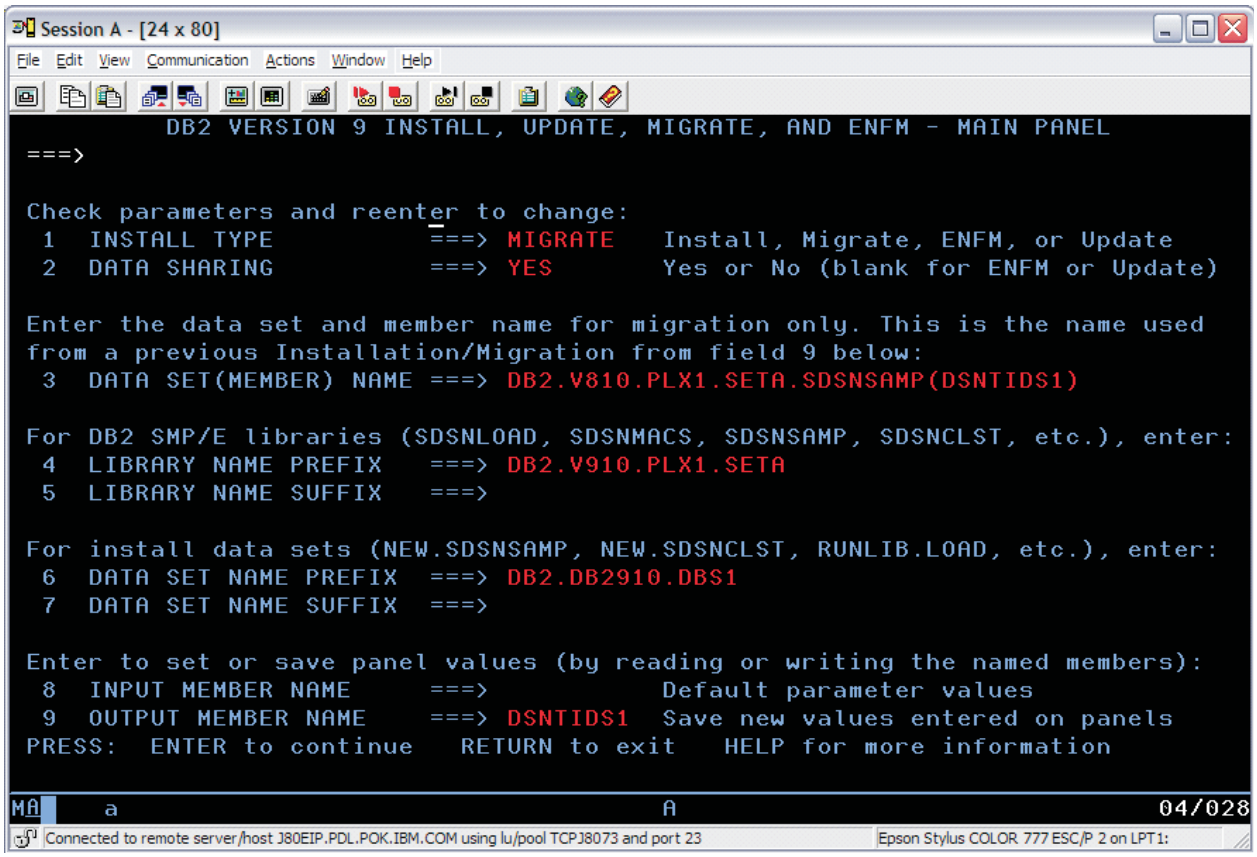


Figure 14. DSNTIPA1

When we pressed enter, the pop-up screen DSNTIPP2 appeared as shown in Figure 15.

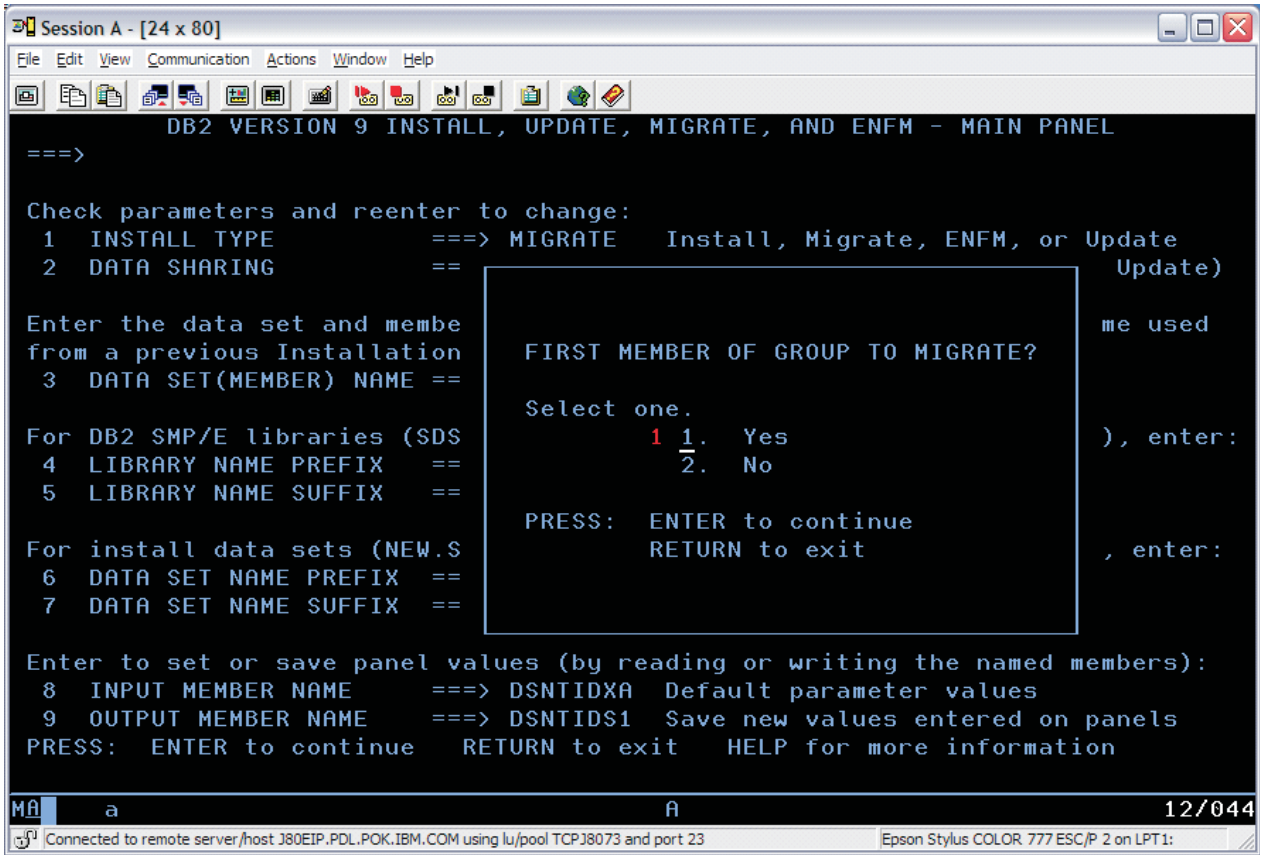


Figure 15. DSNTIPP2

We entered '**1**' to reflect that this was the first member of the data sharing group to be migrated to DB2 Version 9. From this point, we scrolled through the panels and accepted the existing values; upon completion, we placed the tailored JOBS in DB2.DB2910.DBS1.NEW.SDSNSAMP and PROCS in DB2.DB2910.DBS1.NEW.SDSNTEMP as shown in the following:

```
DSNT489I CLIST EDITING 'DB2.DB2910.DBS1.NEW.SDSNSAMP(DSNTIJMV)', INSTALL JCL
DSNT489I CLIST EDITING 'DB2.DB2910.DBS1.NEW.SDSNSAMP(DSNTIJIN)', INSTALL JCL
DSNT489I CLIST EDITING 'DB2.DB2910.DBS1.NEW.SDSNSAMP(DSNTIJTC)', INSTALL JCL
DSNT489I CLIST EDITING 'DB2.DB2910.DBS1.NEW.SDSNSAMP(DSNTIJTM)', INSTALL JCL
DSNT489I CLIST EDITING 'DB2.DB2910.DBS1.NEW.SDSNSAMP(DSNTIJIC)', INSTALL JCL
DSNT489I CLIST EDITING 'DB2.DB2910.DBS1.NEW.SDSNSAMP(DSNTIJVC)', INSTALL JCL
DSNT489I CLIST EDITING 'DB2.DB2910.DBS1.NEW.SDSNSAMP(DSNTIJSG)', INSTALL JCL
DSNT489I CLIST EDITING 'DB2.DB2910.DBS1.NEW.SDSNSAMP(DSNTIJOS)', INSTALL JCL
DSNT489I CLIST EDITING 'DB2.DB2910.DBS1.NEW.SDSNSAMP(DSNTIJEX)', INSTALL JCL
DSNT489I CLIST EDITING 'DB2.DB2910.DBS1.NEW.SDSNSAMP(DSNTIJGF)', INSTALL JCL
DSNT489I CLIST EDITING 'DB2.DB2910.DBS1.NEW.SDSNSAMP(DSNTIJFT)', INSTALL JCL
DSNT489I CLIST EDITING 'DB2.DB2910.DBS1.NEW.SDSNSAMP(DSNTIJPD)', INSTALL JCL
DSNT489I CLIST EDITING 'DB2.DB2910.DBS1.NEW.SDSNTEMP(DSNU)',     CLIST
DSNT489I CLIST EDITING 'DB2.DB2910.DBS1.NEW.SDSNTEMP(DSNH)',     CLIST
DSNT489I CLIST EDITING 'DB2.DB2910.DBS1.NEW.SDSNTEMP(DSNHC)',    CLIST
DSNT489I CLIST EDITING 'DB2.DB2910.DBS1.NEW.SDSNTEMP(DSNEMC01)', CLIST
DSNT489I CLIST EDITING 'DB2.DB2910.DBS1.NEW.SDSNSAMP(DSNTIJCX)', MIGRATE JCL
DSNT489I CLIST EDITING 'DB2.DB2910.DBS1.NEW.SDSNSAMP(DSNTIJRI)', INSTALL JCL
IKJ52338I DATA SET 'DB2.V910.PLX1.SETB.SDSNSAMP(DSNTIJRI)' NOT LINE NUMBERED, USING NONUM
DSNT489I CLIST EDITING 'DB2.DB2910.DBS1.NEW.SDSNSAMP(DSNTIJFV)', FALL BACK JCL
DSNT489I CLIST EDITING 'DB2.DB2910.DBS1.NEW.SDSNSAMP(DSNTIJUZ)', INSTALL JCL
```

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 8 are still available for use. Failure to do so will prevent verifying that a successful migration to DB2 Version 9 compatibility mode has been made.
- Ensured that no utilities are running before migrating to DB2 Version 9. When the migration to Version 9 compatibility mode has been completed, any outstanding utilities that were started under Version 8 cannot be restarted or terminated under Version 9.

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. We chose not to run this JOB at this time since we had active applications running on the DB2 V8 members during migration.

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 9. 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 16 on page 86.

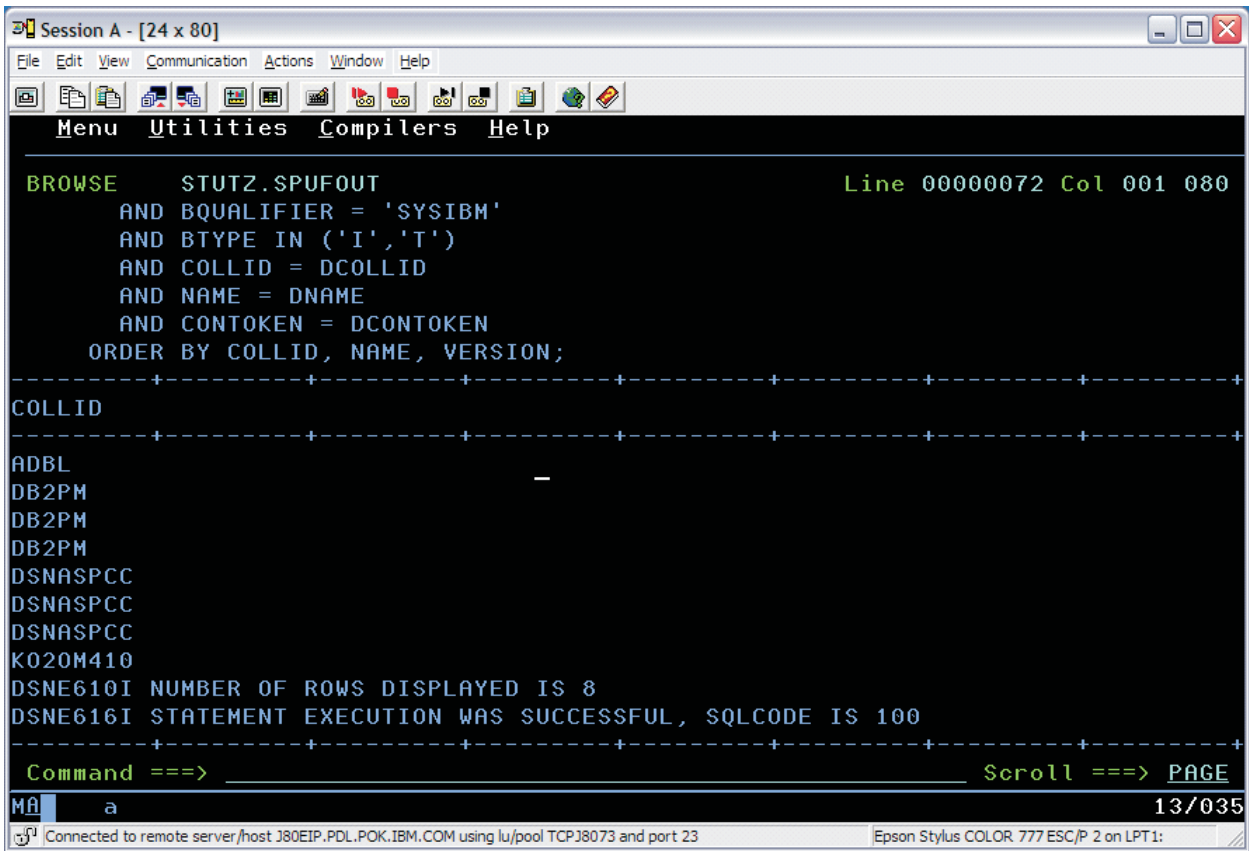


Figure 16. Query output to find packages that will be invalidated when migrating to DB2 Version 9

Migration Step 4 is another optional step to check for consistency between catalog tables through running the queries contained in DB2.DB2910.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, (GC18-9846-00)*, recommends using the Version 9 job DSNTIJIC . We followed the recommendation.

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

- **Making DB2 load modules available to TSO and batch users -** . Since we run with multiple versions of DB2 V8 and V9 we are using symbolics and extended aliases for TSO and Batch users. SDSNEXIT:

```
DB2.DB2910.DBSG.SDSNEXIT , SDSNLOAD: DB2.DBSG.SDSNLOAD
                  SYMBOLIC:   DB2.&DBSGVER..&DB2PLEX..&DBSGSET..SDSNLOAD
                 EXTENDED ALIAS:
  DEFINE ALIAS (                                          -
  NAME(DB2.DBSG.SDSNLOAD)                                -
  SYMBOLICRELATE(DB2.&DBSGVER..&DB2PLEX..&DBSGSET..SDSNLOAD))
```

- **Making DB2 CLISTs available to TSO and batch users: DSNTIJVC** - Our logon proc DB29PLX1 must again be updated to add DB2.DB2910.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 DB29PLX1), 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 updated logon proc DB29PLX1 to add DB2.DB2910.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 DB29PLX1 to reflect the concatenation of the DB2 English DB2I panels as follows:
  - DB2.DB2910.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 instructs us to stop all DB2 V8 activity or else fallback procedures may fail; prior to stopping data sharing member DBS1, we insured that there were no incomplete utilities (**-DBS1 DISPLAY UTILITY(*)**), and that no databases were in restrict or advisory status (**-DBS1 DISPLAY DATABASE(*) SPACE(*) RESTRICT** and **-DBS1 DISPLAY DATABASE(*) SPACE(*) ADVISORY**, respectively); DBS1 was then brought down.

We skipped optional **Migration Step 10** (**Back Up your DB2 Version 8 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; expect a return code of 888 if the BSDS has already been converted to the new format.

Special considerations for (Migration Step 11): Step DSBTCNVB converts your BSDS to a New Format. This can be accomplished prior to the migration .We made a decision to convert to the new format prior to the migrations. Following is the DSBTCNVB step:

```
CONVERT THE BSDS TO NEW FORMAT
  NOTE: RC = 888 MEANS BSDS WAS ALREADY CONVERTED
```

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

Migration Step 13 defines DB2 V9 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 UPDATED manually RENAME renames the current DB2 procedures in proclib. We skipped this step, however. The DB2 startup procs for DBS1 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.DB2910.DBSG.PROCLIB.

We renamed the startup procs for DBS1 that reside in PET.PROCLIB (as per the RENAME step of DSNTIJMV). Next, we copied the new V9 startup procs for DBS1 from DB2.DB2910.DBSG.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 V9 early code, we skipped **Migration Step 16**.

Member DBS1 of data sharing group DBSG was then started (**Migration Step 17**) successfully. As the DISPLAY GROUP command shows in the example below, the level of the data sharing group DBSG is now 910 and it is in compatibility mode (MODE(C)); the DB2 level of DBS1 reflects that it is now running DB2 Version 9 code.

```
RESPONSE=J80
 DSN7100I  @DBS1 DSN7GCMD
 *** BEGIN DISPLAY OF GROUP(DSNDBSG ) GROUP LEVEL(910) MODE(C)
                   PROTOCOL LEVEL(2)  GROUP ATTACH NAME(DBSG)
 ------------------------------------------------------------------
 DB2                                    DB2 SYSTEM    IRLM
 MEMBER   ID  SUBSYS CMDPREF   STATUS   LVL NAME      SUBSYS IRLMPROC
 -------- --- ----   --------  -------- --- --------  ----   --------
 DBS1       1 DBS1   @DBS1     ACTIVE   910 J80       IRS1   DBS1IRLM
 DBS2       2 DBS2   @DBS2     ACTIVE   810 JB0       IRS2   DBS2IRLM
 DBS3       4 DBS3   @DBS3     ACTIVE   810 JF0       IRS3   DBS3IRLM
 ------------------------------------------------------------------
```

Migration Step 18. We submitted and ran DSNTIJTC successfully. The job periodically issued message DSNU777I in SYSPRINT to indicate migration progress, as shown in the message DSNU777I which displays CATMAINT progress:

```
DSNU1044I PROCESSING SYSIN AS EBCDIC
DSNU050I             CATMAINT UPDATE
DSNU750I       CATMAINT UPDATE PHASE 1 STARTED
DSNU777I       CATMAINT UPDATE STATUS - VERIFYING CATALOG IS AT CORRECT LEVEL FOR MIGRATION.
DSNU777I       CATMAINT UPDATE STATUS - BEGINNING MIGRATION SQL PROCESSING PHASE.
DSNU777I       CATMAINT UPDATE STATUS - BEGINNING ADDITIONAL CATALOG UPDATES PROCESSING.
DSNU777I       CATMAINT UPDATE STATUS - UPDATING DIRECTORY WITH NEW RELEASE MARKER.
DSNU752I       CATMAINT UPDATE PHASE 1 COMPLETED
DSNU010I       UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
```

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:

- 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.

Indexes can be put into advisory rebuild pending start during migration to DB2 Version 9 when columns are added to the index; DSNTIJRI rebuilds such indexes, and **Migration Step 20** deals with this. DSNTIJRI was executed successfully and we received a return code of 4, the result of several empty indexes.

In **Migration Step 21**, DSNTIJTM was executed to assemble, link-edit, bind, and invoke DSNTIAD. DSNTIJTM ran successfully.

In **Migration Step 22** we ran job DSNTIJSG according to the instructions specified. This step ended as expected.

Special considerations for Migration Step 22:

- In migration mode, job DSNTIJSG does not create any of the objects that are required for XML schema support. You can create these objects only after you have fully migrated to Version 9.
- If you bound special SPUFI packages and plans in Version 8, you need to bind those packages again in Version 9.1. You do not need to bind the plan again. For example, to update special SPUFI packages that were created for use by SPUFI users who require a TSO terminal CCSID of 1047, issue the following commands:

```
BIND PACKAGE(TIAP1047) MEMBER(DSNTIAP) -
     ACTION(REPLACE) ISOLATION(CS) ENCODING(1047) -
     LIBRARY('prefix.SDSNDBRM')
BIND PACKAGE(SPCS1047) MEMBER(DSNESM68) -
     ACTION(REPLACE) ISOLATION(CS) ENCODING(1047) -
     LIBRARY('prefix.SDSNDBRM')
BIND PACKAGE(SPRR1047) MEMBER(DSNESM68) -
     ACTION(REPLACE) ISOLATION(RR) ENCODING(1047) -
     LIBRARY('prefix.SDSNDBRM')
```

- In Version 9.1, SPUFI provides an option to select data with a cursor isolation level of Uncommitted Read. To add a special package and plan with ISO(UR) for SPUFI users who require a TSO terminal of CCSID 1047, issue the following commands:

```
BIND PACKAGE(SPUR1047) MEMBER(DSNESM68) -
     ACTION(REPLACE) ISOLATION(UR) ENCODING(1047) -
     LIBRARY('prefix.SDSNDBRM')
BIND PLAN(SPUR1047) -
     PKLIST(*.SPUR1047.DSNESM68, -
            *.TIAP1047.DSNTIAP) -
     ISOLATION(UR) ENCODING(1047) ACTION(REPLACE)
```

Because some views might have been marked with view regeneration errors during the migration to Version 9 compatibility mode, we performed **Migration Step 23** and identified the views with the following query:

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

The query found zero rows. However, if views had been found to have regeneration errors, the following alter command would correct the errors:

```
ALTER VIEW view_name REGENERATE;
```

In **Migration Step 24** we took another image copy of the directory and catalog after they were successfully migrated to V9, and submitted job DSNTIJIC (see **Migration Step 5** on page 86 for details). Execution of DSNTIJIC completed successfully.

The next step verifies the DB2 Version 9 subsystem that is now in Compatibility Mode; only selected Version 8 IVP jobs can be executed as outlined in the DB2 Version 9.1 for z/OS Installation Guide, **Migration Step 25**. After performing the necessary modifications, we ran these IVPs and received the expected results.

Finally, optional **Migration Step 26** 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, (GC18-9846-00)*, under Chapter 10 page 361 *"Enabling stored procedures and user defined functions"*. Since we had already enabled WLM stored procedures under DB2 Version 8, this step was skipped.

DB2 V8 and V9 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 9 compatibility mode was DBS2. 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 8 startup procs for the member being upgraded with their Version 9 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.DB2910.SDSNCLST(DSNTINST)'
```

We filled in the first panel as shown in Figure 17 on page 91.

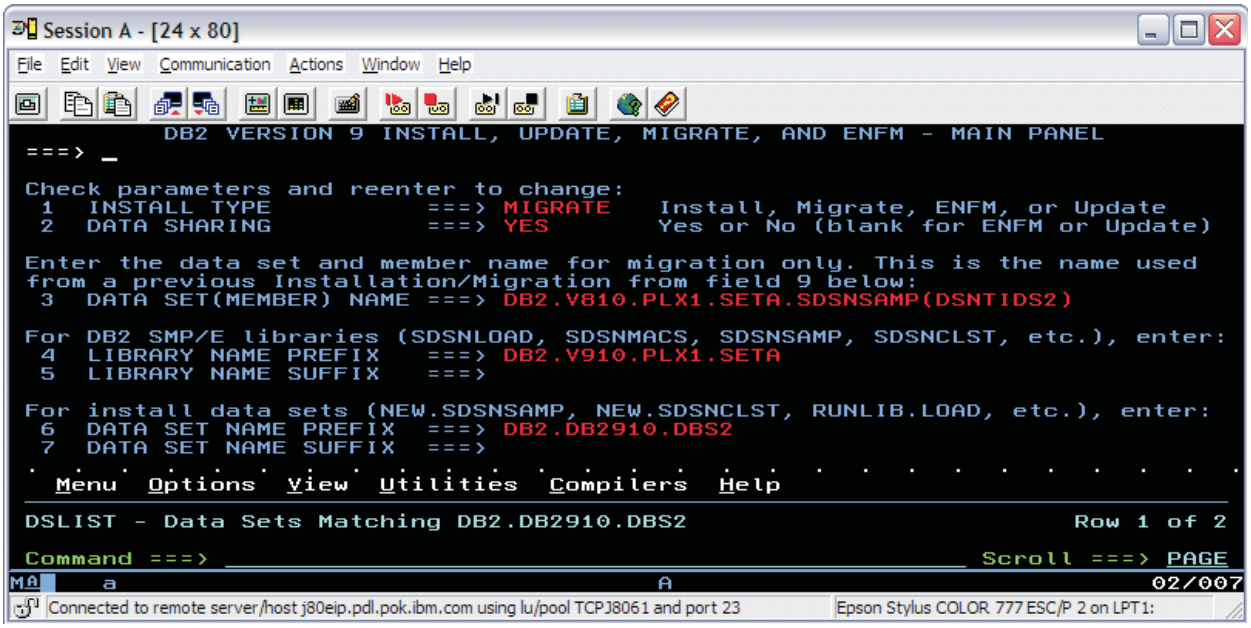


Figure 17. Executing DSNTINST in preparation for migrating the next member of the data sharing group

Pressing enter, we obtained the following pop-up screen as shown in Figure 18.

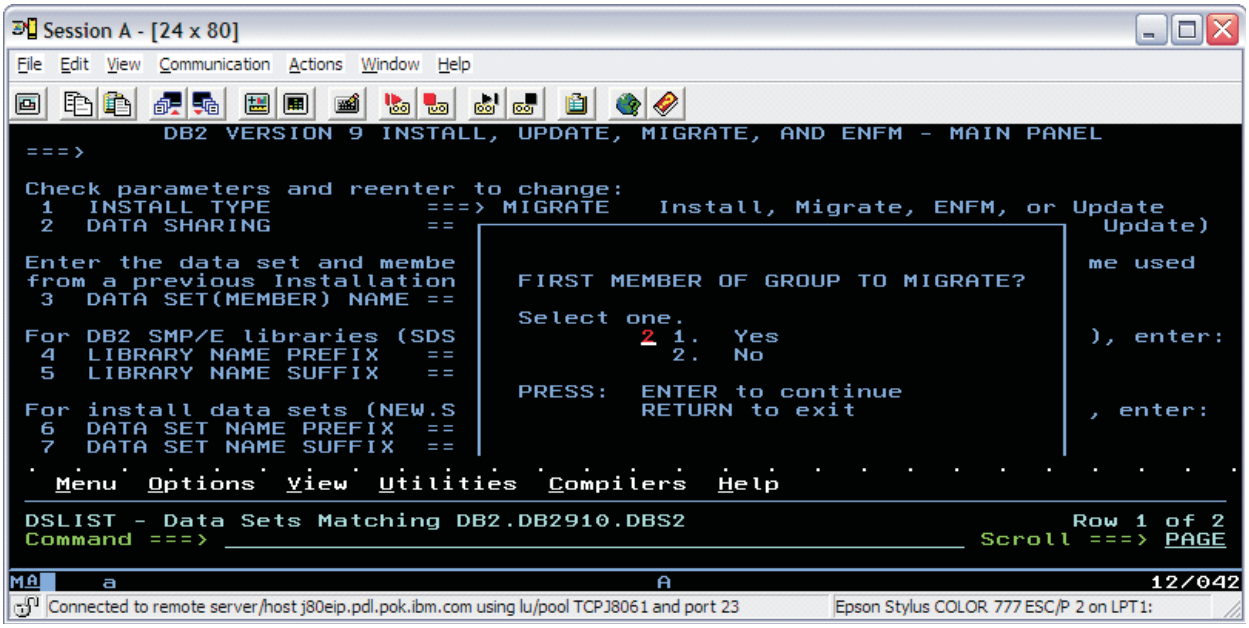


Figure 18. DSNTIPP2 pop-up screen

From this point, we scrolled through 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 DBS2, as shown in Figure 19 on page 92.

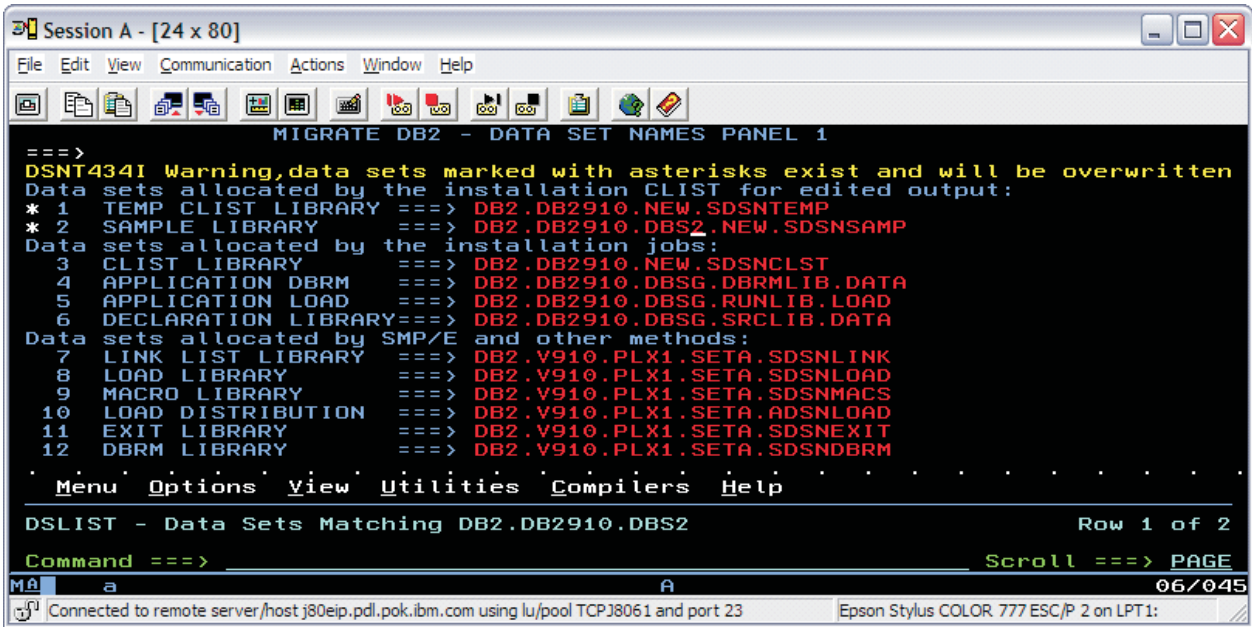


Figure 19. DSNTIPT - Data Set Names Panel 1

We placed the tailored migration JCL in DB2.DB2910.DBS2.NEW.SDSNSAMP as can be seen in the following example:

```
DSNT478I BEGINNING EDITED DATA SET OUTPUT
DSNT489I CLIST EDITING 'DB2.DB2910.DBS2.NEW.SDSNSAMP(DSNTIJMV)', INSTALL JCL
DSNT489I CLIST EDITING 'DB2.DB2910.DBS2.NEW.SDSNSAMP(DSNTIJTM)', INSTALL JCL
DSNT489I CLIST EDITING 'DB2.DB2910.DBS2.NEW.SDSNSAMP(DSNTIJGF)', INSTALL JCL
DSNT489I CLIST EDITING 'DB2.DB2910.DBS2.NEW.SDSNSAMP(DSNTIJFT)', INSTALL JCL
DSNT489I CLIST EDITING 'DB2.DB2910.DBS2.NEW.SDSNSAMP(DSNTIJFV)', FALL BACK JCL
DSNT489I CLIST EDITING 'DB2.DB2910.DBS2.NEW.SDSNSAMP(DSNTIJUZ)', INSTALL JCL
```

DBS2 was then brought down and DSNTIJUZ was executed after removing the SMP/E step; it ran successfully.

Next, we used steps RENAME and DSNTIPM of job DSNTIJMV to rename the existing Version 8 startup procedures for DBS2 and to add the new Version 8 startup procedures to proclib.

We then started DBS2 successfully in compatibility mode, as can be seen in the following example:

```
RESPONSE=J80
 DSN7100I  @DBS1 DSN7GCMD
 *** BEGIN DISPLAY OF GROUP(DSNDBSG ) GROUP LEVEL(910) MODE(C)
                   PROTOCOL LEVEL(2)  GROUP ATTACH NAME(DBSG)
 ------------------------------------------------------------------------
 DB2                                      DB2 SYSTEM    IRLM
 MEMBER   ID  SUBSYS CMDPREF   STATUS   LVL NAME      SUBSYS IRLMPROC
 -------- --- ----   --------  -------- --- --------  ----   --------
 DBS1       1 DBS1   @DBS1     ACTIVE   910 J80       IRS1   DBS1IRLM
 DBS2       2 DBS2   @DBS2     ACTIVE   910 JB0       IRS2   DBS2IRLM
 DBS3       4 DBS3   @DBS3     ACTIVE   810 JF0       IRS3   DBS3IRLM
 ------------------------------------------------------------------------
```

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

```
                RESPONSE=J80
DSN7100I  @DBS1 DSN7GCMD
*** BEGIN DISPLAY OF GROUP(DSNDBSG ) GROUP LEVEL(910) MODE(C)
                   PROTOCOL LEVEL(2)  GROUP ATTACH NAME(DBSG)
------------------------------------------------------------------------
DB2                                      DB2 SYSTEM    IRLM
MEMBER   ID  SUBSYS CMDPREF   STATUS   LVL NAME      SUBSYS IRLMPROC
-------- --- ----   --------  -------- --- --------  ----   --------
DBS1       1 DBS1   @DBS1     ACTIVE   910 J80       IRS1   DBS1IRLM
DBS2       2 DBS2   @DBS2     ACTIVE   910 JB0       IRS2   DBS2IRLM
DBS3       4 DBS3   @DBS3     ACTIVE   910 JF0       IRS3   DBS3IRLM
------------------------------------------------------------------------
```

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 9. The process is outlined below:

- "Preparing for new function mode"
- "Enabling new function mode" on page 96
- "Running in new function mode" on page 98
- "Verifying the installation using the sample applications" on page 98

Preparing for new function mode

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

- **Important:** All members of a data sharing group must have been successfully migrated to Version 9.1 compatibility mode before commencing the enabling-new-function mode process.
- A point of consistency needs to be created for the catalog and directory before enabling new-function mode. The Quiesce Utility should be used to establish a point of consistency for the catalog and directory table spaces; note that DSNDB01.SYSUTILX should be quiesced by itself. Updates to the DB2 catalog and directory should be avoided while in enabling new-function mode. **This is the only time when applications were brought down; planning is therefore essential to reduce the amount of down time.**
- Run the installation CLIST using the ENFM option on panel DSNTIPA1.

After insuring a point of consistency for the catalog and directory , the installation CLIST was executed; panel DSNTIPA1 was completed as shown in Figure 20 on page 94.

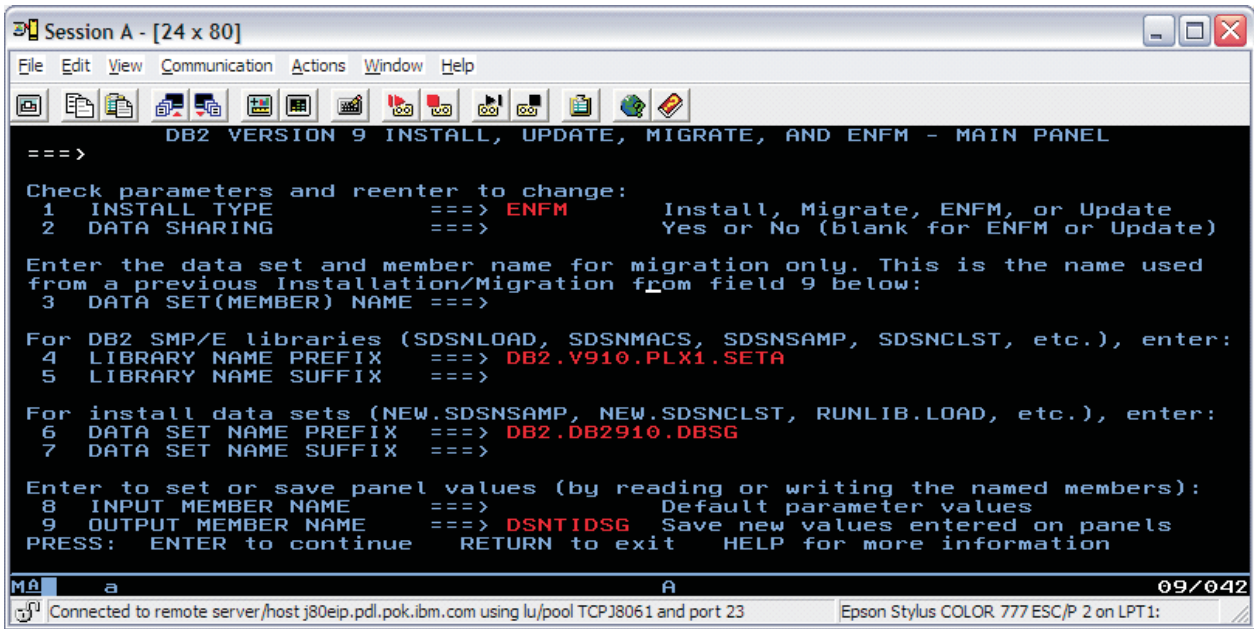


Figure 20. Executing DSNTINST in preparation for enabling-new-function-mode

Press enter twice to display panel DSNTIP00; space was calculated as shown in Figure 21 and Figure 22 on page 95.

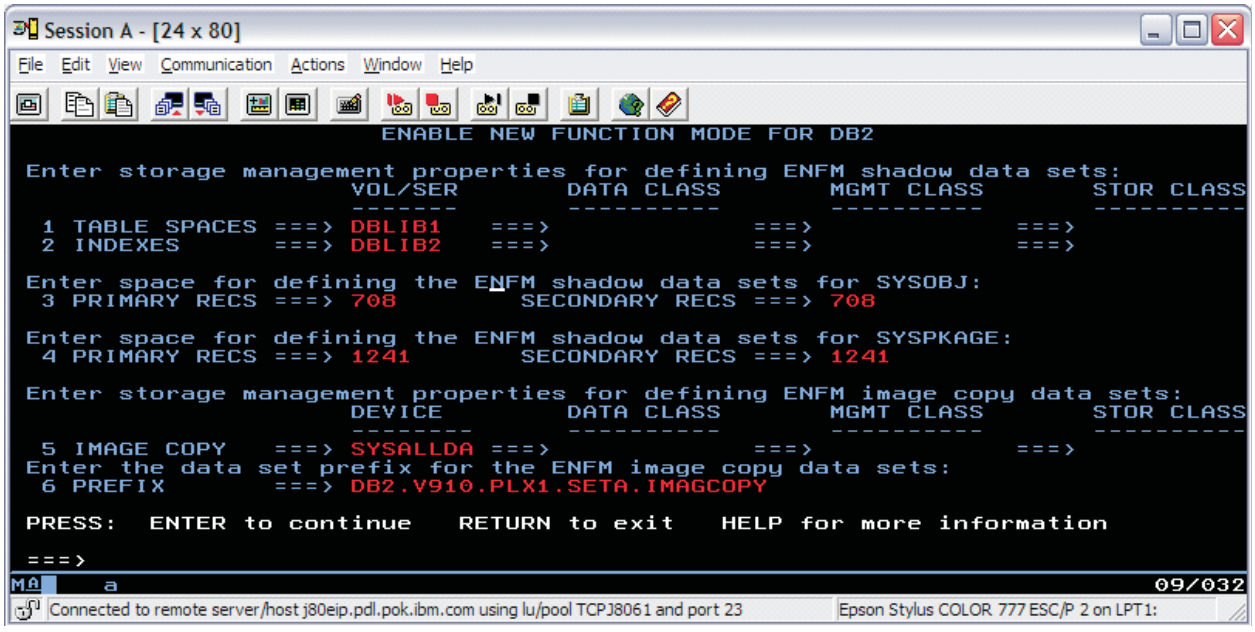


Figure 21. DSNTIP00 first panel

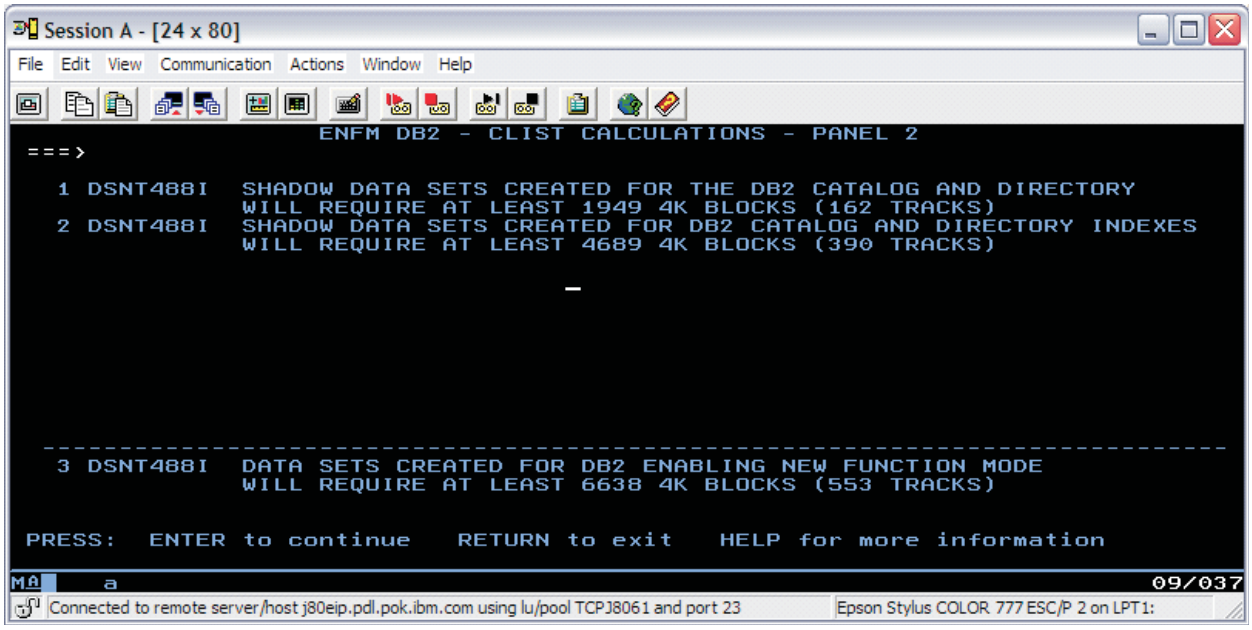


Figure 22. DSNTIP00 second panel

We accepted calculated values and pressed enter to continue.

This was the last panel displayed. When we pressed enter, the generation of the enabling-new-function mode job along with the DB2 Version 9 sample jobs, occurred as shown in the following three screen images:

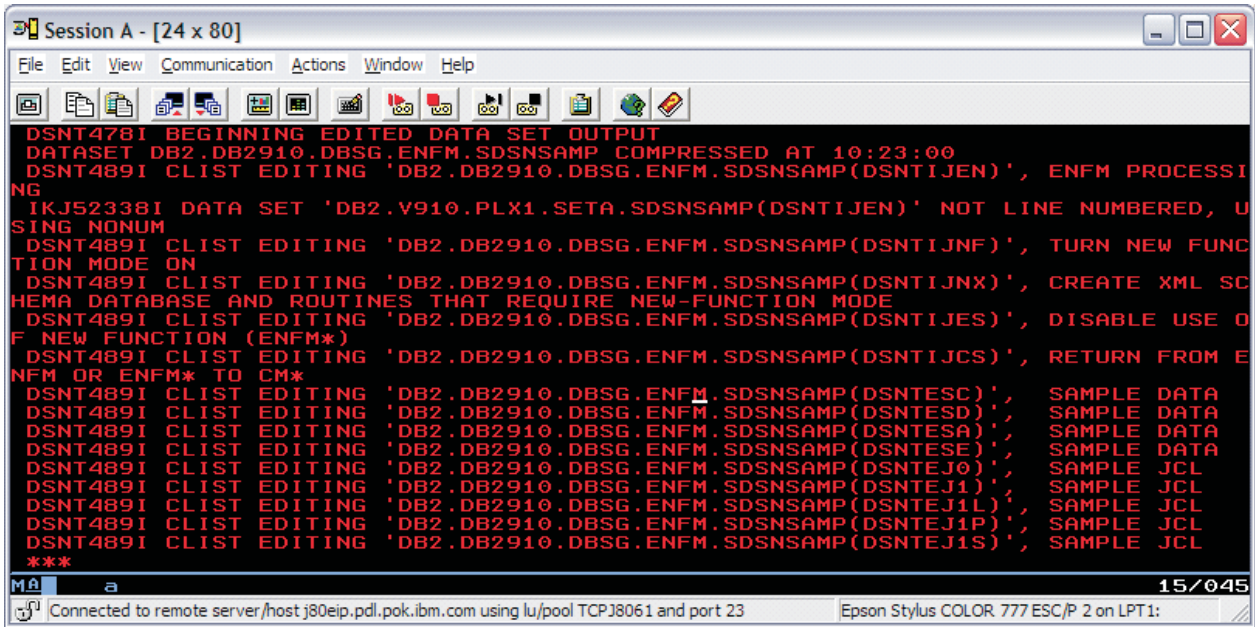


Figure 23. DSNT478I beginning data set output

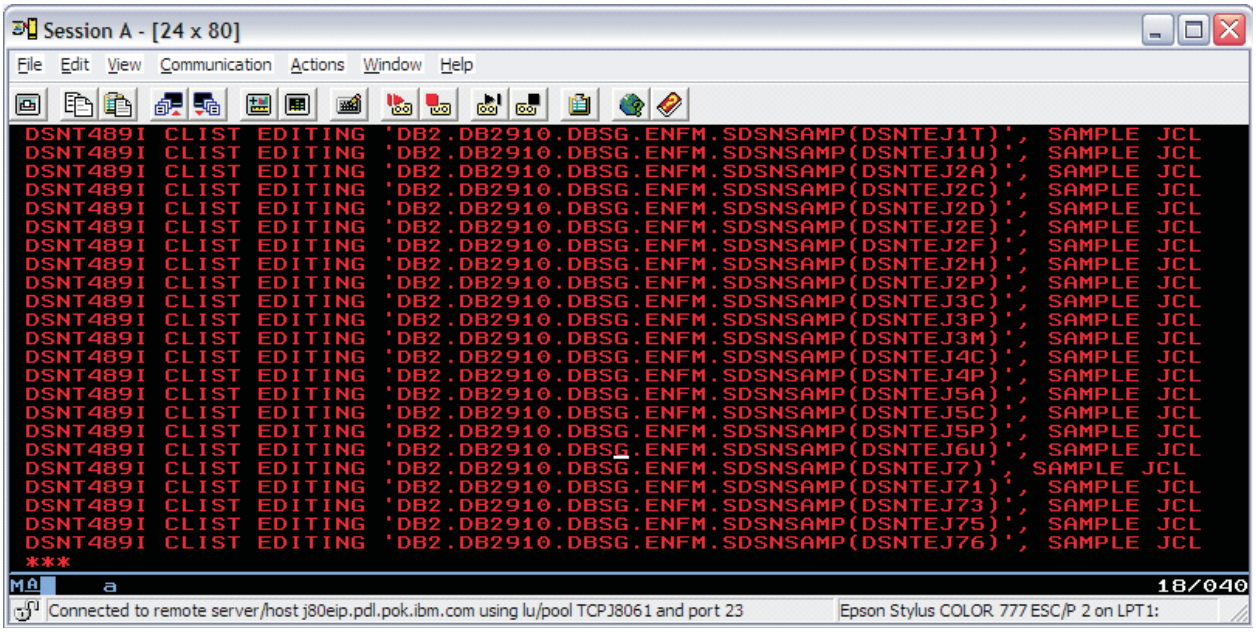


Figure 24. DSNT489I CLIST editing

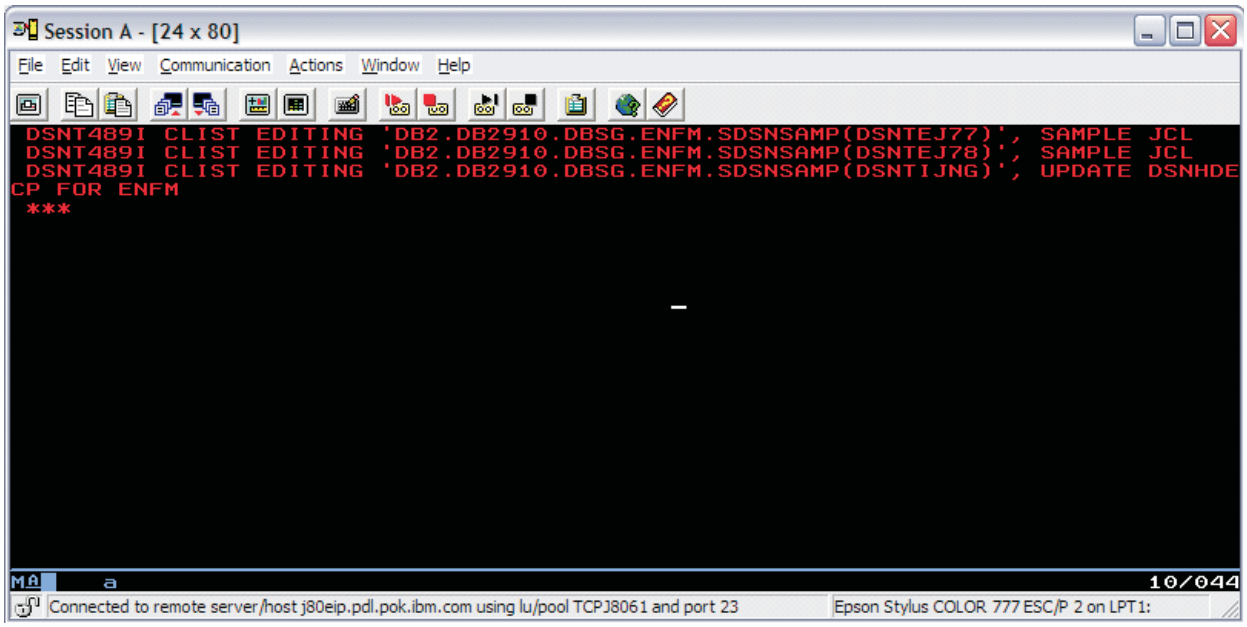


Figure 25. Completion of the preparation before enabling Version 9 new function mode

Enabling new function mode

For **Step 1**, we executed DSNTIJEN and received the messages shown in Figure 27; DSNTIJEN performs the following functions:

- Saves the current RBA or LRSN in the BSDS
- Changes types and lengths of existing catalog columns
- Changes buffer pool for the SYSOBJ table space
- Changes page size of the SYSOBJ table space

- Copies the RTS from the user table spaces to new table spaces in the catalog
- Creates a new index, DSNRTX03, on SYSINDEXSPACESTATS.

The following are the results of the conversion steps:

```
DSNUECM0 - CATENFM START PHASE 1 STARTED
DSNUECM0 - CATENFM START STATUS - VERIFYING CATALOG IS AT CORRECT LEVEL FOR ENFM
DSNUECM0 - CATENFM START PHASE 1 COMPLETED
DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = DSNENFM.ENFM0100
DSNUGTIS - PROCESSING SYSIN AS EBCDIC
DSNUGUTC -  CATENFM CONVERT INPUT SYSOBJ
DSNUECM0 - CATENFM CONVERT PHASE 1 STARTED
DSNUECM0 - CATENFM CONVERT PHASE 1 COMPLETED
DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0


DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = DSNENFM.ENFM0110
DSNUGTIS - PROCESSING SYSIN AS EBCDIC
DSNUGUTC -  CATENFM CONVERT INPUT SYSPKAGE
DSNUECM0 - CATENFM CONVERT PHASE 1 STARTED
DSNUECM0 - CATENFM CONVERT PHASE 1 COMPLETED
ALTER COLUMN "SEQNO" SET DATA TYPE
DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0



DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = DSNENFM.ENFM1200
DSNUGTIS - PROCESSING SYSIN AS EBCDIC
DSNUGUTC -  CATENFM CONVERT INPUT SYSRTSTS
DSNUECM0 - CATENFM CONVERT PHASE 1 STARTED
DSNUECM0 - CATENFM CONVERT PHASE 1 COMPLETED
DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
```

Step 2 recommends taking an image copy of the catalog and directory at this point.

For **Step 3**, we ran DSNTIJNF which places the DB2 subsystem in new function mode; the job ended with return code zero as shown below:

```
DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = DSNENFM.ENFM9700
DSNUGTIS - PROCESSING SYSIN AS EBCDIC
DSNUGUTC -  CATENFM COMPLETE
DSNUECM0 - CATENFM COMPLETE PHASE 1 STARTED
DSNUECM0 - CATENFM COMPLETE STATUS - ENTERING NEW FUNCTION MODE (NFM).
DSNUECM0 - CATENFM COMPLETE PHASE 1 COMPLETED
DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
```

Step 4 is concerned with executing DSNTIJNX, which creates objects for XML Schema Repository (XSR) support. We submitted this job and it ran to completion successfully.

In **Step 5**, DSNTIJNG rebuilds DSNHDECP to specify new function mode as the default by specifying NEWFUN=YES.

Note: If you use more than one DSNHDECP member, modify and update each to use NEWFUN=YES.

To verify the data sharing group was now in new function mode, we issued a DISPLAY GROUP COMMAND; as can be seen in Figure 26 on page 98, MODE(N) has replaced MODE(C):

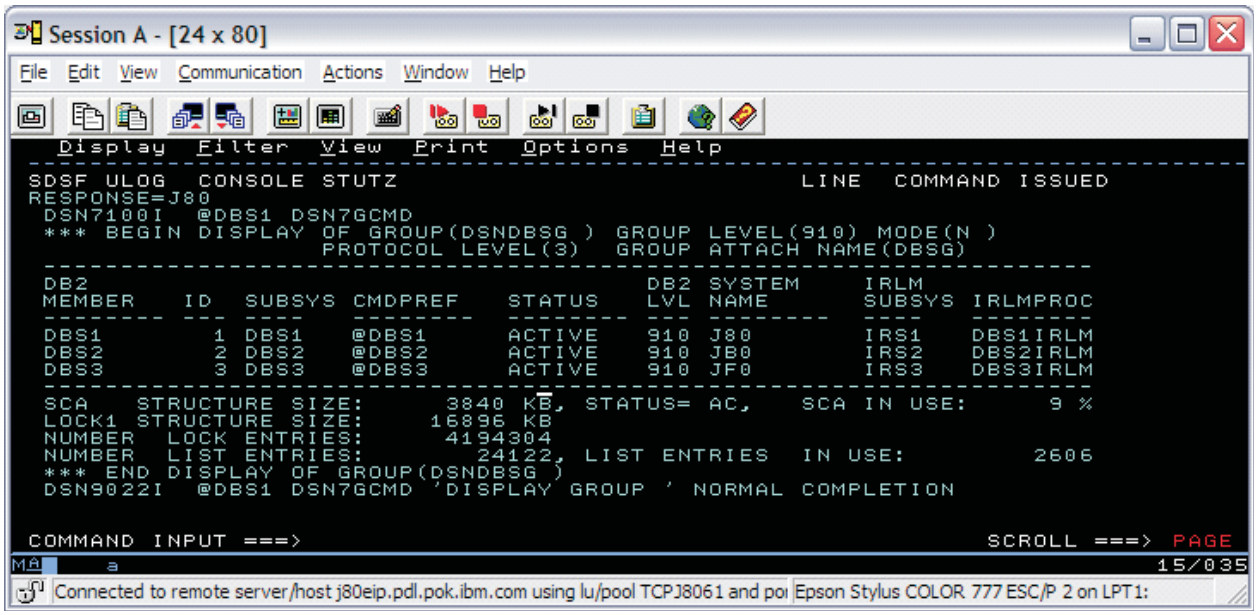


Figure 26. DISPLAY GROUP command showing the data sharing group is now in new function mode

Running in new function mode

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.

Verifying the installation using the sample applications

Using the sample applications provided in DB2.DB2910.DBSG.ENFM.SDSNSAMP, we performed verification of DBSG migration to DB2 Version 9 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.DB2910.DBSG.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.DB2910.DBSG.PROCLIB. This library must be APF authorized (we dynamically added it to the APF authorization list before proceeding).

Planning for verification

Before performing any of the verification phases, you must make certain decisions about your verification strategy. DB2 system administrators and system administrators for ISPF, TSO, batch, IMS, and CICS must be involved in these decisions. With these system administrators:

- Determine the verification phases that you plan to perform. Examine the description of each verification phase in this chapter, and determine which phases apply to your needs.
- Identify any phases that you want to modify before you perform them. Verification is designed to run with little interaction on your part. This chapter does not discuss how to modify any of the phases, but you can adapt any of the seven phases to your needs. If this is your intent, identify and describe any modifications you plan to make.
- Establish additional testing steps to complete the verification. The verification phases and the jobs that you run to perform them are valuable tools for testing DB2. They are not a substitute for a thorough subsystem test. You must plan and perform your own additional testing to complete the verification. To help you assess which additional tests might be necessary, examine the sample applications that are provided with DB2.

We executed the following IVP jobs after every change to the environment (hardware, software) . Your may choose to execute more or less than what we schedule to run . Based on your needs you may choose to run the IVP's on a different cycle than what we have setup.

```
DSNTEJ1
DSNTEJ1L
DSNTEJ1P
DSNTEJ1S
DSNTEJ2A
DSNTEJ2C
DSNTEJ2D
DSNTEJ2E
DSNTEJ2H
DSNTEJ2P
DSNTEJ3C
DSNTEJ3M
DSNTEJ3P
DSNTEJ6U
DSNTEJ7
DSNTEJ71
DSNTEJ73
```

| DSNTEJ75
| DSNTEJ76
| DSNTEJ77
| DSNTEJ78

Chapter 7. Implementing IMS JDBC Connector (formerly IMS Java)

The IMS JDBC Connector allows you to write Java application programs that access IMS databases using JDBC. JDBC is the SQL-based standard Java interface for database access. The IMS Java implementation of JDBC supports a selected subset of the full facilities of the JDBC 2.1 API. This subset allows you to do everything that traditional IMS applications that use DL/I calls can do.

We used the following documentation to help us implement IMS Java:

- *IMS Version 9: IMS Java Guide and Reference*, SC18-7821

Setting up the Java API libraries

To use IMS Java, the following API libraries need to be set up in the UNIX System Services environment:

- IBM SDK for z/OS Java 2 Technology Edition
- IMS Java API

Steps for installing the IBM SDK for z/OS Java

We performed the following steps to install the IBM SDK for z/OS Java:

1. Created a new zFS file system *OMVSWS.JAVA14.UK04987.ZFS*

2. Created new directory */java/java14UK04987*

3. Mounted the zFS file system to the new directory

4. Downloaded the latest IBM Java SDK for z/OS from

   ```
   http://www.ibm.com/servers/eserver/zseries/software/java/
   ```

   into the new directory

5. Extracted the SDK file:

   ```
   pax –ppx –rzf UK04987.PAX.Z
   ```

6. Set up *symlink /java/curimsj* to point to */java/java14UK04987/J1.4*:

   ```
   ln –s /java/java14UK04987/J1.4 /java/curimsj
   ```

Steps for installing the IMS Java API

We performed the following steps to install the IMS Java API:

1. Restored IMS Java HFS to *OMVSWS.IMS910.D012805.FS*

2. Created new IMS Java directory */ims910/d012805*

3. Mounted the HFS file system to the new directory

4. Created symlink */imsjava/current* to point to */ims910/d012805*.

Running the dealership sample

The IMS Java API came with the dealership sample application and sample databases. To verify that IMS Java was properly installed, we set up and ran the sample application in a JMP region.

Steps for installing the sample application

We performed the following steps to install the sample application:

1. Created a new zFS file system and mounted it on */imsjava/dealership*. This directory would hold all files used by the sample application.

2. Copied */imsjava/current/samples/samples.tar* to */imsjava/dealership/samples.tar*

3. Extracted the tarball:

```
tar xvf /imsjava/dealership/samples.tar
```

A directory called "*samples*" and a file called "*samples.jar*" should be extracted. The file *samples.jar* contained the compiled binary files of the dealership application.

Steps for installing the sample databases

We performed the following steps to install the sample databases:

1. Copied the extracted database sources to a PDS:

```
ogetx '/imsjava/dealership/samples/dealership/databases/' 'D10.IMSJAVA.DEALERSHIP.IVP' lc suffix
```

2. Modified the IMS system definition stage 1 input statement to include the dealership databases and transaction:

```
DATABASE       ACCESS=UP,DBD=AUTODB
DATABASE       ACCESS=UP,DBD=EMPDB2
DATABASE       ACCESS=UP,DBD=AUTOJL
DATABASE       ACCESS=UP,DBD=SINDEX11
DATABASE       ACCESS=UP,DBD=SINDEX22

APPLCTN PSB=AUTPSB11,PGMTYPE=TP,SCHDTYP=PARALLEL

TRANSACT CODE=AUTRAN11,PRTY=(7,10,2),INQUIRY=NO,MODE=SNGL,  X
      MSGTYPE=(SNGLSEG,NONRESPONSE,99)
```

3. Loaded the dealership sample databases.

Steps for setting up the JMP regions

We performed the following steps to set up the Java Message Processing (JMP) regions:

1. Created the master JVM member DLRJVMMS in our IMS proclib:

```
-Dibm.jvm.shareable.application.class.path= >
/imsjava/dealership/samples.jar:
-Dibm.jvm.trusted.middleware.class.path= >
/imsjava/current/imsjava.jar:
-Dibm.jvm.events.output=stdout
-verbose:Xclassdep
-Xinitacsh128k
-Xinitsh128k
-Xmaxf0.6
-Xminf0.3
-Xmx64M
-Xoss400k
-verbose:gc
-Xcheck:nabounds
```

---

2. Created the worker JVM member *DLRJVMWK*:

```
-verbose:Xclassdep
-Xmaxf0.6
-Xminf0.3
-Xmx64M
-Xoss400k
-verbose:gc
-Xcheck:nabounds
```

---

3. Created the environment JVM member *DLRJVMEV*:

```
LIBPATH=/java/curimsj/bin/classic:/java/curimsj/bin:/imsjava/current/
```

---

4. Created the application JVM member *DFSJVMAP*:

```
AUTPSB11=samples/dealership/ims/IMSAuto
```

Note: The name of this member must be *DFSJVMAP*.

---

5. Created the JMP output and error HFS files:
   a. Created the JMP output file */imsjava/dealership/logs/JVM.out*
   b. Created the JMP error file */imsjava/dealership/logs/JVM.err*
   c. Changed the access permission of both files to **777**:

```
chmod 777 /imsjava/dealership/logs/JVM.out
chmod 777 /imsjava/dealership/logs/JVM.err
```

---

6. Created JMP procedure *DLRJMP91*:
   a. Set the following parameters:

```
XPLINK=Y,
ENVIRON=DLRJVMEV,
JVMOPWKR=DLRJVMWK,
JVMOPMAS=DLRJVMMS
```

   b. Set the JAVAOUT and JAVAERR DD statements:

```
//JAVAOUT  DD PATH='/imsjava/dealership/logs/JVM.out'
//JAVAERR  DD PATH='/imsjava/dealership/logs/JVM.err'
```

Steps for running the sample application

We performed the following steps to run the sample application:

1. Created MFS formats for the dealership sample application. The Java source codes in */imsjava/dealership/samples/dealership/ims/io* contained the input/output message formats used by the application.

---

2. Started the JMP region DLRBJMP1 using the JMP procedure, the transaction AUTRAN11, and the program AUTPSB11.

---

3. Defined OMVS segment to the user ID that would be used to execute the application:

   ```
   ALTUSER JDU40001 OMVS(HOME('/u/jdu40001') AUTOUID PROGRAM('/bin/sh'))
   ```

   The JMP region would fail with U0101 abend if no OMVS segment was defined.

---

4. Logged into the IMS console to execute the application using the MFS formats. There were six command codes supported by the sample application as indicated in the */imsjava/dealership/samples/dealership/ims/README* file.

Chapter 8. Implementing IMS SOAP Gateway

IMS SOAP Gateway allows you to enable your IMS application to become a Web Service. Different types of clients can submit SOAP requests into IMS to drive the business logic of the back end IMS application.

We used the following documentation to help us in implementing IMS SOAP Gateway:

- IMS SOAP Gateway Documentation from the IMS SOAP Gateway website:
  `www.ibm.com/software/ims/soap`
- *IMS Version 9: IMS Connect Guide and Reference*, SC18-9287
- *IMS Version 9: Utilities Reference: System*, SC18-7834
- *IMS Version 9: IMS Java Guide and Reference*, SC18-7821

Setting up the IMS SOAP Gateway

To use IMS SOAP Gateway, the following components needed to be installed:

- **IMS SOAP Gateway server:** Performs the communication between the web service client and IMS Connect. The SOAP Gateway accepts a SOAP message from the client, converts it to an IMS XML input message, and sends it to IMS Connect using TCP/IP. It then processes and returns the output message to the client in the same manner.
- **IMS Connect user exit routine HWSSOAP1:** Called by the IMS OTMA Adapter to process TCP/IP client data. It translates client data to EBCDIC and appends proper headers for IMS. It also translates IMS data back to ASCII and removes the headers before sending it back to the client.
- **IMS Connect XML Adapter:** Converts incoming XML messages into the format understood by the application.

Steps for installing the IMS SOAP Gateway

We performed the following steps to set up the IMS SOAP Gateway on a Windows system and on an SUSE Linux Enterprise Server (SLES) system:

1. Downloaded IMS SOAP Gateway from the IMS SOAP Gateway website at:
   `www.ibm.com/software/ims/soap`

---

2. Ran the *imssoap920* file
   - For Windows:
     a. Executed the *imssoap920win.exe* file
   - For SLES:
     a. Changed the permission bits of the *imssoap920zlinux.bin* file to make it executable:
        `chmod +x imssoap920zlinux.bin`
     b. Executed the file in console mode:
        `./imssoap920zlinux.bin –console`

---

3. Followed the InstallShield Wizard to complete the installation.

Steps for installing the user exit routine

We performed the following steps to install the user exit routine:

1. Downloaded the user exit routine *HWSSOAP1* from the IMS SOAP Gateway website into a PDS

2. Compiled and linked the exit to the IMS Connect resource library

3. Added the routine to the exit concatenation in the IMS Connect configuration file:

   ```
   EXIT=(HWSCSLO0,HWSCSLO1,HWSIMSO0,HWSSOAP1)
   ```

4. Restarted IMS Connect to pick up the change.

Steps for installing the XML Adapter

We performed the following steps to install the XML Adapter:

1. Downloaded and installed APAR PK24912

2. Added the adapter statement in the IMS Connect configuration member:

   ```
   ADAPTER=(XML=Y)
   ```

3. Created exit definition *HWSEXIT0* in the IMS proclib:

   ```
   EXITDEF(TYPE=XMLADAP,EXITS=(HWSXMLA0),ABLIM=1,COMP=HWS)
   ```

4. Specified the exit definition in the BPE configuration member:

   ```
   EXITMBR=(HWSEXIT0,HWS)
   ```

5. Setup character conversion support from EBCDIC to UTF-8 by issuing the command:

   ```
   SETUNI ADD,FROM=1140,TO=1208,TECH=R,DSN=SYS1.SCUNTBL,VOL=PETPA2
   ```

   Note: UTF-8 to EBCDIC conversion support is also required. The CCSID for UTF-8 is 1208.

6. Restarted IMS Connect to pick up the change.

Enabling IMS applications as web services

We enabled two IMS applications as web services – one written in Java and the other written in COBOL.

Steps for enabling a Java application as a web service

The XML Adapter feature was not available when we converted our Java application, and therefore we had to modify the application to process XML data.

We performed the following steps to enable a Java application as a web service:

1. Generated the WSDL file using WebSphere Developer for zSeries

---

2. Generated XML schema using the IMS *DLIModel* utility. The XML schema describes the XML view of an IMS database.
   a. Created a DLIModel control statement with the following parameters:

```
OPTIONS PSBds=D10.IMSJAVA.DEALERT.JOURNAL.NEW
    DBDds=D10.IMSJAVA.DEALERT.JOURNAL.NEW
    GenJavaSource=YES
    Package=samples.dealership
    GenXMLSchemas=RETRIEVE
```

   Note: Our environment was set up to store the XML documents in decomposed mode.

   b. Ran *DLIModel* utility using BPXBATCH

---

3. Modified the master and worker JVM members to include the XML schema directory:

```
-Dhttp://www.ibm.com/ims/schema-resolver/file/path=          >
/imsjava/dealertest/dlimout/
```

---

4. Modified the Java application to process XML data:
   a. Needed to trim the incoming messaging before parsing it to remove the unrecognizable extra characters attached to the end of the message
   b. The Java API *javax.xml.parsers.DocumentBuilder* was used to parse the incoming XML message
   c. The IMS Java methods *storeXML* and *retrieveXML* were used to store/retrieve XML documents:

```
String query = "SELECT retrieveXML(OrderSegment) AS OrderXMLDoc
FROM Dealer.OrderSegment WHERE DealerSegment.DealerNo = '1234' AND
OrderSegment.OrderNo = '123456'
```

   d. The output messages had to be converted to ASCII characters
   e. The output messages had to be converted to lower cases. This was because the deserializer used element names starting with lowercases to match the Java standard.

---

5. Deployed the WSDL file using the deployment utility.

Steps for enabling a COBOL application as a web service

We performed the following steps to enable a COBOL application as a web service:

1. Installed the XML Adapter

---

2. Created the COBOL copybook file

---

3. Generated the web service artifacts using WebSphere Developer for zSeries, including the WSDL file, the XML converter program, and the correlator file

---

4. Uploaded the XML converter program to the mainframe host

   _____

5. Compiled the XML converter program

   _____

6. Restarted IMS Connect to pick up the change

   _____

7. Deployed the WSDL file using the deployment utility.

   _____

Chapter 9. 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 V1R8

z/OS UNIX made several enhancements in z/OS V1R8. In this section, we cover the following topics:

z/OS UNIX Directory List

In this section we just wanted to make you aware of a new ISPF utility, namely the z/OS UNIX Directory Listing Utility.

This ISPF panel based utility can be accessed through option 17 on ISPF Utilities Menu (option 3). It enables users to edit, browse, create, delete, rename, copy, and replace z/OS UNIX files from within ISPF. Its interface is very similar to that of ISPF Data Set List Utility (3.4).

This directory list utility is designed to assist mainly the users that don't spend much time with z/OS UNIX and are more experienced with the TSO/ISPF interface. It handles basic z/OS UNIX file tasks and is not aimed at users such as z/OS UNIX file system administrators.

For more details and instructions on how to use this utility, see the z/OS UNIX Directory List Utility (Option 3.17) section within the *z/OS ISPF User's Guide Vol II*, SC34-4823.

Setting and changing the file format from the UNIX System Services shell

Starting with z/OS V1R8, UNIX System Services users now can set the format attribute of a file using the *extattr* z/OS UNIX shell command. The *extattr* command will accept a ″-F″ option flag with values indicating the format of the file and then set the file format accordingly.

Here are the different types of values, both formats and text data delimeters, that one can assign to a file using the extattr command:

```
** Format:
NA not specified
BIN  binary data

** Text data delimeters:
NL new line
CR carriage return
LF line feed
CRLF carriage return followed by line feed
LFCR line feed followed by carriage return
CRNL carriage return followed by new line
```

Please note that the display capability of the file format will not be added to the *extattr* command since the *ls* command already has this capability. Please see the examples below for details.

Example 1: Displaying the values that are assigned to a file using the *extattr* command

The second column of the ″*ls -H*″ command will display the values that are assigned to a file using the *extattr* command, such as:

```
ls -H tempdir/

total 0
-rw-r--r--  ----   1 LORAIN0  sys1           0 Sep 11 08:23 tempFile1
-rw-r--r--  ----   1 LORAIN0  sys1           0 Sep 11 08:23 tempFile2
```

The above output shows that tempFile1 and 2 are not assigned a file format yet.

Example 2: Adding the BIN format by using the *extattr* command

Using the *extattr* command below we can add the BIN format to tempFile1:

```
extattr -F BIN tempdir/tempFile1
```

The *ls* command now shows:

```
ls -H tempdir/

total 0
-rw-r--r--  bin    1 LORAIN0  sys1           0 Sep 11 08:23 tempFile1
-rw-r--r--  ----   1 LORAIN0  sys1           0 Sep 11 08:23 tempFile2
```

Example 3: Adding the CRNL text data delimiter to a file

Using the below *extattr* command we can add the CRNL text data delimiter to the tempFile2 file:

```
extattr -F CRNL tempdir/tempFile2
```

Now the *ls* command displays:

```
ls -H tempdir/

total 0
-rw-r--r--  bin    1 LORAIN0  sys1           0 Sep 11 08:23 tempFile1
-rw-r--r--  crnl   1 LORAIN0  sys1           0 Sep 11 08:23 tempFile2
```

For more information on the *extattr* command please see *z/OS UNIX System Services Command Reference*, SA22-7802.

z/OS UNIX System Services: Displaying z/OS UNIX Latch Contention

The D OMVS,WAITERS | W display command was implemented in z/OS V1R7. In z/OS V1R8 it is enhanced to provide additional information. Originally it displayed z/OS UNIX Mount Latch activity and outstanding cross system messages. Starting with z/OS V1R8, it displays z/OS UNIX File Latch activity and all other waiting threads in the system as well.

Following is some of the information that you can display using this command (remember that the output is for the specific system on which the command was entered):

- The task that is holding the LFS Mount or File System Latch
- The reason why the task started holding the latch
- What that task is doing
- The tasks waiting for that task 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 four different sections: ″Mount Latch Activity″, ″Outstanding Cross System Messages″, ″File System Latch Activity″ and ″Other Waiting Threads″.

If there is some ″Mount Latch Activity″ or ″File System Latch Activity″ in the system, the related section displays the following:

- Who is holding the latch
- Who is waiting for the Mount latch
- What the holder is doing at the moment
- How long the latch has been held
- How long each waiter has been waiting for the latch
- The file that the File System Latch holder is currently accessing
- If the File System Latch is held exclusively or shared

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.

Similar information (please see the examples below for details) is displayed for all other waiting threads as well.

Please see ″Understanding UNIX System Services Latch Contention″ in *z/OS MVS Diagnosis: Reference* for a better understanding of the UNIX System Services latch contention concept.

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 and File System Latch Activity, just like Sample 5.

Sample 6: The ″File System Latch Activity″ section, new in z/OS V1R8:

```
FILE SYSTEM LATCH ACTIVITY:
         USER   ASID     TCB            SHR/EXCL                      AGE
     -------------------------------------------------------------------------
     Latch 432 FILE SYSTEM:  THE.FILESYS.NAME
      HOLDER(S):
        User10  0044  00880460           SHR                       00:12:08
             IS DOING: NFS ReadCall
             FILE: somefilename                (88,1234)
        User11  0045  00880460           SHR                       00:15:58
             IS DOING: NFS ReadCall
             FILE: somefilename                (88,1234)
         WAITER(S):
        OMVS    000E  008E9B58           EXCL                      00.01.
```

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

SHR/EXCL
If the latch is being hold exclusively or if it is being shared

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 The file the holder is accessing

Sample 7: The ″Other Waiting Threads″ section, new in z/OS V1R8:

```
OTHER WAITING THREADS:
       USER     ASID   TCB         PID           AGE
     ------------------------------------------------
       USER01   0021 00908070     1234         00:12:41
         IS DOING: NFS Readdir / Running
         FILE: nfsdirname                   (33,5432)
         FILE SYSTEM: HOST12.AJAX.DIRECTORY
```

```
  HOLDING: File System Latch #123 SHR
USER03   0041 00908070   786534         00:12:41
  IS DOING: BRLM Wait
  FILE: FileNameIsHere               (22,845)
  FILE SYSTEM: AJAX.DS88.ZFS
```

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

PID PID of the thread

AGE How long the thread has been waiting for the latch

IS DOING
What the thread is doing

FILE The file the thread is accessing

FILE SYSTEM
The file system the thread is accessing

HOLDING
The latch that the thread is holding

Note: This display is very useful in diagnosing OMVS latch contention and hangs. See ″Procedure: Diagnosing and resolving mount latch contention″ in *z/OS MVS Diagnosis: Reference* for more information.

Enhancements to the DISPLAY OMVS,F command

To assist in file system shutdown, the *DISPLAY OMVS,F* console command has been enhanced to filter for file system ownership by a specific system as well as other filtering criteria. The format resulting from the output of these new filtering commands has not changed from the current display. The output will still be BPXO045I display contents (or BPXO042I for valuespecified NOT FOUND). The amount of the display shown will be dependant on the filter option that was used.

The new *D OMVS,F* filtering syntax is as follows. The options can not be combined:

```
D OMVS,F,NAME=xx
```

or

```
D OMVS,F,N=xx
```

(where *xx* is the name of a file system).

Using Wildcards

One wildcard is permitted in the name provided. For example:

```
d omvs,f,name=ZOS18.*.HFS
```

would display file system information for ZOS18.SY1.HFS, ZOS18.SY2.HFS, ZOS18.NLS.HFS and ZOS18.LPP.HFS .

```
d omvs,f,name=ZOS18.L*.HFS
```

would display file system information for ZOS18.LPP.HFS.

Using quotation marks

We found that if you supply the file system name without quotes or with single quotes, the display completed successfully. When we used double quotes, we received a **BPXO042I** message indicating the file system was not found as seen in the examples below:

```
D OMVS,F,N=OMVSSPN.SYSPLEX.ROOT2.ZFS
BPXO045I 14.51.49 DISPLAY OMVS 147
OMVS     0010 ACTIVE             OMVS=(00,TP)
TYPENAME   DEVICE ----------STATUS----------- MODE  MOUNTED    LATCHES
ZFS             1 ACTIVE                      RDWR  09/27/2006  L=14
  NAME=OMVSSPN.SYSPLEX.ROOT2.ZFS                    22.21.02    Q=0
  PATH=/
  AGGREGATE NAME=OMVSSPN.SYSPLEX.ROOT2.ZFS
  OWNER=JB0      AUTOMOVE=Y CLIENT=Y

D OMVS,F,N='OMVSSPN.SYSPLEX.ROOT2.ZFS'
BPXO045I 14.51.40 DISPLAY OMVS 137
OMVS     0010 ACTIVE             OMVS=(00,TP)
TYPENAME   DEVICE ----------STATUS----------- MODE  MOUNTED    LATCHES
ZFS             1 ACTIVE                      RDWR  09/27/2006  L=14
  NAME=OMVSSPN.SYSPLEX.ROOT2.ZFS                    22.21.02    Q=0
  PATH=/
  AGGREGATE NAME=OMVSSPN.SYSPLEX.ROOT2.ZFS
  OWNER=JB0      AUTOMOVE=Y CLIENT=Y

D OMVS,F,N="OMVSSPN.SYSPLEX.ROOT2.ZFS"
BPXO042I 14.51.32 DISPLAY OMVS 134
OMVS     0010 ACTIVE             OMVS=(00,TP)
  "OMVSSPN.SYSPLEX.ROOT2.ZFS" NOT FOUND
```

Displaying file system information by system

To display all the file systems that are owned by system *xx*.

```
D OMVS,F,OWNER=xx
or
D OMVS,F,O=xx
(where xx is a system name)
```

Displaying file system in an ‘exception’ state

To display file systems in an ‘exception’ state (for example; quiesced, unowned, in recovery...)

```
D OMVS,F,EXCEPTION
or
D OMVS,F,E
```

Displaying file systems by type

To display all file systems of the type, *xx*.

```
D OMVS,F,TYPE=xx
D OMVS,F,T=xx
(where xx is a PFS type)
```

For example: D OMVS,F,TYPE=ZFS will display all Physical File System (PFS) zFS type file systems.

Preventing mounts during file system ownership shutdown

With z/OS V1R8, automounted fle systems will not be mounted once the

```
F BPXOINIT,SHUDOWN=FILEOWNER
```

console command has been accepted. No console message will be issued if an automount is attempted. Explicit mounts will still be accepted during and after the file system shutdown. A message will be issued if a mount or ownership change

occurred during the execution. A line in the existing **BPXM048I** console message will be issued that includes the number of file systems that were acquired or mounted during the execution. This is only if the system that is executing the file system shutdown

- becomes the owner of a newly mounted file system, or
- acquires a file system due to a move operation (only for 'f bpxoinit,shutdown=filesys') before the completion of the shutdown.

An example of the message that may occur if 2 file systems were mounted during the shutdown is:

```
BPXM048I  BPXOINIT FILESYSTEM SHUTDOWN INCOMPLETE.
3 FILESYSTEM(S) ARE STILL OWNED BY THIS SYSTEM.
2 FILESYSTEM(S) WERE MOUNTED DURING THE SHUTDOWN PROCESS.
```

Distributed BRLM (Byte Range Lock Manager) with Lock Recovery Support

When all systems in a sysplex are at the z/OS V1R6 or later release level, distributed BRLM (where lock manager is initialized on every system in the sysplex) is the default instead of a single central BRLM. This allows a file system to move while byte range locks are held for files in that file system. However, if there was a system failure, the corresponding locking history was lost. Prior to V1R6, you may receive an *enomove* return code which would prevent the filesystem move.

When all systems in a sysplex are at the z/OS V1R8 or later release level, distributed BRLM (Byte Range Lock Mananger) has lock recovery support. Lock recovery support backs up each lock in the application's system when the actual lock is stored in another system. Therefore, when a system fails in a shared file system sysplex and a file system changes owners, the corresponding locking history changes BRLM servers. This allows executing applications that have identified BRLM locks for files in these file systems to continue. Processes, that have identified BRLM locks for files or that have made use of byte range locks in file systems that are unmounted or not moved, may receive a signal and possibly terminate.

Locks are lost if the file system

- can not be recovered
- unmounts
- was identified as automove(no), or
- a lock for a file in the file system was not successfully backed up.

When Locks are lost, access by existing applications is prevented to files that those existing applications have locked. When byte range locks are lost, processes that have used byte range locking will receive the following:

- Processes accessing open files for which byte range locks are held will receive an I/O error. To continue file access, the file must be closed and reopened.
- Any process that has made use of byte range locking is issued a signal. The default signal is a SIGTERM and an SEC6 with reason code 0D258038, which will terminate the process.

  Note: BPX1PCT (the physical file system control callable service) can be used to specify a different signal to notify the process that BRLM has failed. This allows the user or application to catch or ignore the signal and react in a user defined manner.

The BRLM recovery is for system failures. However, when we used the soft file system shutdown:

```
F BPXOINIT,SHUTDOWN=FILESYS|FILEOWNER
```

or the OMVS shutdown

```
F OMVS,SHUTDOWN
```

the processes using byte range locks on the file systems that have the attribute of *AUTOMOVE(UNMOUNT)* or *AUTOMOVE(NO)* were not signaled.

When we used the STOPPFS option on the modify OMVS command to stop the Physical File System, zFS:

```
F OMVS,STOPPFS=ZFS
```

there were signals (SIGTERM by default) to processes using byte range locks on those zFS file systems that have the attribute of *AUTOMOVE(UNMOUNT)* or *AUTOMOVE(NO)*. These behaviors may not be the same in your environment and may change in the future.

Using the _UNIX03 z/OS UNIX Shell environment variable

The UNIX 03 Product Standard is the mark for systems conforming to Version 3 of the Single UNIX Specification. It is a significantly enhanced version of the UNIX 98 Product. For more information on this standard please see The Open Group's website:

```
http://www.unix.org
```

In z/OS V1R8, some UNIX System Services utilities implemented support for the UNIX 03 specification. _UNIX03 is an environment variable, when set to YES, the utilities that have implemented support for the UNIX 03 specification will conform to it. Please note that this variable is only needed when the syntax or behavior of the new implementation conforming to UNIX 03 conflicts with the existing implementation.

The following are two utilities that support the UNIX 03 specification:

- *cp*
- *mv*

cp utility

In z/OS V1R8, the OMVS shell utility *cp* has 3 ('-H', '-L', '-P') new options to handle symlink processing during a recursive copy ('-R' or '-r' option flags). However, there was already an existing '-P' option for the *cp* utility. It was used for specifying the parameters needed to create a sequential data set. To resolve this conflict, the _UNIX03 environment variable can be used by *cp* to decide whether to do '-P' for symbolic links handling or '-P' for sequential data set creation. If _UNIX03 is set to YES, *cp* will process '-P' for symbolic links handling. If it is set to anything else, *cp* will process '-P' for creating a sequential data set. Another new option for the *cp* utility is '-W'. It works the same way as today's '-P' option. It is provided so that users can create sequential data sets while _UNIX03 environment variable is set to YES as well.

Here are what the 3 new cp options, mentioned above, do:

-H When the **-H** option is specified, *cp* follows symbolic links specified as a source

operand on the command line. Following a symbolic link means that an exact copy of the file that is linked will be created rather than a copy of the symbolic link itself.

-L When the **-L** option is specified, *cp* behaves the same way it does when **-H** is specified. However, it also follows the symbolic links that are found during tree traversal.

-P When the **-P** option is specified, *cp* does not follow any symbolic links.

Another new option for the cp utility is:

-W

-W works the same way as today's '-P' option. It is provided so that users can create sequential data sets while _UNIX03 environment variable is set to YES as well.

Examples of UNIX System Services utilities that implement support for the UNIX 03 specification

Set the _UNIX03 environment variable to YES.

```
export _UNIX03=YES
```

Recursively copy directory dir1 to dir2. Use the -P option so that no symbolic links are followed.

```
cp –r –P dir1 dir2
```

Set the _UNIX03 environment variable to anything but YES.

```
export _UNIX03=NO
```

Next, use the -P option to specify the parameters needed to create a sequential data set. The command below will copy file1 into a new sequential data set named uss.test0.

```
cp -P "RECFM=U,space=(5,1)" file1 "//'uss.test0'"
```

Leave the _UNIX03 option set to anything but YES. Use the '-W' option to create a sequential data set called uss.test1.

```
cp -W "seqparms='RECFM=U,space=(5,1)'" file1 "//'uss.test1'"
```

Set the _UNIX03 environment variable to YES. Use the '-W' option to create a sequential data set called uss.test2. 'cp -W' behaves the same no matter what _UNIX03 is set to.

```
cp -W "seqparms='RECFM=U,space=(5,1)'" file1 "//'uss.test2'"
```

mv utility

In z/OS V1R8, the OMVS shell utility *mv* has 1 new option as well ('-W'). It serves the same exact purpose as the existing *mv* option '-P'. It is implemented purely for consistency purposes between the *cp* and *mv* utilities. Since the *mv* utility does not have any option conflict issues, the _UNIX03 environment variable does not need to be set to YES for the *mv* to process the '-W' option.

Implementing /etc/inittab in z/OS UNIX

Starting with z/OS V1R8, you can use the */etc/inittab* file to start daemons, system processes and execute shell scripts or commands at z/OS UNIX initialization. This file is processed by */etc/init*, only once, during z/OS UNIX initialization. However, */etc/inittab* allows you to assign, the command entries you listed in this file, an attribute that will allow them to restart automatically when they end.

/etc/inittab file is optional to use and is not configured by default. z/OS V1R8 comes with a sample */etc/inittab* file. In order to start using */etc/inittab* all you need to do is simply copy that file into your */etc* directory and then customize it according to your needs.

The sample */etc/inittab* looks like this:

```
etcrc::wait:/etc/rc > /dev/console 2>&1
inetd::respfrk:/usr/sbin/inetd /etc/inetd.conf
msgend::once:/bin/echo Done processing /etc/inittab > /dev/console
:end of file
```

A *':'* is used as a delimiter as well as a comment character, so the last line is just a comment. The format followed for the rest of the file is:

```
Identifier:RunLevel:Action:Command
```

The first line in the above sample identifies the process it would like to be executed as *etcrc*. The action that it would like us to take is wait (more on that below). Finally, the actual command it would like us to run is *"/etc/rc > /dev/console 2>&1"*.

Please notice the two *':'* in a row after *etcrc*. That means that we are not assigning a RunLevel to this entry. That is because the RunLevel entry field is not supported in z/OS UNIX. It is in the *inittab* entry to be compatible with other UNIX implementations.

Here is a summary of the different values you can assign to the action entry field. For details, please see *z/OS UNIX System Services Planning*, GA22-7800.

once
: Starts the process, continues scanning the */etc/inittab* file and does not restart the process when it ends.

respawn
: Starts the process, continues scanning the */etc/inittab* file and restarts the process when it ends.

respfrk
: Starts the process, continues scanning the */etc/inittab* file. If the process issues a fork, the respawn attribute is transferred to the forked child process and the original process is **respawn**ed when the child process ends. This transfer takes place only for the first fork. If the process does not fork at all, then **respfrk** will behave the same way **respawn** does. Also note that this option can not be found on any other UNIX systems.

wait
: Starts the process, waits for it to end before continuing to scan the **/etc/inittab** file. The process is not restarted when it ends.

BPX_INITTAB_RESPAWN environment variable

z/OS V1R8 introduces a new environment variable, _BPX_INITTAB_RESPAWN, which enables you to start processes with the respawn attribute even after a system had already been IPLed. You can set this environment variable to YES or NO from the z/OS UNIX shell.

YES
You can set it to YES so that any processes that are spawned (non-local), in this shell, will be run with the respawn attribute and will behave like they were started from */etc/inittab* with the respawn attribute.

NO
You can set it to NO so that future processes that are spawned (non-local), in this shell, will not be started with the respawn attribute. This has no effect on the processes that are already up and running in the system.

Note: The BPX_INITTAB_RESPAWN environment variable will be ignored when the process is also SHAREAS=YES, since the two are mutually exclusive!

For more information on the _BPX_INITTAB_RESPAWN environment variable, please take a look at the ″_BPXK Environment Variables″ section of *z/OS UNIX System Services Planning*, GA22-7800.

Identifying whether a process has been started with the respawn attribute

z/OS V1R8 introduces a couple of ways to tell if a process, that is up and running, is assigned the respawn attribute.

1. **Using the *ps* UNIX shell utility**

   A new format specification, *attr*, is added to the *ps* command. Using this format specification through the -o option, users can request to display process attributes. The attributes that they could have can be listed as:

   - respawnable process (R)
   - permanent process (P) or
   - shutdown blocking process (B).

   For example: A subset of the ″*ps -ef -o attr,comm*″ command could look something like this:

```
ATTR COMMAND
P    IRRSSM00
R    /usr/sbin/cron
B    HZSTKSCH
-     CSQXJST
```

2. **Using the display OMVS MVS console command**

   A new indicator is added to the display OMVS MVS system command's output in order to indicate whether a process has been started with the respawn attribute. It is added to the STATE column of the ″D OMVS,ASID″ and ″D OMVS,PID″ displays as the 5th indicator.

   For example: A subset of the ″D OMVS,ASID=ALL″ command output looks like this:

```
OMVS     0010 ACTIVE             OMVS=(00,TP)

USER     JOBNAME  ASID        PID       PPID STATE   START     CT_SECS

SETUP    BPXOINIT 0047          1          0 MR---- 21.42.35    38.116
```

```
  LATCHWAITPID=         0 CMD=BPXPINPR
  SERVER=Init Process                    AF=    0 MF=00000 TYPE=FILE
OZ2      TEST1    0118      65678   33620109 1SI--R 08.45.16      .023
   LATCHWAITPID=         0 CMD=sleep 1h
```

Take a look at the value under the STATE column of the job named TEST1. It says, 1SI--R. A 5th indicator value of R under the STATE column means that the job TEST1 was started with the respawn attribute.

Stopping a process that was started, by /etc/inittab, with the respawn attribute

One question that might come up is ″What if I started a process throguh */etc/inittab* with the respawn attribute but I want to stop it and I don't want it to restart?″

If a respawnable process ends and then ends again after being restarted within 15 minutes of its first ending, then message BPXI083D is displayed. The message identifies the process entry and asks whether to retry or ignore the error.

For example the message could look something like this:

```
9813 R JF0      *9813 BPXI083D ReSpawnable process TEST1 ended. Reply R to restart the process.
Anything else to end the process.
```

If you would like to keep it down simply reply with anything but an R.

Implementing /etc/inittab in the zPET environment

Here is the way we implemented /etc/inittab in the zPET environment:

```
etcrc::wait:/etc/rc > /dev/console 2>&1
inetd::respfrk:/usr/sbin/inetd /etc/inetd.conf
cron::respfrk:/usr/sbin/cron
msgend::once:/bin/echo Done processing /etc/inittab > /dev/console
:end of file
```

We had to comment out *inetd* and *cron* statements in our */etc/rc* before implementing the above.

Initially you can simply copy the */samples/inittab* file into your */etc* directory, comment out the *inetd* entry in your */etc/rc* file and then implement */etc/inittab*!

If you would like to, you could move the start of all your daemons from your */etc/rc* file to your */etc/inittab* file. Then assign them the appropriate actions so that you can take advantage of the functionality that */etc/inittab* provides.

Finally, again if you are interested, you could go all out and completely replace your */etc/rc* file with the */etc/inittab* file. The capability is there. However, some installations may still want to keep all their commands and scripts in */etc/rc* and keep */etc/inittab* focused on the daemons. That is perfectly acceptable to do.

Moving to 64-bit Java and JDK 5

We have begun moving our Java applications to using the 64-bit JDKs and JDK 5.

Overall, we have found our applications to be upward compatible and able to run on the 64-bit and/or JDK 5 versions with no changes needed. You should still plan on testing your applications against the newer levels and review them for any possible migration actions that may be needed.

See the Sun website for additional information on incompatibility between the levels:

http://java.sun.com/javase/technologies/compatibility.jsp

While not there yet, we anticipate many of the products that require Java will soon support the latest levels. Check your product's support pages to determine what level of Java it supports. In the meantime, we got started with our ″stand-alone″ Java applications. Many of these are run by the users from Unix System Services shells or utilities (such as BPXBATCH).

Juggling Java versions

Juggling Java versions, levels, and service refresh levels and mixing and matching them with application and product requirements has always been a challenge. What used to be complicated enough has increased due to the new version and flavors of Java available. Currently, there are 4 JDK's of interest available for z/OS:

1. JDK 1.4.2 31-bit
2. JDK 1.4.2 64-bit
3. JDK 5 31-bit
4. JDK 5 64-bit

The good news is that there is a high degree of compatibility between the levels. Our stand-alone applications that were running on the JDK 1.4.2 31-bit level continued to run without changes on all of the other levels (with the exception of the MEMLIMIT change possibly required for newer JDK's. See "Increasing MEMLIMIT" and "Changing system-wide default for MEMLIMIT" for more on that).

While you can still move between JDK 1.4.2 and JDK 5 and continue to use a 31-bit version, you should generally be able to take advantage of the 64-bit code by simply changing the Java level used by the application.

Increasing MEMLIMIT

As pointed out in the Readme files that come with the 64-bit JDK 1.4.2 and both of the JDK V5 versions (31 and 64-bit), the z/OS MEMLIMIT parameter should be set to 256MB or greater. These versions of Java use (and require) ″**above the bar**″ memory, or memory in the 2G and above address range.

There are a number of ways of setting the MEMLIMIT for an address space, depending upon the environment the application is run in and the parameters used to start. For example, for our Java applications using BPXBATCH, we added ″MEMLIMIT=256M″ to the JCL.

See Chapter 4, Using the 64-bit Address Space, in the *z/OS MVS Programming: Extended Addressability Guide*, SA22-7614 for full details on using storage above the 2GB address range. Especially helpful is a diagram that shows how the MEMLIMIT is determined for a process (Figure 4-2. How the System Chooses which MEMLIMIT Applies).

Changing system-wide default for MEMLIMIT

The system default for MEMLIMIT, if not specified in the SMF parameter, is 0M, which means that no storage ″**above the bar**″ (above 2G address) is available to a process. For an application to run that requires storage above 2G, it must override this system default by one of a number of way, as referenced above.

We found that for many of our Java users who were using Unix System Services (OMVS, Telnet, and others), generally inherited the system-wide default of 0M. This caused the newer versions to fail when Java is starting, as it tries to load its code

above the 2G ″bar″. Initially, we set the MEMLIMIT for the user's working with newer JDKs in a variety of ways (see above).

After a bit, this became cumbersome dealing with individual cases and was hampering our use and our move to 64-bit.

We decided to change the system-wide default to be at least 256M. There currently is no recommended value for this setting, except that it is recommended NOT to use NOLIMIT. With NOLIMIT, a run-away process could exhaust system resources.

While the 256MB setting is a minimum required for Java, we chose a value of 512M. This allows for some additional room to grow. Now the general user community could begin to use the newer Java versions without hitting this initial limit. Only users or applications that require more than 512M above the 2G address would have to take alternative measures.

To implement this, the following update was made to the SYS1.PARMLIB(SMFPRM00) member where we added:

```
MEMLIMIT(512M)
```

Reference Information

The following reference information was used in moving to 64-bit Java and JDK 5:

- Java on z/OS website:

  `http://www.ibm.com/servers/eserver/zseries/software/java/`
- z/OS Internet Library:

  `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`
- *z/OS MVS Initialization and Tuning Reference*, SA22-7592
- *z/OS MVS JCL Reference*, SA22-7597
- *z/OS MVS Programming: Extended Addressability Guide*, SA22-7614

BPXBATCH enhancements in z/OS V1R8

BPXBATCH, which is a Unix System Services utility for running Unix System Services shell scripts and commands in a batch environment, has been enhanced in z/OS V1R8 for usability.

- Prior to this release, STDOUT and STDERR DDs had to use z/OS UNIX files. Now in z/OS V1R8, these DDs can be represented by SYSOUT, PDSEs, PDS and Sequential datasets.
- Prior to z/OS V1R8, STDENV would only permit SYSIN, PDS or Sequential datasets. In z/OS V1R8 PDSEs will be permitted.

We ran many BPXBATCH jobs using different representatives for STDOUT and STDERR and found them to work as delivered in z/OS V1R8.

New BPXBATCH messages

Following are new error messages that will be issued if you allocate the PDSEs, PDSs or sequential datasets incorrectly.

BPXM012I

Error message BPXM012I will be issued stating that there is an OPEN failure for the dataset if the PDSEs, PDSs or Sequential datasets do not have a non-zero record length (LRECL) and a defined record format (RECFM).

BBPXM080I

Error message BPXM080I is issued when the record length is not large enough to hold the line of output. BPXM080I states that the data was truncated. This can happen for both fixed and variable blocked datasets. For variable blocked data sets, the first four bytes of each record, record segment, or block make up a descriptor word containing control information. You must allow for these additional 4 bytes in the specified LREC if you intend to avoid truncation of the output to the STDOUT and STDERR DDs. If two members of the same partitioned data set are to be used for STDOUT and STDERR output, then using a PDSE is required. Using a PDS will lead to either a **213** abend and, if running within a batch job, the jobstep ending abnormally, or the output not appearing in the members as expected.

Currently, if a MVS data set is specified on STDOUT or STDERR, BPXBATCH ignores the data set and defaults to */dev/null*. To remain as compatible as possible with this behavior, the new support in z/OS V1R8 will do the same defaulting if the MVS data set type is not supported (for example, DD Dummy, Terminal, SYSIN, and others), or if the MVS data set cannot be opened by BPXBATCH. BPXM081I will be displayed indicating when this default behavior is being taken by BPXBATCH.

BPXMTEXT support for zFS reason codes

A new zFS function being delivered in z/OS V1R8 is the ability to easily determine what a given zFS reason code error means. Currently, zFS reason codes must be looked up in *z/OS Distributed File Service Messages and Codes*, SC24-5917 to determine what the reason code means. Now, the description text and action text associated with a zFS reason code can be displayed using the *bpxmtext* command. The resulting text will match what is in the publication. This function works for reason codes that are in the form of EF*nnxxxx* or that is of the form 6*nnn*. The following are two examples of using the command.

1. **bpxmtext EF17606E:** We executed the following command from ISPF.6:

   ```
   bpxmtext EF17606E
   ```

   The results were:

   ```
   zFS Thu Aug 10 15:41:00 EDT 2006
   Description: Incorrect, undefined or inconsistent arguments.
   Action: Verify that the new size on grow is greater than the current size. The
   zfsadm aggrinfo command displays the current size in 1K blocks. Divide this
   amount by 8 to get the current size in 8K blocks. If this is not the case for
   the reason code, contact the service representative
   ```

2. **bpxmtext EF17624E:** We executed the following command from ISPF.6:

   ```
   bpxmtext EF17624E
   ```

   The results were:

   ```
   zFS Thu Aug 10 15:41:00 EDT 2006
   Description: Aggregate not found.
   Action: The aggregate specified cannot be found. Correct the aggregate name
   and try again.
   ```

z/OS zFS enhancements in z/OS V1R8

z/OS zFS made several enhancements in z/OS V1R8. In this section, we cover the following topics:

- "Deny mounting of a zFS file system contained in a multi-file system aggregate when running in sysplex mode on z/OS V1R8" on page 126

• "Stop zFS (modify omvs,stoppfs=zfs)" on page 127

Deny mounting of a zFS file system contained in a multi-file system aggregate when running in sysplex mode on z/OS V1R8

zFS multi-file system aggregates are not fully supported and are restricted in a sysplex environment (shared-file system). With z/OS V1R8, zFS enforces this by not allowing a mount of a file system that is contained in a multi-file system aggregate, and when zFS is running in z/OS UNIX Shared file system mode. You must migrate your zFS file systems in a multi-file system aggregate to zFS compatibility mode aggregates which have a single file system per aggregate. You must copy the data using a z/OS V1R7 or earlier system or by using a non-shared file system environment. For more information, see *z/OS Distributed File Service zSeries File System Administration*, SC24-5989 for migrating from one file system to another.

The following message is issued during

- explicit attach (using *zfsadm* attach or IOEZADM)
- processing define_aggr during zFS start up, and
- format processing when formatting a multi-file system aggregate (zfsadm format or IOEAGFMT with *-compat* omitted):

```
IOEZ00552I Multi-file system aggregates are restricted and support will be removed; plan to migrate.
```

When a mount of a file system in a multi-file system aggregate is attempted, a mount error occurs with a return code of **00000079** and reason code of **EF096800**. This relays to the user that the mount of the file system contained in multi-file system aggregate is not allowed.

Note that the following message may appear, which indicates this file system will be non-mountable in a sysplex:

```
IOEZ00316I The file system to be mounted, filesystem, is part of a multi-file system aggregate.
```

Note: Make sure you take steps to move the file system to a compat aggregate.

Following are examples of what happens when you attempt to mount a multi-file system aggregate file system on z/OS V1R8:

- Using the shell mount command:

```
$ mount -f OMVSSPN.MULTI.FS1.ZFS /multizfs
FOMF0504I mount error: 79 EF096800
EINVAL: The parameter is incorrect
Description: Mount for file system contain in multi-file system aggregate is not allowed
```

  Note that the return code 79 indicates: 121(0079x) **EINVAL The parameter is incorrect**.

- Using the TSO mount command:

```
MOUNT FILESYSTEM('OMVSSPN.MULTI.FS1.ZFS') TYPE(ZFS)
  MODE(READ) MOUNTPOINT('/multizfs') NOAUTOMOVE

 RETURN CODE 00000079, REASON CODE EF096800.  THE MOUNT FAILED FOR FILE SYSTEM OMVSSPN.MULTI.FS1.ZFS.
```

- Using the shell *bpxmtext* command to display the reason code shows:

```
$ bpxmtext EF096800
zFS Tue Jun 20 09:15:22 EDT 2006
Description: Mount for file system contain in multi-file system aggregate is not allowed

Action: Using a release of z/OS prior to z/OS V1R8, attach the aggregate,
mount the file system and copy the file system data to a compatibility mode aggregate.
```

Stop zFS (modify omvs,stoppfs=zfs)

With V1R8, a new operator modify command is introduced to stop zFS. You can no longer use the *stop zfs* (P ZFS) command to terminate the zFS physical file system. If the *stop zfs* command is entered, it results in a message informing you to use the *stoppfs* option of the modify omvs command. The zFS Physical File System must be running outside of the Kernel as a colony address space. For more information, see *z/OS Distributed File Service zSeries File System Administration*, SC24-5989.

The console command, *"modify omvs,StopPFS=pfsname"* replaces *Stop zfs* as the console command to terminate zFS.

The *modify omvs,StopPFS=pfsname* will generate a prompt to the operator:

```
STOP OF PFS pfsname REQUESTED, REPLY 'Y' TO PROCEED. ANY OTHER REPLY WILL CANCEL THIS STOP.
```

Some examples of the commands and their messages are:

```
P ZFS
IOEZ00523I zFS no longer supports the stop command. Please issue f omvs,stoppfs=zfs

 F OMVS,STOPPFS=ZFS
*0251 BPXI078D STOP OF ZFS REQUESTED. REPLY 'Y' TO PROCEED. ANY OTHER REPLY WILL CANCEL THIS STOP.
R 251,Y
IEE600I REPLY TO 0251 IS;Y
```

When zFS terminates, the **BPXF032D** message should appear to allow you to restart or ignore the zFS filesystype, for example:

```
*0252 BPXF032D FILESYSTYPE ZFS TERMINATED.  REPLY 'R' WHEN READY TO RESTART. REPLY 'I' TO IGNORE.
```

In a non-sysplex environment all the zFS file systems will be unmounted and zFS will be terminated. In a sysplex environment an attempt will be made to move any of the zFS file systems that are owned by that system to another system so that the termination of zFS is not disruptive. File systems that were mounted with *AUTOMOVE(UNMOUNT)* will be unmounted if sysplex-unaware for the mount mode or if the PFS is non-remote and fully sysplex-aware.

In the LFS (z/OS UNIX Logical File System), for all zFS file systems mounted and owned by this system, those designated as *AUTOMOVE(NO)* or *AUTOMOVE(UNMOUNT)* that are sysplex-unaware for the mount mode will be globally unmounted. If the zFS PFS is non-remote and is sysplex-aware for both modes (fully sysplex-aware), then those file systems designated as *AUTOMOVE(UNMOUNT)* will be globally unmounted. The rest of the file systems will be moved to another system (if owned by the non-remote zFS PFS) and then converted to function shipping as an LFS client to maintain availability.

Chapter 10. 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 migrated our WebSphere for z/OS queue managers from V5.3.1 to V6.0. 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. As we continue our testing with WebSphere MQ V6.0 we will start to describe our experiences here as well as in future releases of the test report.

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 or higher. 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 three coupling facilities(CF1, CF2 and CF3) as defined in the preflist in the structure definitions. (See “Coupling facility details” on page 12 for details about our coupling facilities.)

The following is the structure definition for our CSQ_ADMIN structure:

```
STRUCTURE NAME(MQGPCSQ_ADMIN)
          INITSIZE(18668)
          MINSIZE(18668)
          DUPLEX(ENABLED)
          SIZE(20480)
          ALLOWAUTOALT(YES)
          PREFLIST(CF3,CF2,CF1)
          REBUILDPERCENT(1)
          FULLTHRESHOLD(85)
```

We also have the following five 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 applications
- BOOK – for our BookStore workload (uses DB2, WMQ and Websphere Application Server)

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,CF1)
          REBUILDPERCENT(1)
          FULLTHRESHOLD(85)
```

The other four 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, BOOK and MSGQ1 structures are recoverable and backed up daily.

Recovery behavior with queue managers 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 messages could be:

  ```
  CSQ5003A !MQJA0 CSQ5CONN Connection to DB2 using DBWG pending, no active DB2
  CSQ5026E !MQJA0 CSQ5CONN Unable to access DB2, RRS is not available
  ```

  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 DBW3
  ```
- 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 up to 18668 to accommodate the increase in usage.

Running WebSphere MQ 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. 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 192.168.49.30 192.168.49.32 192.168.49.33 192.168.49.38
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.PDL.POK.IBM.COM, which maps to the 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(DQMSSL.CSQ9.TO.MQGP) +
REPLACE +
CHLTYPE(SDR) +
XMITQ(DQMMQGP.QSG.XMITQ) +
TRPTYPE(TCP) +
DISCINT(10) +
```

```
CONNAME('MQGP.PDL.POK.IBM.COM(1415)') +
SSLCIPH(TRIPLE_DES_SHA_US) +
DESCR('DQM SDR CHANNEL TO SHARED RCVR CHANNEL ON MQGP')

DEFINE CHANNEL(DQMSSL.CSQ9.TO.MQGP) +
REPLACE +
CHLTYPE(RCVR) +
QSGDISP(GROUP) +
TRPTYPE(TCP) +
SSLCAUTH(REQUIRED) +
SSLCIPH(TRIPLE_DES_SHA_US) +
DESCR('SHARED RCVR CHANNEL FROM J90 FOR DQM')
```

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(DQMCSQ9.QSG.XMITQ) +
REPLACE +
STGCLASS(DQMSTG) +
DESCR('SHARED XMITQ QUEUE FOR DQM TO J90') +
QSGDISP(SHARED) +
MAXDEPTH(2000) +
TRIGGER +
TRIGDATA(DQMSSL.MQGP.TO.CSQ9) +
INITQ(SYSTEM.CHANNEL.INITQ) +
USAGE(XMITQ) CFSTRUCT(MSGQ1)
```

Example: The following are our definitions for the local sender channel and the remote receiver channel:

```
DEFINE CHANNEL(DQMSSL.MQGP.TO.CSQ9) +
REPLACE +
CHLTYPE(SDR) +
XMITQ(DQMCSQ9.QSG.XMITQ) +
QSGDISP(GROUP) +
TRPTYPE(TCP) +
DISCINT(15) +
CONNAME(J90EIP.PDL.POK.IBM.COM) +
SSLCIPH(TRIPLE_DES_SHA_US) +
DESCR('SHARED SDR CHANNEL TO J90 FOR DQM')

DEFINE CHANNEL(DQMSSL.MQGP.TO.CSQ9) +
REPLACE +
CHLTYPE(RCVR) +
TRPTYPE(TCP) +
SSLCAUTH(REQUIRED) +
SSLCIPH(TRIPLE_DES_SHA_US) +
DESCR('DQM RCVR CHANNEL FROM SHARED SDR CHANNEL ON MQGP')
```

Enabling WebSphere MQ Security

We recently went through the task of enabling MQ security for the z/OS queue managers in our zPET environment. WebSphere MQ provides an interface to an external security manager which, in our case, is Resource Access Control Facility (RACF). When we decided to enable security for our queue managers, we took a step back to determine the best approach for our environment. Our simple approach to controlling security was to use queue-sharing group level of security for our queue managers that were members of a queue-sharing group and queue manager level of security for the rest of the queue managers in our environment which are not members of queue-sharing groups.

Referencing the 'System Setup Guide' section ″Using RACF classes and profiles″ we first verified that the WebSphere MQ classes were activated in RACF. As in most customer environments we then used our 'test plex' as our starting point for enabling MQ security. Our 'test plex' consists of 3 z/OS images each running a queue manager at V6.0. These 3 queue managers are all members of the queue-sharing group MQGT. Since all 3 queue managers are members of the same queue-sharing group we decided to use queue-sharing group level of security. We started by defining a basic set of profiles to each of the WebSphere MQ classes.

Reference Material

We found the following reference material useful when working with WebSphere MQ Security:

- WebSphere MQ for z/OS Security (Technical Conference) which is a good overview located at:

  `http://www.gse.org.uk/wg/racf/docs/apr2005/GSE-%20WebSphere%20MQ%20zOS%20Security.pdf`
- **WebSphere Message Broker (WMB):** which outlines the necessary authority required by the broker. Search for ″Authorization required″ and then select ″Summary of required access (z/OS)″ at:

  `http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/index.jsp`
- **WebSphere MQ Explorer:** which outlines the necessary authority required by the MQ Explorer. Search for ″Authorization to use WebSphere MQ Explorer″ at:

  `http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/index.jsp`
- **SDSF:** The following links document the necessary authority required by SDSF:
  - Communications:

    `http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/ISF4CS50/3.7?SHELF=ISF4BK50&DT=20050707140821`
  - **WebSphere:**

    `http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/ISF4CS50/7.29?SHELF=ISF4BK50&DT=20050707140821`
  - **SDSF Customization Wizard:** provides assistance in defining security for SDSF's use of MQ:

    `http://www-03.ibm.com/servers/eserver/zseries/zos/wizards/sdsf/sdsfv1r1/`

Once we had the basic profiles defined we started to enable security for each queue manager one at a time, resolving problems as they arose. We used RACF groups to grant authorities instead of individual userids which should make maintaining this security much easier. After enabling security for our 'test plex' we moved on to our 'production plex'. Our 'production plex' consists of 10 z/OS images each running a queue manager at V6.0. Of these 10 queue managers 4 of them are members of the same queue-sharing group MQGP. For the 4 queue managers that are members of a queue-sharing group we implemented security using the queue-sharing group level of authority. For the other queue managers we implemented the queue manager level of security.

Problems encountered

Following are some of the problems we encountered:

1. **WebSphere V6 Explorer:**
   a. After enabling security for our z/OS queue managers our connection to these queue managers using the WebSphere MQ Explorer were rejected with the following error:

```
ICH408I USER(dodaro ) GROUP(        ) NAME(???                )  932
   LOGON/JOB INITIATION - USER AT TERMINAL          NOT RACF-DEFINED
 IRR012I  VERIFICATION FAILED. USER PROFILE NOT FOUND.
```

   The userid was being sent to the host in 'lower case' and RACF was rejecting it. We installed fix pack 6.0.1.1 (U200247) for WebSphere MQ V6.0 to resolve this problem.

   b. With security enabled for our z/OS queue managers our connection was rejected with the following error:

```
ICH408I USER(DODARO) GROUP(SYS1    ) NAME(####################) 015
  MQGT.AMQ.BF0F023EF3019DB9 CL(MQQUEUE )
  PROFILE NOT FOUND - REQUIRED FOR AUTHORITY CHECKING
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
```

   The queue being created was using the incorrect prefix 'AMQ.**' instead of 'AMQ.MQEXPLORER.**'. APAR IC50201 will resolve this issue.

2. **Mixed case' queue names:** After enabling security in our zPET environment we ran into a situation trying to access one of our queues. WebSphere MQ supports 'mixed case' for their queue names. We had a queue named 'Trade3BrokerTestQueue'. When we attempted to access this queue we received the following racf error:

```
ICH408I USER(WAS5SSR3) GROUP(WASSRGP ) NAME(WAS 5 APPSVR SR 3
  MQGT.Trade3BrokerTestQueue CL(MQQUEUE )
  PROFILE NOT FOUND - REQUIRED FOR AUTHORITY CHECKING
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
```

   RACF currently does not allow defining 'mixed case' profiles for the MQ classes. To get around this situation we created a profile named 'MQGT.T*' and granted the necessary authority to this profile. Until RACF supports 'mixed case' profiles we would suggest that if you use 'mixed case' that your queue name is prefixed with enough characters in 'upper case' (for example TRADE3BrokerTestQueue} which will allow you to properly protect your queues.

3. **WebSphere Message Broker and WebSphere Application Server:** After enabling security for our z/OS queue managers we experienced problems when connecting to our queue managers from these applications when the userid being sent to the host was in 'lower or mixed case' and subsequently was rejected by RACF. This was the case with the WMB toolkit running on Windows and connecting to z/OS config mgr.. Here we changed the userid on Windows to be in 'uppercase'. This was also the case for WebSphere Application Server when the JMS resource was defined using a 'lowercase' userid causing the listener not to start. Again, here we were able to get around this problem by changing the JMS resource definition in WebSphere Application Server to use an 'uppercase' userid.

```
MQJMS2013: invalid security authentication supplied for MQQueueManager   at startup.

CSQ8MSTR has: ICH408I USER(setup   ) GROUP(        ) NAME(???            )
LOGON/JOB INITIATION - USER AT TERMINAL          NOT RACF-DEFINED
```

Migrating to Websphere Message Broker Version 6

Before migrating to Websphere Message Broker (WMB) V6 our WBIMB configuration consisted of three brokers at the WebSphere Business Integration Message Broker V5 level on one sysplex and two brokers at the same level on another sysplex. We used the WBIMB V5 toolkit on Windows connecting to a Configuration Manager which was also on Windows. We migrated all of our brokers to WMB V6, created a new broker, and added a z/OS Configuration Manager to one sysplex while the other one is still using the old Configuration Manager in our Windows machine. The option to have a z/OS configuration manager is new with WMB V6.

Changes from WBIMB V5 to WMB V6

The following are some of the changes we had to make from WBIMB V5 to WMB V6.

Directory structure changes

WMB V6 added a new HOME directory separate from the COMP directory used in WBIMB V5. Our new directory structure looks as follows:

```
/wmb60/basecode    contains the product code
/wmb60/COMP        contains a directory for each broker or configmgr
/wmb60/HOME        contains a directory for each broker and config mgr which has files bipprof and ENVFILE
```

Each of the above directories is mounted off of a separate ZFS filesystem.

DB2 DSNAOINI file changes

The WMB V6 started task JCL uses the *dsnaoini* from a dataset instead of using the one in the broker directory as WBIMB V5 did.

/wmb60/HOME/CSQ2BRK/bipprof has a statement:

```
export DSNAOINI=//\'WMB.CSQ2BRK.DSNAOINI\(BIPDSNAO\)\'
```

The broker ENVFILE contains the statement:

```
DSNAOINI=//'WMB.CSQ2BRK.DSNAOINI(BIPDSNAO)'
```

This points to the z/OS dataset member to get the values it needs.

We followed the migration instructions in the WMB V6 Information Center

```
http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/index.jsp
```

See the section titled ″Migrating from Version 5.0 products″ for the WebSphere Message broker product. We migrated a broker first, then the toolkit, and did the configuration manager last.

XML changes

The new WMB V6 broker requires the XML Toolkit for z/OS, Pgm 5655-J51. This was installed and referenced in the broker bipprof file XMLTOOLKIT=/ixm/ixm/IBM/xml4c-5_5 as well as in the ENVFILE.

Broker migration

Some of the things to watch out for with the Broker migration are:

- Be sure to never edit the files in the broker registry directory. If changes need to be made use **printf** ″**changed value**″ **> filename** . Editing can often add CR or LF characters which the broker does not handle well.

- When migrating, the component directory is the previous version's component directory. The HOME directory is new for WMB V6.

As a pre-migration task we backed up our broker databases and toolkit workspace data. We also backed up the component directories for the brokers. Then we followed the steps outlined in the section ″Migrating from WebSphere Business Integration Message Broker Version 5.0 to WebSphere Message Broker Version 6.0″ sub-topic ″Migrating a Version 5.0 broker to Version 6.0 on z/OS″ found at:

`http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/index.jsp`

Note: The Unix System Services environment variables of the userid running the migration jobs will be copied to the broker ENVFILE in the HOME directory. Be careful and review the ENVFILE to be sure you don't have variables set that you don't want for the broker.

We edited the /wmb60/HOME/CSQABRK/*ENVFILE* to remove all entries it added from */u/lorain0/.profile*. The jobs were run from userid *lorain0*.

All migration jobs ran successfully.

Toolkit migration

On Windows we backed up the WBIMB databases *WBICMDB* and *DWCTRLDB*. Then we used the '**export**' function in the wbimb v5 toolkit to save all projects and create a file structure for them.

We then ran the setup.exe for the new WMB V6 Toolkit. The install was successful and the toolkit was able to connect to the WBIMB V5 Configuration Manager as well as the V5 (not yet migrated) and V6 brokers on z/OS.

Configuration Manager migration on Windows

The following scripts were run as documented in the WMB V6 Information Center:

- *mqsimigratecomponents -c configmgr* pre-check (Note: Don't use the config mgr name.
- *mqsimigratecomponents configmgr* do the migration
- *mqsimigratecomponents configmgr* do the migration

  These all succeeded so we then started the config mgr
- *mqsistart configmgr*

The conf mgr started successfully but when we started the toolkit it failed to connect to the new configmgr. The event log had:

```
( ConfigMgr ) Unexpected exception in ConfigurationManager class 'initialize' method; exception text:
 ''java.lang.NoSuchFieldError: msgToken'', ''msgToken''.   An exception was caught by the ConfigurationManager class
 'initialize' method while the Configuration Manager was being started or stopped. The exception text is:
 'java.lang.NoSuchFieldError: msgToken', 'msgToken'.
```

We found an IBM technote at:

`http://www.ibm.com/support/docview.wss?rs=849&context=SSKM8N&dc=DB520&uid=swg21229211&loc=en_US&cs=UTF-8&lang=en`

describing this error which says:

Problem

Configuration Manager start up fails with BIP1002E.
java.lang.NoSuchFieldError: msgToken exception on Configuration Manager start up.

Cause

This problem occurs if the Config Manager is connected to a WebSphere® MQ V6 queue manager, but does not have the MQ Java™ Client classes located on the CLASSPATH used by the profile.

Solution

Add the following JARs to the CLASSPATH (all from inside the WMQ installation's lib directory):

```
providerutil.jar
com.ibm.mqjms.jar
ldap.jar
jta.jar
jndi.jar
jms.jar
connector.jar
fscontext.jar
```

We added each of the above jar files to the windows CLASSPATH as shown below:

```
C:\Program Files\IBM\WebSphere MQ\Java\lib\providerutil.jar;C:\Program Files\IBM\WebSphere
MQ\Java\lib\com.ibm.mqjms.jar;C:\Program Files\IBM\WebSphere MQ\Java\lib\ldap.jar;C:\Program Files\IBM\WebSphere
MQ\Java\lib\jta.jar;C:\Program Files\IBM\WebSphere MQ\Java\lib\jndi.jar;C:\Program Files\IBM\
MQ\Java\lib\jms.jar;C:\Program Files\IBM\WebSphere MQ\Java\lib\connector.jar;C:\Program Files\IBM\WebSphere
MQ\Java\lib\fscontext.jar;C:\Program Files\IBM\WebSphere MQ\Java\lib\com.ibm.mq.jar;
```

Then we tried a simple deploy which failed with *"the deployment message was addressed to a broker with a UUID 21f01d8e-0a01-0000-0080-ea101ddff920, but this does not match the UUID of the running broker (09fede6a-0a01-0000-0080-d8b172fb79c9)."*

This was fixed by altering the universally unique identifier (UUID) using the Configuration Manager Proxy API Exercisor. Start the Configuration Manager Proxy API Exerciser. This is a sample application that demonstrates the capabilities of the Configuration Manager Proxy (a comprehensive Java interface that allows you to control broker domains programmatically). To start this application, we performed the following steps:

- On Windows, click Start > IBM WebSphere Message Brokers 6.0 > Java Programming APIs > Configuration Manager Proxy API Exerciser.
- Connect to the configmgr.
- Right click the broker name then select set UUID
- Enter the new UUID value.

This is a sample application that demonstrates the capabilities of the Configuration Manager Proxy (a comprehensive Java interface that allows you to control broker domains programmatically).

Creating a z/OS Configuration Manager

We followed the instructions in the WebSphere Message Broker V6 Information Center titled "Creating a Configuration Manager on z/OS".

This task went very well with no problems.

Then we switched the WMB V6 Toolkit on Windows to connect to this new z/OS Configmgr. The userid used by the Windows machine was called '*mqtest*' in lowercase. When we deployed to the configmgr it received this RACF error:

```
SYSTEM.BROKER.CONFIG.QUEUE could not be opened (MQ reason code 2035 while trying to open the queue)
ICH408I USER(mqtest  ) GROUP(        ) NAME(???
  LOGON/JOB INITIATION - USER AT TERMINAL          NOT RACF-DEFINED
```

We had to change the Windows userid to be uppercase as lowercase userids will not work on z/OS when connecting to WebSphere MQ resources.

Then we had to authorize the Toolkit user to access the configmgr per the instructions in the WMB Info Center: See the section ″Ensure that your toolkit machine and user ID has the appropriate authorization on the z/OS Configuration Manager.″

In SDSF, grant access to your user ID. For this to work on all machines, enter:

```
'/F <started task name>,CA U=<userID>,A=YES,P=YES,X=F'
```

or to grant access to your user ID for a specific machine, enter:

```
'/F <started task name>,CA U=<userID>,A=YES,M=<machine name>,P=YES,X=F'
```

Verify the above by entering:

```
'/F <configmgrname>,LA
```

We then used the command:

```
'/F MQZ2CMGR,CA U=mqbroker/MQSTEST,A=YES,P=YES,X=F
```

The response message was +BIP8071I MQZ2CMGR 2 Successful command *completion*. We then recycled the configmgr and the toolkit then connected to configmgr successfully.

You cannot switch back and forth between configuration managers when deploying to the same brokers because of the UUID's that are assigned to the brokers. If you do try to deploy using a different configuration manager than was controlling the broker beforehand you will get an error message like:

```
BIP2045E: Broker CSQ2BRK running on WebSphere queue manager CSQ2 did not process
a deployment message, because it was addressed to a broker with a different identifier.
```

This message usually means that an attempt has been made to assign the broker to a second (or a reinitialized) Configuration Manager.

Each broker is identified by a universally unique identifier (UUID) which is allocated when the Message Brokers Toolkit or Configuration Manager Proxy creates a definition for the broker. When deployment occurs, a UUID check is made to help prevent accidental deployment of changes to brokers not under the control of the Configuration Manager. In this case, the deployment message was addressed to a broker with a UUID 7c2e2517-0d01-0000-0080-ae77adc1960f, but this does not match the UUID of the running broker (17794370-0701-0000-0080-c2dfca2e3733).

To switch to the new configmgr we used the Configuration Manager Proxy API Exercisor to change the UUID as described above.

EDSW – High Availability for WebSphere MQ-IMS bridge application

Our eDSW workload consists of a request message for an IMS transaction being placed in a queue monitored by the WebSphere MQ-IMS Bridge. The request message contains the IMS data in the IIH header. After the request has been processed by an IMS region, a message gets placed on the reply queue. The application uses the WebSphere MQ Java Message Service classes.

We have the workload spread across a couple of IMS regions handling the request messages in the shared queue through the WebSphere MQ-IMS Bridge. The clients

are TPNS users on a third system running in a third LPAR, whose queue manager is also a member of the queue sharing group. These clients use the queue sharing group name as a parameter for both the connection and the reply fields, instead of using the individual queue manager to improve high availability. Figure 27 shows the message flow.

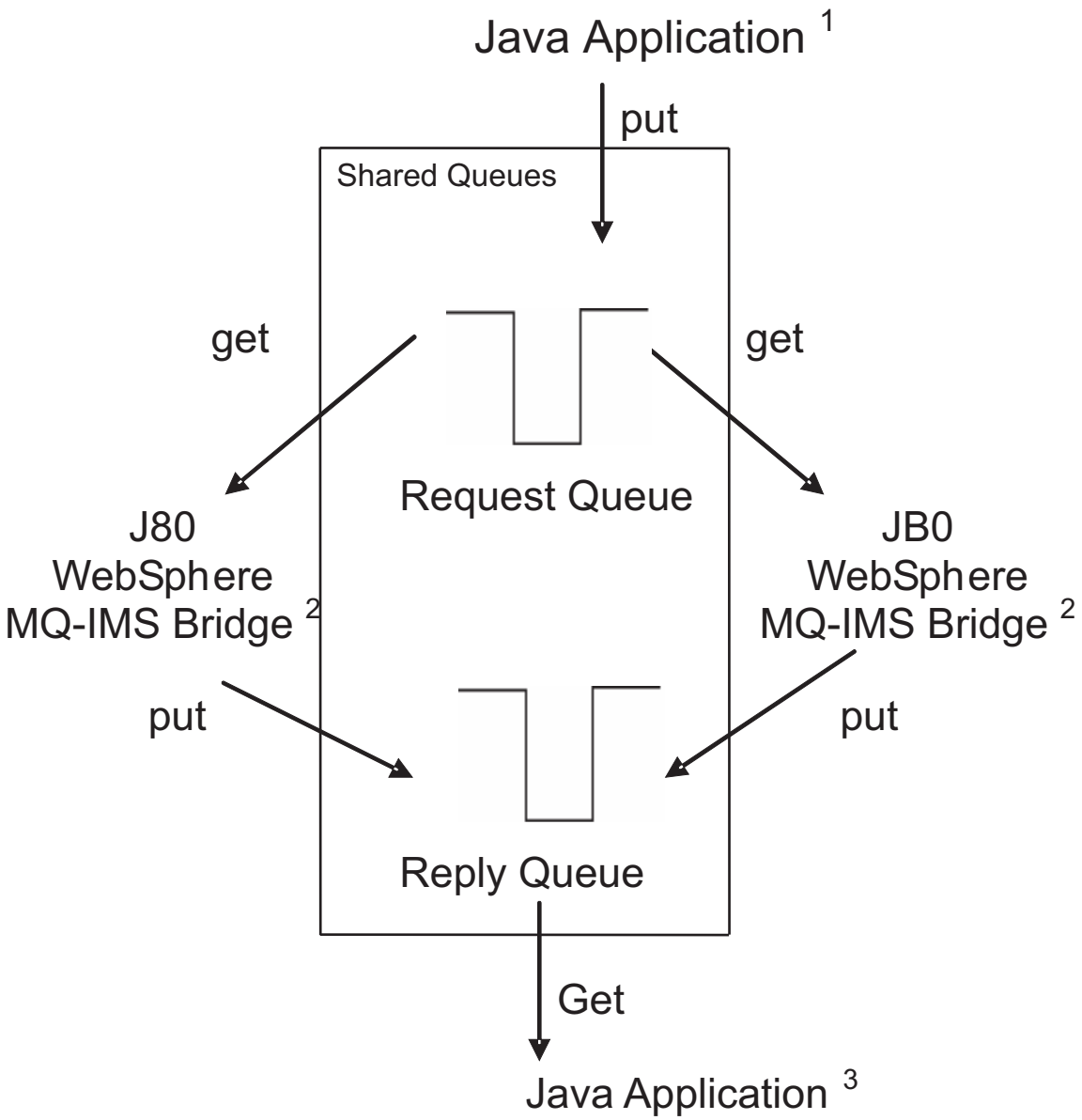


Figure 27. EDSW workload message flow

Notes:

1. Java Application is putting the messages in the shared request queue.
2. Either WebSphere MQ – IMS Bridge can pick up the message, execute the request, and put the reply in a different shared queue.
3. Java Application picks up the reply message.

Chapter 11. 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 V1R8. 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 V1R8 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 V1R8 WebSphere test environment

This section provides an overview of our z/OS V1R8 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 V1R8, our WebSphere test environment includes the following products:

- WebSphere Application Server for z/OS Version 6.0.2, service level cf130631.22
- IBM SDK for z/OS, Java 2 Technology Edition V1.4.2 (August 24, 2006 Build Date, PTF UK17593)
- 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)
- Added our zBank application (and J2EE server 7 for it)
- Implemented an enhancement of the zBank application, zCredit
- Added eWLM monitoring
- Security enhancements (TAM, TAI++, WebSeal on zLinux)
- Removed Node JH0 from P1 Cell (system removed from our test environment).

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 seven different J2EE servers, each running different applications (as described below)

Our T2 cell is configured as follows:

- Resides entirely on one of our test systems (Z2)
- Contains seven 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 three production systems in our sysplex (J80, JB0 and JF0)
- Contains six different clusters, each of which spans all three 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 T2 and T3 cells in addition to the above six applications:

- J2EE server 7 runs our zBank application used for security testing and accessing DB2

Figure 28 on page 146 shows the server address spaces in our P1 cell.

Note: The wsp1s1 cluster is not shown in the diagram.

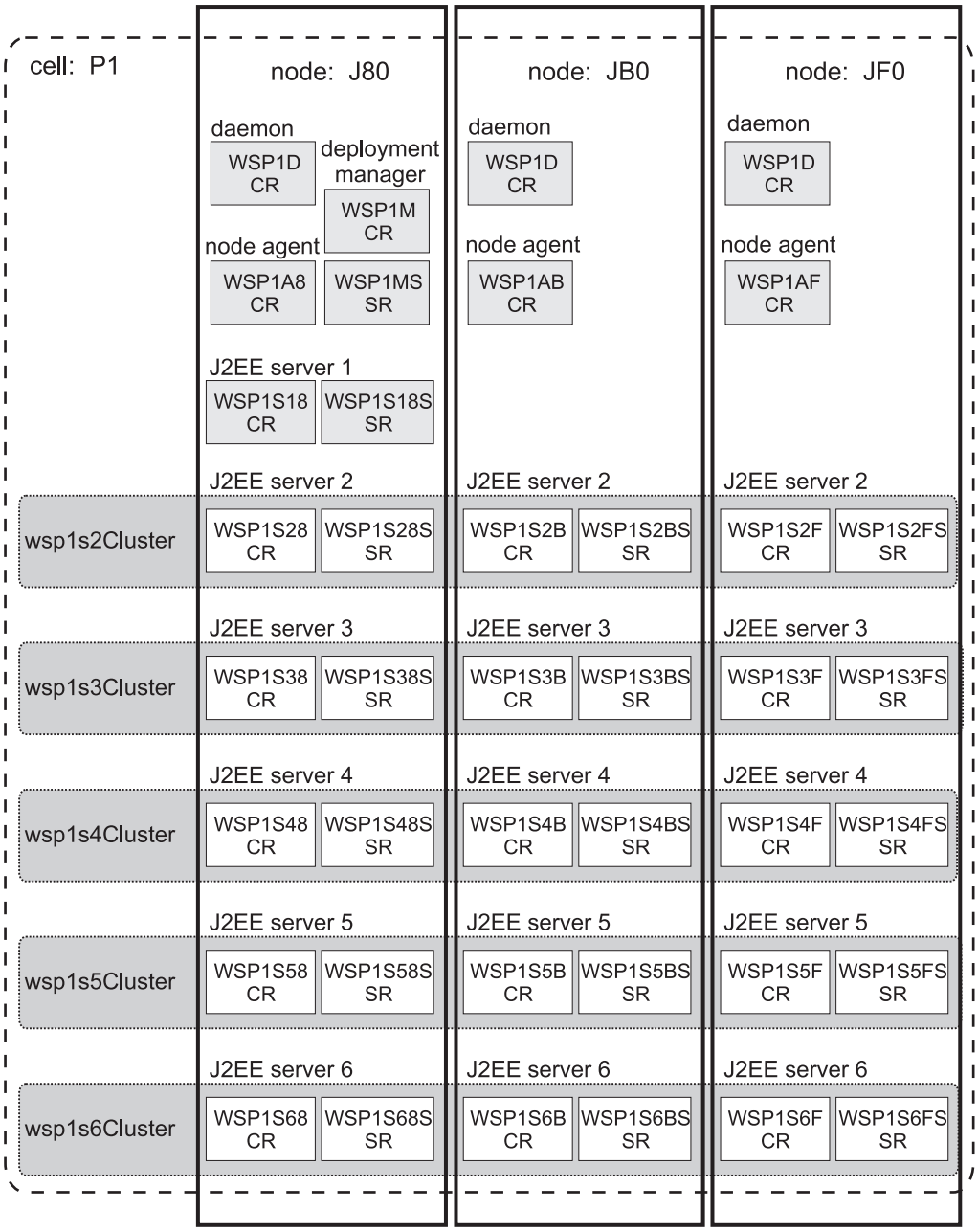


Figure 28. Our WebSphere for z/OS V6 configuration

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)

[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"
- "Setting up eWLM for Application and System Monitoring" on page 150

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 29 on page 148 is a diagram of our setup.

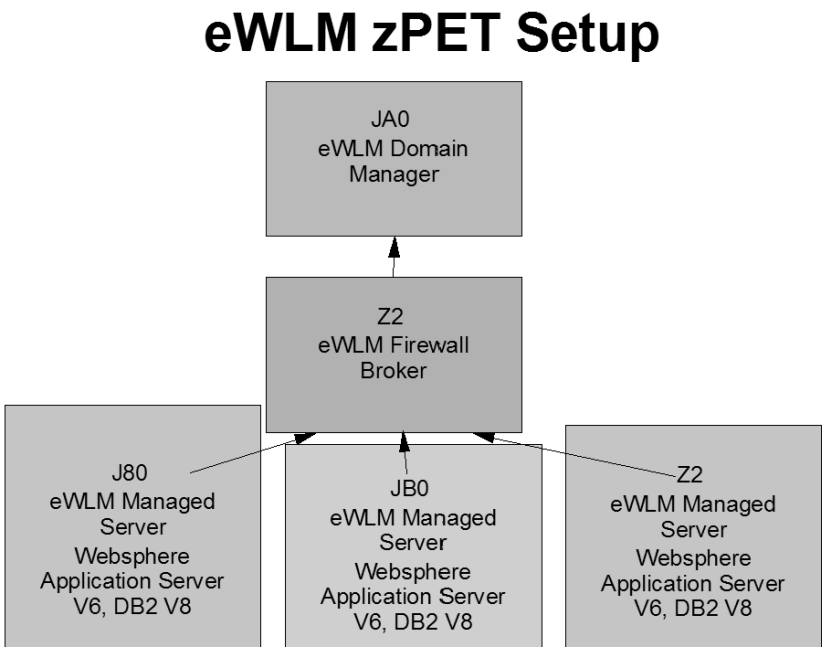


Figure 29. eWLM zPET setup

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 31 on page 150 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 30, go to “monitor” then “Transaction Classes”. Select the appropriate Transaction Class then “Application Topology” from the pull-down.

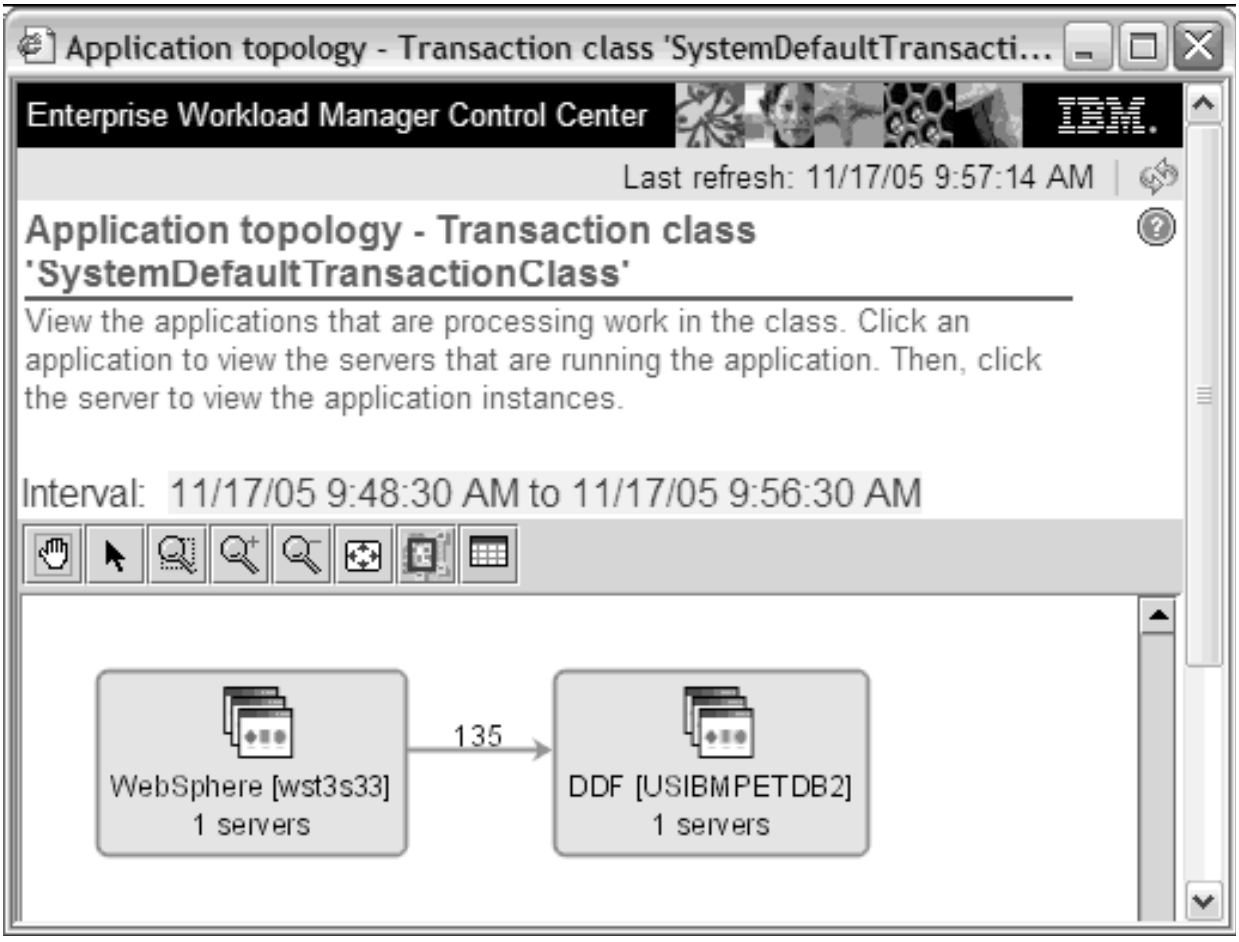
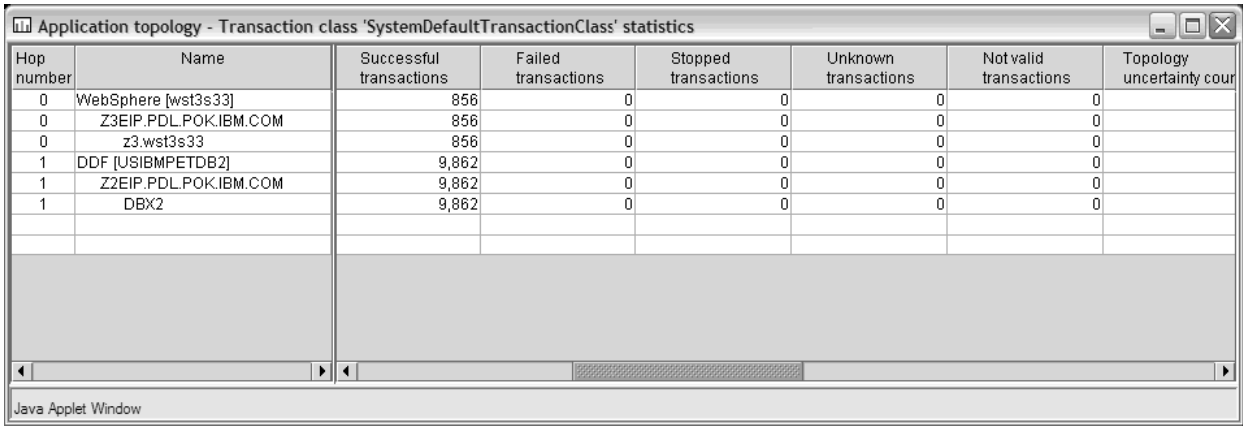


Figure 30. eWLM Control Center

If you click on the table button you will get the view showing Figure 31 on page 150



Hop number	Name	Successful transactions	Failed transactions	Stopped transactions	Unknown transactions	Not valid transactions	Topology uncertainty cour
0	WebSphere [wst3s33]	856	0	0	0	0	
0	Z3EIP.PDL.POK.IBM.COM	856	0	0	0	0	
0	z3.wst3s33	856	0	0	0	0	
1	DDF [USIBMPETDB2]	9,862	0	0	0	0	
1	Z2EIP.PDL.POK.IBM.COM	9,862	0	0	0	0	
1	DBX2	9,862	0	0	0	0	

Figure 31. 'SystemDefaultTransactionClass' statistics

Setting up eWLM for Application and System Monitoring

Our test setup consists of two parallel sysplexes. The first we call Plex1 which has 10 lpars and another, we call Plex2, that has 3 lpars. We installed eWLM 2.1 on both of our sysplexes to monitor systems as well as WebSphere Application Server workloads that use DB2. Plex1 has a Domain Manager with Managed Servers on all members of that same sysplex. Included in that domain are Managed Servers connecting to the z/OS Domain Manager from AIX , Windows, and OS/400 systems. Plex2 has a domain with only z/OS Managed Servers connected at this time.

Setting up zFS filesystems

At the time of writing this document we are at fixpack 30 for eWLM 2.1. Our filesystems consist of zFS datasets for product code, Domain Manager data and logs, and Managed Server data and logs. Below is a list of zFS datasets we use and their current size.

```
Product Filesystems:     Size in 3390 cyls:   Mountpoint:
OMVSWS.VE210.VEEWLM.JUN29        40          /ewlm20/ewlmdm_Jun29
OMVSWS.VE210.VEEWLMMS.JUN29      28          /ewlm20/ewlmms_Jun29
OMVSWS.VE210.VELIB.JUN29         26          Not used at this time. *
OMVSWS.VE210.VEWAS.JUN29       2528          /ewlm20/vewas_current
```

Note: The VELIB zFS is used for the Global Configuration Repository which we are not using at this time.

We separated the data filesystems for the Domain Manager from the Managed Server data as shown below.

- The Domain Manager uses the following filesystems which are defined as type zFS:

```
Local Filesystems:         Size in cyls:          Usage:
OMVSWS.EWLM20.DM.DATA.ZFS     1500      contains the data required for each domain server
OMVSWS.EWLM20.DMLOGS.ZFS       150      contains the startup log data and error data.
OMVSWS.EWLM20.WSV2.CONFIG.HFS  160      contains the WebSphere Application Server
```

- The Managed Servers use the following filesystems, also of type zFS:

```
Local Filesystems:          Size in cyls:    Usage:
OMVSWS.EWLM20.MS.DATA.ZFS       1000      contains server database data.
OMVSWS.EWLM20.MSLOGS.ZFS         150      contains server startup and error logs.
```

Defining filters by application

We defined filters by application so each application has it's own filter. The application name ″WebSphere:APPLICATION_SERVER″ was chosen because we are running the IBM WebSphere Application Server 6.0.2. eWLM provides the following application definitions:

```
Application name: IBM Webserving Plugin
Description: Filters for IHS/Apache, IIS and the IBM WebSphere Web server plug-ins.

Application name: IBM DB2 Universal Database
Description: IBM DB2 Universal Database

Application name: WebSphere
Description: Filters for IBM WebSphere Application Server 5.0 through 6.0.1

Application name: WebSphere:APPLICATION_SERVER
Description: Filters for IBM WebSphere Application Server 6.0.2 and later.

Application name: (\*)
Description: Default Application
```

Our filters are defined using URI match as shown in the example below:

```
Name: Trade
Description: Trade Application
Application name: WebSphere:APPLICATION_SERVER
Group name:(\*)
Service class name: WAS Service Class
Rule:
EWLM:URI stringMatch /trade/(\*)
```

Problems encountered

Following are problems we encountered in our eWLM setup and monitoring:

1. When we applied service to eWLM, the Managed Servers failed to start giving us the following error message:

   ```
   Error information: com.ibm.wlm.ea.PolicyFailureException
     eWLM Platform Service return code: -4007
   ```

   If a Managed Server comes up with other than the ewlm default domain policy, you need to start and stop each Managed Server individually before starting all of them up together. Therefore, do not have more than 1 Managed Server up at any given time until all have come up themselves on the new policy.

   In our situation we exported the old domain policy, applied new ewlm service, imported the old policy into the domain, then attempted to start the Managed Servers. Recycling each Managed Server individually fixed the problem.

2. When logging onto the Control Center, userids and passwords must be entered in uppercase due to restrictions with WebSphere Application Server 6.0. If you enter a lower or mixed case userid or password you will get an error message saying *″User null is not assigned an EWLM role that includes access to this item. ″*

Sample eWLM reports

We use eWLM to monitor our WebSphere Application Server workloads and obtain reports showing performance data for specific intervals.

You can see a sample application topology report in the ″Samples″ section of our website which is located at:

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

What we tested

Following are what we tested with eWLM monitoring:

zIIP and zAAP reporting: The intention of this test was to show that the zAAP and zIIP CPU utilization is reported by eWLM for a system that is running an eWLM enabled application that uses DB2 and Java. In order to test this we needed to make sure that the zAAPs and zIIPs get included in the number of processors that eWLM reports at the Detail Managed Server report for the system where the WebSphere Application Server/DB2 application runs. All of the processors are included in the count. A comparison was made between the CPU utilization that RMF reports and that shown by eWLM in it's detailed Managed Server report. The comparison was successful.

eWLM service class correlation: The intention of this test was to show that we can assign a WLM service class to an eWLM transaction or service class. To perform our test we:

- Created a eWLM service class *WASTrade* and Transaction class *Trade*. These were assigned to Trade6 workload that runs under z/OS. This is an ARM enabled WebSphere workload.
- Created a subsystem called EWLM and in the classification rules for that subsystem we assigned a WLM service class called EWLMWORK to an eWLM transaction class. We installed and activated the policy.
- Ran the Trade6 application on a z/OS V1R8 system and could see that its enclaves were assigned to the EWLMWORK service class, which indicates that the service class correlation works.

Using SAF (RACF) on our TCPIP.PROFILE port reserves

We have added some additional security into our TCPIP setups to prevent un-authorized programs from binding to ports that are used by our WebSphere and HTTP Servers. We already had used PORT and PORTRANGE statements in our TCPIP.PROFILEs to mark the ports used by our servers, but in many instances, only the jobname was specified. In the PORTRANGE statements, we generally had ″OMVS″ for the jobname as the range covered many different jobnames for our WebSphere Application Server cells.

Adding SAF protection to our PORT and PORTRANGE statements has helped close this hole. Now, only authorized users/groups are allowed to bind to the reserved ports.

Reserving TCPIP Port usage to a RACF userid/group

The SAF parameter of the TCP/IP profile PORT and PORTRANGE statements indicates that the port(s) are reserved for users that are permitted to the RACF resource. If an application tries to bind to a port that has the SAF keyword specified, but the user ID associated with the application is not permitted to the resource, the BIND socket call fails.

Setting up an example for WebSphere Application Server T1 Cell servers on PET System Z1

Our WebSphere Application Server T1 Cell servers use ports in the 9500-9699 range and all userids the servers run under are in the WASCFGGP RACF group.

SAF protection allows only users in the WASCFGGP group to connect as listener on this range of ports on system Z1. A resource name (resname) of WST1SRV was used. This name is arbitrary, but we decided to use the convention of

WS cell_name SRV

where

- WS indicates the WebSphere Application Server team
- cell_name is the WebSphere Application Server Cell ID (since ours our 2 characters)
- SRV is a constant indicating the WebSphere Application Server servers

To implement this, the following changes were made:

- **TCPIP.PROFILE(Z1)** was updated to add:

```
PORTRANGE 9500 200 TCP OMVS NOAUTOLOG NODELAYACKS SAF WST1SRV  ; WAS T1 Cell Servers
```

- **RACF** updates were:

  Define EZB.PORTACCESS.Z1.TCPIP.WST1SRV in the SERVAUTH class and grant WASCFGGP READ access

```
RDEFINE SERVAUTH EZB.PORTACCESS.Z1.TCPIP.WST1SRV UACC(NONE)
PERMIT  EZB.PORTACCESS.Z1.TCPIP.WST1SRV CLASS(SERVAUTH)  ID(WASCFGGP) ACC(READ)
SETROPTS RACLIST (SERVAUTH) REFRESH
```

The z/OS Communications Server (TCPIP) defined template for the RACF SERVAUTH resource is:

```
EZB.PORTACCESS.sysname.tcpname.safname
```

where:

- EZB.PORTACCESS is a constant
- *sysname* is the value of the MVS &SYSNAME system symbol (substitute your sysname, Z1 in above example)
- *tcpname* is the name of the procedure used to start the TCP stack (substitute your jobname, TCPIP in above
- *safname* is the 1-8 character name following the SAF keyword (WST1SRV in above example, you pick this name)

Reference 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 section.

- z/OS Internet Library:

  `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`

- z/OS V1R8.0 Communications Server IP Configuration Guide SC31-8775-08

  `http://publibz.boulder.ibm.com/epubs/pdf/f1a1b351.pdf`

- z/OS V1R8.0 Communications Server IP Configuration Reference (SC31-8776-09)

  `http://publibz.boulder.ibm.com/epubs/pdf/f1a1b451.pdf`

Setting up WebSphere Developer for zSeries (WDz) on PET Plex systems

This section describes our experiences installing and configuring the WebSphere Developer for zSeries (WDz).

On our z/OS systems, we installed and configured the following products:

- WebSphere Studio Enterprise Developer Options for z/OS V6.0.1 (Program number 5724-L44, FMID HEDS500)
- IBM WebSphere Developer for zSeries RSE + ICU V6.0.1 (Program number 5724-L44, FMID H001600)
- IBM WebSphere Developer for zSeries JES Job Monitor V6.0.1 (Program number 5724-L44, FMID H002600)

On the workstation side, we currently have installed:

- IBM Rational Application Developer V6.0.1.1 with Interim Fix 002
- WebSphere Application Developer for zSeries V6.0.1 with Interim Fix 003

Overall installation and configuration

Our initial install and setup was a daunting task that really required a team effort. Much of this was due to the number of products that were required for all of the functionality we wanted. These products also crossed all the ″traditional boundaries″ between MVS and Unix System Services, and between mainframes and workstations.

Some of these were fairly complicated setups.

In addition to the WDz and WSED products, changes were required in numerous other products on the zSeries side, spanning quite a range of skills. Knowledge (and authority to change!) was required in the following areas:

- Unix System Services, configurations, filesystems and user setups
- TCPIP networking setups and configurations
- APPC
- RACF (or security product configurations)

The good news is that once we had things going, it has worked well!

We also really like the integration of the Remote System Explorer within the RAD/WDz environment on the workstation. One of the values of this product is that it provides a common interface to all files accessible on z/OS. The value is in being able to have a common look and feel for all of our files and being able to drag/drop between MVS, Unix System Services and WDz projects without looking up ftp commands, which is a time saver.

You do have to be careful with some types of files, particularly with ASCII-EBCDIC conversions, but WDz provides settings for the various types of files and how to transfer them (WDz > Windows > Preferences > Remote Systems > Files) and handles most with no changes to the defaults required.

We often have Unix System Services skilled people who have limited skill in handling MVS files, as well as MVS programmer that do not have much experience from a Unix System Services perspective. This product eliminates those problems. We can easily move files from one file system to another with very little effort.

This really gives a new face to z/OS!

On the workstation side

The following are some steps and suggestions we recommend doing on the workstation side:

- Be prepared with a lot of disk space, it's a monster! Well, WDz is not too bad, but because WDz installs on top of RAD in your workstation environment, the whole package can take a significant amount of space. This can increase also depending upon the options and features you choose to install. In addition to the space needed for the actual product directories, space will also likely be needed to down-load and unpack the install directories. Please see the WDz prerequisites for determining how much disk space is needed for the product(s).
- Allow some time to perform the install. While the install itself is pretty straight-forward, it will take awhile to just perform the RAD install. After installing RAD, WDz is installed on top of RAD, followed by any other RAD options and features that you may also want to install. Once all that is complete, you'll want to run the Rational Product Updater to pick up the latest fixes. Adding it all up, it can take a 1/2 to a whole day (you get better with each install!)
- While the Rational Product Updater can make it nearly a breeze to pick up updates, fixes and options, some of the download/setups can also be extremely large. For the larger updates though, you will generally get a warning and instructions for how to download and unpack the updates to a local filesystem prior to applying.
- Install RAD into a location with a short path name (such as C:\RAD60) rather than the defaults. The path name of the directory RAD is installed in on your workstations is used in the various projects within RAD as a root. We have seen various problems with RAD due to limitations on the workstations with items such as classpath length. When coupled with the project name or run time directories, the paths can get very long. For example, when running a test server, the generated classpath became excessively long, causing the server not to start. Cutting down on the length of the install directory alleviated the issue.
- Create a separate directory to hold your workspaces and projects outside of the installed product directories. On startup, WDz will prompt for the workspace directory. Having this outside of the product helps separate out your work, plus makes it easier if you ever have to re-install RAD or switch to a newer version. Again, also try to use a short name for this directory, as it will be coupled with the workspace and project names in various places.

Setting up the zSeries side of WDz

The following are some steps and suggestions we recommend doing when setting up the zSeries side of WDz:

- Initially we got started with V6.0 and the setup was very difficult. Later service level (V6.0.1) simplifies setup process and combines setup instructions into one document.
- See latest *WebSphere Developer for zSeries Host Configuration Guide" SC31-6930-00"*, available from the WDz Library page:

  `http://www.ibm.com/software/awdtools/devzseries/library/`

  Most of the instructions for post SMP/E setup/configurations for the WDz and WSED products have been moved to this book.
- Setup for this level is also a bit easier. For the WDz product, there is only one file that really needs to be updated (*rsed.envvars*).
  - Now you need to update the *<wd4z_instal>/rse/lib/rsed.envvars* file

– Unfortunately, you still need to update this file in the product's smp/e directories. See "Hints/Tips" on page 157 for more information.

Setting up the JES job monitor for WDz

The following are some steps and suggestions we recommend doing when setting up the JES job monitor for WDz:

- The JES job monitor product comes as product: "IBM WebSphere Developer for zSeries JES Job Monitor V6.0.1 (Program number 5724-L44, FMID H002600)." We configured the product using instructions provided in the "Activating IBM WebSphere Developer for zSeries JES job monitor" chapter of the "WebSphere Developer for zSeries Host Configuration Guide".
- We used the configuration file FEJJCNFG as it was sent (lucky us! All the defaults, such as timezone, just happen to fit our environment. Please review this file for changes that may need to be made for your environment).
- We chose to set our WDz JES job monitor as a proc so that the start up could be easily automated through our automation product
- Our port of choice was the default 6715. This port was important because it must be specified not only in your start up proc but also in each of your client's WDz workstation properties file in order to get access to your MVS system.
- We also chose to set up the WDz JES job monitor on 2 images on our plex as interfaces to WDz.

Setting up the IBM WebSphere Developer for zSeries RSE + ICU V6.0.1

The following are some steps and suggestions we recommend doing when setting up the IBM WebSphere Developer for zSeries RSE + ICU V6.0.1:

- There are two address spaces that we set up as part of product: IBM WebSphere Developer for zSeries RSE + ICU V6.0.1 (WD4ZMRBS for MVS file access) and WD4ZURBS for Unix System Services file access).
- We configured the product using instructions provided in the Activating IBM WebSphere Developer for zSeries RSE + ICU chapter of the "WebSphere Developer for zSeries Host Configuration Guide".
- We used the configuration file FEJJCNFG as it was sent (again, we were lucky all the values fit our environment).
- We chose to use the default ports for these address spaces (WD4ZMRBS -- port 3500) and WD4ZURBS -- port 3501) so no change would need to be made on the client's WDz MVS or Unix System Services properties file.
- We used the default RSE_Portrange-8108-8118 defined in the *rsed.envvars*.
- We set these up as started tasks so that we could automate their start up and shutdown through our automations product.

  As part of this install, you must set up an APPC transaction program. This was a little tricky. The documentation assumes that you will be setting up a base lu with the options mentioned in the "Defining the APPC Transaction for the TSO Commands Service" section of the "WebSphere Developer for zSeries Host Configuration Guide".

  We have other workloads which use APPC in our environment which were using the base lu and the options required by WDz conflicted with the ones we already had defined for the base lu. So we needed to define another APPC lu to be used for the WDz function. See the following Tech doc for set up instructions should you encounter a problem getting to your MVS files:

  `http://www.ibm.com/support/docview.wss?rs=2294&context=SS2QJ2&context=SS2JX4&dc=DB520&uid=swg21213973&loc=en_US&cs=utf-8&lang=en`

- We also had to be creative since we have multiple systems in the sysplex running the WDz address spaces. We defined an ACBNAME of WD4Z*&SYSCLONE* in our APPCPMxx member. The *&SYSCLONE* variable is a symbolic in our parmlib which basically gets set to a unique system name. This allowed us to use the same APPCPMxx member on all our systems.

  We set up an entry in our *<hlq>*.VTAMLST member as:

  ```
  WD4Z*       APPL APPC=YES........
  ```

  The _FEKFSCMD_PARTNER_LU variable in the <wd4z_install>/rse/lib/rsed.envvars file was updated to reflect our appc lu name. See “Configuring WDz for multiple systems” on page 158 for how we handled multiple system set up.

Setting up the Websphere Studio Enterprise Developer Options for z/OS(WSED)

Customization of the WebSphere Studio Enterprise Developer Options for z/OS V6.0.1 product went quite smooth. For the most part this involved customization of the procs sent with the product and storing them in our production proclib.

We kept the default names for the procs for ease of set up for the WDz clients.

Hints/Tips

Following are some hints and tips in setting up WebSphere Developer for zSeries (WDz).

Where to look for output

One of the biggest problems we initially had was where to look for output when things went wrong. Since there are a number of components involved in the various operations, it can be difficult to determine where the problem might be. Here’s some of the more common places for error information we found useful:

```
- <user_home> directory
- <user_home>/.eclipse/RSE
```

Note: This directory must be previously created. See the techdoc for troubleshooting RSE below.

```
- /tmp/auth.log
- /tmp/debug.log
- /tmp/daemon.log
- TSO job output for jobname specified to APPC for the FEKFRSRV transaction (We used "WD4ZTSO").
- Operator console messages
- RAD/WDz logs on workstation
```

Troubleshooting

See the Support page from the WebSphere Developer for zSeries web site, especially for the ″Technotes″ for information on debugging. Many of the issues and error conditions we ran into are documented in links there.

In particular, see the following:

- *Troubleshooting Remote Server Explorer (RSE) on WebSphere Developer for zSeries V6.0* at:

  ```
  http://www.ibm.com/support/docview.wss?rs=2294&context=SS2QJ2&
  context=SS2JX4&dc=DB520&uid=swg21214997&loc=en_US&cs=utf-8&lang=en
  ```

- *Troubleshooting TSO Commands Service for WebSphere Developer for zSeries V6.0* at:

  ```
  http://www.ibm.com/support/docview.wss?rs=2294&context=SS2QJ2&
  context=SS2JX4&dc=DB520&uid=swg21213973&loc=en_US&cs=utf-8&lang=en
  ```

- *Troubleshooting: Gathering detailed logging for WebSphere Developer for zSeries problem determination* at:

  ```
  http://www1.ibm.com/support/docview.wss?rs=2294&context=SS2QJ2&
  context=SS2JX4&dc=DB520&uid=swg21218484&loc=en_US&cs=utf-8&lang=en
  ```

Using a symbolic link for product configurations

We defined a symbolic link (/wd4z/current) that points to the actual location where our smp/e product is mounted. For example /wd4z/current -> /wd4z/wd4z601/wd4z. In our case, for each new service level, we receive a dumped copy of the SMP/E filesystem(s) from our build group, which we restore and mount on our systems. Each new service level is mounted at a unique location and then only the symbolic link (/wd4z/current) needs to be updated.

This makes for easier configuration setups and allows us to change service levels without needing to update:

- rsed.envvars settings
- inetd.config
- user settings in RAD
- ccubldw.sh (Enterprise Developer Unix System Services Remote Build Server)

Configuring WDz for multiple systems

When we tried to configure WDz on multiple systems all using the same smp/e code (mounted as shared filesystem), we ran into a problem because each system required a unique setting in the rsed.envvars for the _FEKFSCMD_PARTNER_LU_ variable. Since the rsed.envvars file is embedded within the smp/e product directories, it makes it difficult to do this.

To work around this limitation, we replaced the file at

```
<wd4z_install>/rse/lib/rsed.envvars
```

with a symbolic link to a file in the system specific /etc directories.

A unique copy of the rsed.envvars file was then placed in each system's /etc directory that we setup for running WDz.

We did the following to perform this:

- Copied <wd4z_install_dir>/rse/lib/rsed.envvars to each system configured for WDz's /etc directory (for example; /Z1/etc/rsed.envvars)
- Backed up <wd4z_install_dir>/rse/lib/rsed.envvars as rsed.envvars.orig
- Created new symlink for <wd4z_install_dir>/rse/lib/rsed.envvars to $SYSNAME/etc/rsed.envvars using the following Unix System Services command:

  ```
  ln -s '$SYSNAME'/etc/rsed.envvars <wd4z_install_dir>/rse/lib/rsed.envvars
  ```

  on our system Z1 for example, when coupled with our symlink for the product (see above),

/wd4z/current/lib/rsed.envvars resolves to /Z1/etc/rsed.envvars. In the /Z1/etc/rsed.envvars file, we specified the system unique APPC LU name configured for WDz:

```
_FEKFSCMD_PARTNER_LU_=WD4ZZ1
```

Networking

- We used the default ports for the various products. This made it easier for the RAD/WDz workstation users to get setup since they could also generally use the defaults. Once they had WDz installed on their workstations, setup and connecting to the zSeries system required only knowing the server's hostname and their userid/password on the system. Taking the defaults at this point saved them a lot of confusion.

  Again, we were lucky here in that the ports and ranges required were not already in use by any other product or application in our systems. You should check with your networking setups to determine which ports to use.
- For the same reason of ease of setup on the workstation side, we also preferred the INETD RSE Daemon setup, rather than the INETD REXEC (Unix System Services) setup. Using REXEC, the users would generally have to change the script name used by REXEC, as we did not install WDz products in the default directories on the z/OS side (/usr/lpp/wd4z). However, since we used a symbolic link for this location on the zSeries side (/wd4z/current) this was a one-time setup change for the users.
- Our networking setups on our systems use TCPIP.ETC.SERVICES, rather than /etc/services described in the setup book. Check with your networking group to find out what is used in your installation.
- You will likely need to refresh resolver to pick up the change in TCPIP.ETC.SERVICES. This can be done dynamically with:

  ```
  "F RESOLVER,REFRESH"
  ```
- After updating the /etc/inetd.config file, we nohup'd inetd for it to pick up our changes, rather than stopping and restarting. This can be done with the following Unix System Services shell command:

  ```
  kill -HUP <inetd_pid>
  ```

  Note: the <inetd_pid> can generally be found in the /etc/inetd.pid file and by using the ″ps″ Unix System Services command.)

Reference 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 section.

- WebSphere Developer for zSeries homepage (This site has links to it all!):

  ```
  http://www.ibm.com/software/awdtools/devzseries/
  ```
- z/OS Internet Library:

  ```
  http://www.ibm.com/servers/eserver/zseries/zos/bkserv/
  ```

  See the various bookshelves for the additional products required on the zSeries side.

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 159 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 2. Linux virtual servers

The following chapters describe the Linux virtual servers aspects of our computing environment.

Chapter 12. About our environment

In this chapter, the System z Linux Virtual Servers Platform Evaluation Test (zLVS PET) will discuss the implementation of a set of reference architectures for service oriented architectures and traditional web workloads. The information presented describes the sample workloads executed in the IBM labs as well as the system configurations, recommendations, and best practices discovered by our team. Though we focus on modeling customers who perform workloads based on web technologies, many of the tools, configurations and recommendations apply to anyone deploying highly available Linux instances. In addition to that focus, this document provides insights to help make your databases, directory servers, edge components, directory servers, and security infrastructure more reliable and fault tolerant.

Regular readers of this test report who closely follow our environment reference architecture may notice some important changes. Due to customer feedback and interaction, the lab has redesigned a few key aspects of our critical workload infrastructure. In previous iterations of our test environment, Linux installations were constrained to a plurality of IBM z/VM instances executing a single physical mainframe. Our readers and customers have indicated to us a strong trend towards LPAR stand alone Linux instances for certain workloads and applications. Additionally many customers discussed their desire to see more cross mainframe discussion and implementation in our test environment and subsequent reports.

As a result of this reader feedback and customer insights, we saw an opportunity to enhance the reliability, availability, and serviceability of our infrastructure and applications. No longer do we describe HA mechanisms contained within a single LPAR as we have previously addressed. In this document, you will read about the experiences found in the lab when doing hands on integration testing across z/VM LPARS to make workloads and applications even more robust. But we didn't stop there. Our HA solutions described in this document extend to deployments that span multiple physical mainframes consisting of z/VM supported Linux instances, as well as native LPAR configurations.

During our environment redesign, we also had the opportunity to enhance the availability of our networking infrastructure, DASD and disk I/O, and software products. Some of the methods described are shockingly simple ways to exploit the advanced functionality of the system z platform. Other implementations require more advanced setup but are well worth the effort. This document serves to outline these evolutionary steps in the test lab environment, and reflects our continued commitment to performing integration testing in a customer like way based on your feedback.

Our workloads

The workloads we have selected for execution in our test environments run on IBM WebSphere Application Server. For variety we execute two workloads concurrently. One workload, called Trade6, is designed to simulate a corporation that places stock trades and orders. The other application, simply called Bookstore, is modeled in spirit after major on-line book retailers.

The stock trading workload exercises WebSphere Application Server, DB2 on Linux, and WebSphere Message Queue (MQ) in addition to the usual networking and security infrastructure. This particular workload exploits JDBC, including session-based servlets and EJBs.

The bookstore application exercises WebSphere Application Server, MQ, and DB2 z/OS Data Sharing group. This workload includes a web-based portal enabling users to browse for, and order books. Bookstore exploits JDBC, session-based servlets, EJBs, MQSeries, and is populated with an extremely large dataset from the Library of Congress.

Overall configuration

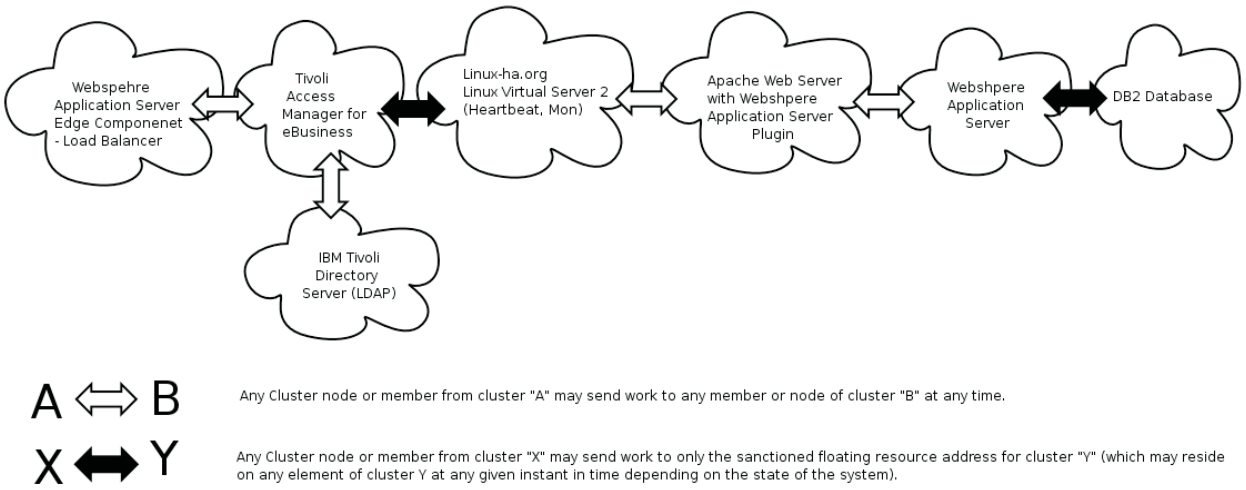


Figure 32. Logical flow of a transaction.

Figure 32 depicts the logical flow of a transaction. The clouds in the picture depict application clusters. Each cluster has members that are split across two z/VM LPARs on two different CEC's. Cluster members of LVS Director, Apache, and WebSphere Application Server are also split across native Linux LPARs.

For a typical new transaction, the flow to access both applications is:

1. Client initiates application request from the "outside" world.
2. The request is handled by the firewall and passed to the TAMe WebSEAL cluster address.
3. WebSphere Application Server Network Deployment Edge Component Load Balancer handles spraying the request to a member of the TAMe WebSEAL cluster.
4. WebSEAL then asks the end user for authentication and authorization information.
5. The end user inputs this information.
6. WebSEAL checks against the LDAP user registry for authentication and authorization. If OK, then WebSEAL passes the request to the Apache cluster address.*
7. The Linux Virtual Server Director handles spraying the request to an available Apache server.
8. The WebSphere Application Server Plug-in that is installed on the Apache server, transfers the request to an available member in the WebSphere Application Server cluster.

9. WebSphere Application Server fulfills the request. If the request involves a transaction to DB2, then WebSphere Application Server uses the JDBC Type 4 driver to pass the request onto DB2.
   a. For Trade, the request goes to the DB2 UDB cluster on Linux.
   b. For BookStore, the request goes to DB2 z/OS datasharing group and shared Message Queue, depending on the type of transaction.
10. Once the request is fulfilled, the response is bubbled back to the client.

System names and usages

Table 11 lists our systems, their hostnames, IP addresses, and usages. Throughout this test report, in discussions as well as examples, references are made to these systems either by their hostnames or their IP addresses.

Table 11. Linux virtual servers system names, IP addresses and usages

Hostname	IP address	Usage
litslb01	192.168.74.99	Backup Load Balancer
litslb02	192.168.74.135	Primary Load Balancer
litstam2	192.168.74.112	IBM Tivoli Access Manager for e-business WebSEAL
litstam3	192.168.74.113	IBM Tivoli Access Manager for e-business WebSEAL
litrwas1	192.168.71.137	WebSphere Application Server
litrwas2	192.168.71.129	WebSphere Application Server
litrwas3	192.168.71.130	WebSphere Application Server
litrwas4	192.168.71.138	WebSphere Application Server Network Deployment
litswas1	192.168.71.101	WebSphere Application Server
litswas2	192.168.71.102	WebSphere Application Server
litswas3	192.168.71.105	WebSphere Application Server
litsdns1	192.168.71.128	Primary DNS Server
litsdns2	192.168.71.171	Secondary DNS Server
litsha21	192.168.71.201	Linux-HA version 2 Linux Virtual Server Director
litsha22	192.168.71.202	Linux-HA version 2 Linux Virtual Server Director
litsha23	192.168.71.203	Linux-HA version 2 Linux Virtual Server Director
litsldap	192.168.71.174	Secondary LDAP server
litsldp2	192.168.71.187	Primary LDAP server
litstat1	192.168.71.220	Apache Web server with WebSphere Application Server Plug-in

Table 11. Linux virtual servers system names, IP addresses and usages (continued)

Hostname	IP address	Usage
litstat2	192.168.71.150	Apache Web server with WebSphere Application Server Plug-in
litstat3	192.168.71.121	Apache Web server with WebSphere Application Server Plug-in
litstat4	192.168.71.145	Apache Web server with WebSphere Application Server Plug-in
litstat5	192.168.71.185	Apache Web server with WebSphere Application Server Plug-in
litstat6	192.168.71.186	Apache Web server with WebSphere Application Server Plug-in
litdat01	192.168.71.104	Backup DB2 UDB
litdat02	192.168.71.117	Primary DB2 UDB

IPLing z/VM

The order in which our Linux application servers come up is critical to our environment initializing properly. We handle this task by utilizing an exec running on AUTOLOG2 after the z/VM system is IPLed. The following is a sample of the exec from one of our z/VM systems.

```
/*********************************************************************/
/* Sample nodeid EXEC for Integration Test.                          */
/*********************************************************************/
/* ensure that the network is up before           */
/* xautologging systems, sleep for 2 minutes      */
  'CP SLEEP 2 MIN'
  'XAUTOLOG LITSDNS2'
  'XAUTOLOG LITSLDP2'
  'XAUTOLOG LITDAT02'
  'XAUTOLOG LITRWAS2'
  'XAUTOLOG LITSWAS2'
  'CP SLEEP 5 MIN'
  'XAUTOLOG LITSTAT3'
  'XAUTOLOG LITSTAT4'
  'CP SLEEP 1 MIN'
  'XAUTOLOG LITSHA21'
  'XAUTOLOG LITSTAM3'
  'CP SLEEP 1 MIN'
  'XAUTOLOG LITSLB02'
exit
```

The first systems we bring up are typically the ones that take the longest to initialize or are required by another Linux system. In our case, the DNS server, LPAP server, DB2 database and the WebSphere Application Servers are brought up first. We provide a five minute delay before bringing up the Apache Servers giving the initial set of Linux systems time to initialize. After a one minute delay, we then start the LVS Director and the Tivoli Access Manager WebSEAL. Finally the WebSphere Application Server Edge Component Load Balancer is brought up.

Automating Linux startup with a profile exec

To further automate our start up process, we utilize a common profile exec for all of our Linux service machines. In our z/VM directory, we specify this shared disk read only which contains the following profile exec.

```
/* Spool the console to the rdr */
'spool console start to *'

/* Prevent CP READ on reconnect */
'SET RUN ON'

/* 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  say "Enter IPL 201 CLEAR      to start linux"
EXIT
```

Adding Linux init scripts

As part of the recovery test, we attempted to add init scripts on all our Linux systems so that when they start, the middleware products and applications start automatically. We will talk about these as we talk about Linux software servers in the Recovering from Software Server Failure section.

Chapter 13. zLVS PET Recovery

In this chapter, we focus on describing:

- How to be prepared for potential problems in a Linux Virtual Server environment
- What indicators are to let you know there's a problem
- What actions to take to recover

The information is organized into the following categories:

Where to find more information: If you are familiar with our December 2006 edition, you might remember that we implemented high availability for Stonesoft StoneGate, WebSphere Application Server Network Deployment Edge Component Load Balancer, LVS Directors, Apache web server, WebSphere Application Server, and DB2. Now we have expanded our high availability coverage to WebSEAL and LDAP, updated our implementation of LVS Directors, and spread our infrastructure to multiple z/VM LPARs and native Linux LPARs across two different CECs.

For the high availability implementation and recovery of Stonesoft StoneGate, WebSphere Application Server, DB2 UDB on Linux, and DB2 z/OS data sharing group, please refer to the *zSeries Platform Test Report*.

Recovery process overview

Our overall recovery process is to follow the general steps described below. In a recovery situation, many activities can be happening simultaneously; some of the steps outlined below might be overlapping. Our first priority is to keep as much of our work processing as possible so that end users are not aware there's a problem.

Here are the general steps we follow:

1. Be prepared. Careful planning ahead of time makes recovery less painful.
2. Document your recovery procedures and have those procedures readily available to anyone who might need them. The information in this chapter provides the foundation for a recovery procedures document you can tailor to your own installation.
3. Recognize the indicators. You need to understand the scope and impact of the problem, and enter the problem details in a log.
4. Gather the appropriate diagnostic documentation. Examples of documentation include a system or application log, a standalone dump, or some type of trace.
5. Start recovery actions; restabilize the systems and the workload. An example of a recovery action would be bringing back the failed crypto CHIPID.
6. Initiate a fix for the problem and return to normal operations; return all applications and systems structures to their normal locations if changed.
7. Identify the root cause of the problem and take corrective action.
8. Record the entire incident in a log.

Recovery from a resource failure

In this section, we discuss what we do when network, disk, cryptographic card, and VM console fails. For network failures, we discuss some “Gotcha’s” we came across during this recovery test. We also discuss two high availability options: VSWITCH and channel bonding. For disk failures, we discuss multipathing DASD to address path failures, and Geographically Dispersed Parallel Sysplex/Peer-to-Peer Remote Copy (GDPS/PPRC) Multiplatform Resilience for DASD failures.

Networking Gotcha’s

Running Linux systems on System z presented a few Gotcha’s that may not be encountered on some other platforms. We felt the numerous benefits of running Linux on System z made it well worth the time we spent investigating this small set of gotcha’s. Most of these issues were related to using OSA and our initial lack of understanding of how the adapter works with multiple Linux systems on a single CEC.

Gotcha #1 – Pri-router and OSA Layer3

Linux systems (native or z/VM guests) running a router or Network Address Translation (NAT) Server using OSA in Layer3 mode need to have pri-router set up. We learned the hard way that you can only have a single Linux system with pri-router per OSA card. Therefore, for each router or NAT Server in your environment, a separate OSA is required. The good news is that you can have as many non pri-router Linux systems sharing that same OSA as you want.

Gotcha #2 – OSA Layer2 setup

On System z you can configure an OSA in Layer2 mode. This removes the limitation of only having one router or NAT Server per OSA. It also removes the need to set pri-router in the Ethernet definitions. To configure Linux to use OSA in layer2 mode, issue the following Linux command:

```
echo 1 > /sys/bus/ccwgroup/drivers/qeth/x.x.xxxx/layer2
```

when defining the Ethernet interface dynamically. Alternately, you can select Layer2 when defining the network interface during Linux install.

Layer2 mode also makes debugging problems with a network sniffer possible because each Linux guest will have a unique MAC address. This is unlike Linux guests using Layer 3 which all share the identical MAC.

The Gotcha – An OSA card will be set to Layer2 or Layer3 mode when the first IP address is registered with the card. Ensure that all users of the OSA are aware which mode is intended. If mistakenly the first Linux system registers an IP with the card in Layer3 mode you will need to reset the OSA to correct this. That means you have to vary the card offline then online to all LPARS. We recommend using OSA to Layer2 mode whenever possible.

Gotcha #3 – Consistent MTU Size for Linux

The default Maximum Translation Unit (MTU) size for SUSE LINUX Enterprise Server 9, SUSE LINUX Enterprise Server 10, and RedHat Enterprise Linux 4 is 1492. Unless overridden in the network interface configuration file, Linux images for those distributions will have a default MTU size of 1492.

In our environment we had some of our Linux images using the default MTU of 1492, while others were setup with a MTU of 1500. These inconsistent MTU sizes caused problems with our web based workloads. The symptoms we experienced

were Web page requests that timed out intermittently. While debugging the problem we discovered it actually was very consistent. It was related to the path a Web page request took from server to server and whether or not the returned page fit within a single packet based on a MTU of 1492.

The solution to the problem was a very easy one. We set the MTU size to 1492 on all systems in our environment. Once we did this we had no problem retrieving Web pages regardless of size or path taken.

To set the MTU size dynamically, we ran the command:

```
# ifconfig eth0 mtu 1492
```

To set the MTU size in the Linux network interface configuration file, so that systems boot with the desired MTU size, go to /etc/sysconfig/network-scripts/ifcfg-<interface> on SUSE, or /etc/sysconfig/network-scripts/ifcfg-<interface> on RedHat, and add:

```
MTU=""
```

Gotcha #4 – IP address is already in use on the LAN

We experienced the following problem on a RedHat Enterprise Linux system. This was a known problem and is documented in the RedHat Knowledgebase. When trying to start a virtual NIC (VSWITCH) device at boot time, we saw the following error messages:

```
HCPIPN2833E Error 'E00A'X adding IP address xx.xx.xx.xx for VSWITCH SYSTEM IT71.
HCPIPN2833E IP address is already in use on the LAN.
```

The problem is related to the virtual NIC being configured in Local mode. When querying the VSWITCH on z/VM, you will see xx.xx.xx.xx in Local mode:

```
CP Q VSWITCH IT71 DET
VSWITCH SYSTEM IT71     Type: VSWITCH Connected: 1    Maxconn: INFINITE
  PERSISTENT  RESTRICTED    PRIROUTER                 Accounting: OFF
  VLAN Aware  Default VLAN: 0001    Default Porttype: Access
  State: Ready
  IPTimeout: 5         QueueStorage: 8
  Portname: UNASSIGNED RDEV: 1104 Controller: DTCVSW2  VDEV:  1104
    VSWITCH Connection:
      RX Packets: 78         Discarded: 139        Errors: 0
      TX Packets: 49         Discarded: 0          Errors: 0
      RX Bytes: 17747                TX Bytes: 8404
      Device: 1106  Unit: 002   Role: DATA
      Options: VLAN_ARP
    Adapter Owner: LITROUT2 NIC: 0700  Name: IT71
      Porttype: Access
      RX Packets: 251        Discarded: 0          Errors: 0
      TX Packets: 183        Discarded: 0          Errors: 0
      RX Bytes: 57815                TX Bytes: 37602
      Device: 0702  Unit: 002   Role: DATA
      VLAN: 0671 Assigned by system
      Router: Primary   IPv4 VLAN 0671
      Options: Broadcast Multicast IPv6 IPv4 VLAN
        Unicast IP Addresses:
          xx.xx.xx.xx          MAC: 02-00-00-00-00-13 Local
          FE80::200:0:100:13   MAC: 02-00-00-00-00-13 Local
        Multicast IP Addresses:
          224.0.0.1            MAC: 01-00-5E-00-00-01
          FF02::1              MAC: 33-33-00-00-00-01 Local
```

To get around this problem we issued the following command:

```
# echo ARP=no >> /etc/sysconfig/network-scripts/ifcfg-eth0
```

Recovering from a network failure: VSWITCH

Description

VSWITCH technology is the most robust way we have found to add Networking HA to our Linux guests with very little setup. The real beauty to Networking HA using a VSWITCH with a Linux guest is that it is completely transparent to the guest itself. No additional configuration was needed in the Linux guest to take advantage of the HA properties of using an HA configured VSWITCH. VSWITCH's technology in z/VM also reduces the number of OSA devices you need to define in your IODF since a single trio of OSA devices attached to a VSWTICH can service numerous Linux guests. In our testing we found this technology to be rock solid with both LAYER2 and LAYER3 VSWITCH.

In this section, we discuss what happens when you lose a VSWITCH controller, an OSA device attached to VSWITCH, and when you experience a failure of Fiber or Cat5 cable connected to OSA used by a VSWITCH.

Preparation

Setting up the VSWITCH for High Availability involves defining two VSWITCH controllers in the z/VM directory and attaching two sets of OSA devices to each VSWITCH defined.

Although written for z/VM 5.1, the white paper at

`http://linuxvm.org/present/misc/vswitch.pdf`

has excellent step by step instructions on setting up a VSWITCH for HA. In z/VM 5.2 you no longer need to define the VSWITCH controller manually.

Additional information on z/VM VSWITCH setup can be found in Chapter 7 of the *z/VM Virtual Machine Operation* manual, or the *TCP/IP Planning and Customization* manual, for detailed information on the commands used.

Loss of a VSWITCH controller

Loss of VSWITCH controller: The failure involved a CP FORCE of one of the VSWITCH controllers to simulate the lost or crash of the VSWITCH controller.

Indicators and Failover: On the z/VM Operator Console messages were displayed noting the loss of the VSWITCH controller and which VSWITCHs they controlled:

```
HCPSWU2843E The path was severed for TCP/IP Controller DTCVSW1.
HCPSWU2843E It was managing device 09C7 for VSWITCH SYSTEM PRVV68.
```

Immediately following were messages indicating that the backup controller had taken control and the VSWITCH was ready:

```
HCPSWU2859I Eligible controllers for VSWITCH SYSTEM PRVV68 do not support GVRP.
HCPSWU2830I VSWITCH SYSTEM PRVV68 status is ready.
HCPSWU2830I DTCVSW2 is VSWITCH controller.
```

No Indicators were reported by the Linux Systems using the VSWITCH – the failure was transparent to the Linux Guest.

Failback Actions and Messages: Once the problem with the VSWITCH controller had been identified and corrected, XAUTOLOG the VSWITCH controller. The controller will activate and serve as the backup controller for the VSWITCH.

Notes:

1. If you add the VSWITCH controllers to the AUTOLOG section in TCPIP profile for the main TCPIP stack, the controller will autolog automatically.
2. We tested this scenario with up to five VSWITCHs attached to the controller. When the failure was initiated it performed flawlessly.

Loss of OSA devices attached to a VSWITCH

The failure involved detaching the OSA devices from an active VSWITCH controller simulating an OSA failure.

Indicators and Failover: On the z/VM Operator Console messages were displayed noting the VSWITCH had detected and error and was taking recovery action followed by a message that the VSWITCH was ready.

```
HCPSWU2830I VSWITCH SYSTEM IT71 status is devices attached.
HCPSWU2830I DTCVSW2 is VSWITCH controller.
HCPSWU2830I VSWITCH SYSTEM IT71 status is in error recovery.
HCPSWU2830I DTCVSW1 is new VSWITCH controller.
HCPSWU2830I VSWITCH SYSTEM IT71 status is ready.
HCPSWU2830I DTCVSW1 is VSWITCH controller.
```

No Indicators were reported by the Linux Systems using the VSWITCH – the failure was transparent to the Linux guest.

Failback Actions and Messages: Have your hardware tech correct the problem with the OSA and place it back online. Once online, the VSWITCH will automatically reattach the devices and make them available for the backup VSWITCH controller.

Failure of Fiber or copper cable connected to OSA used by a VSWITCH

To simulate a failure of Fiber or Cat5 cable connected to OSA used by a VSWITCH we unplugged the fiber from the Ethernet port on the OSA.

Indicators and Failover: On the z/VM Operator Console, messages were displayed noting the loss of connectivity to the LAN:

```
DTCVSW1 : 14:56:16 DTCOSD309W Received adapter-initiated Stop Lan
DTCVSW1 : 14:56:16 DTCOSD082E VSWITCH-OSD shutting down:
DTCVSW1 : 14:56:16 DTCPRI385I    Device IT71C200DEV:
DTCVSW1 : 14:56:16 DTCPRI386I       Type: VSWITCH-OSD, Status: Ready
DTCVSW1 : 14:56:16 DTCPRI387I       Envelope queue size: 0
DTCVSW1 : 14:56:16 DTCPRI388I       Address: C200
DTCVSW1 : 14:56:16 DTCQDI001I QDIO device IT71C200DEV device number C202:
DTCVSW1 : 14:56:16 DTCQDI007I   Disable for QDIO data transfers
DTCVSW1 : 14:56:17 DTCOSD102I VSWITCH-OSD device IT71C200DEV: Restarting device C200 through AUTORESTART
DTCVSW1 : 14:56:17 DTCOSD080I VSWITCH-OSD initializing:
DTCVSW1 : 14:56:17 DTCPRI385I  Device IT71C200DEV:
DTCVSW1 : 14:56:17 DTCPRI386I     Type: VSWITCH-OSD, Status: Not started
DTCVSW1 : 14:56:17 DTCPRI387I     Envelope queue size: 0
DTCVSW1 : 14:56:17 DTCPRI388I     Address: C200
DTCVSW1 : 14:56:17 DTCOSD082E VSWITCH-OSD shutting down:
DTCVSW1 : 14:56:17 DTCPRI385I    Device IT71C200DEV:
DTCVSW1 : 14:56:17 DTCPRI386I       Type: VSWITCH-OSD, Status: CSCH on Write Device
DTCVSW1 : 14:56:17 DTCPRI387I       Envelope queue size: 0
DTCVSW1 : 14:56:17 DTCPRI388I       Address: C200
DTCVSW1 : 14:56:17 DTCOSD361I VSWITCH-OSD link removed for IT71C200DEV
14:56:17 HCPSWU2830I VSWITCH SYSTEM IT71 status is devices attached.
HCPSWU2830I DTCVSW1 is VSWITCH controller.
14:56:17 HCPSWU2830I VSWITCH SYSTEM IT71 status is in error recovery.
HCPSWU2830I DTCVSW2 is new VSWITCH controller.
DTCVSW1 : OSA C200 DETACHED DTCVSW1 C200 BY DTCVSW1
14:56:17 OSA C200 DETACHED DTCVSW1 C200 BY DTCVSW1
DTCVSW1 : OSA C201 DETACHED DTCVSW1 C201 BY DTCVSW1
14:56:17 OSA C201 DETACHED DTCVSW1 C201 BY DTCVSW1
DTCVSW1 : OSA C202 DETACHED DTCVSW1 C202 BY DTCVSW1
14:56:17 OSA C202 DETACHED DTCVSW1 C202 BY DTCVSW1
DTCVSW2 : 14:56:17 DTCOSD080I VSWITCH-OSD initializing:
DTCVSW2 : 14:56:17 DTCPRI385I  Device IT710800DEV:
DTCVSW2 : 14:56:17 DTCPRI386I     Type: VSWITCH-OSD, Status: Not started
DTCVSW2 : 14:56:17 DTCPRI387I     Envelope queue size: 0
```

```
DTCVSW2 : 14:56:17 DTCPRI388I     Address: 0800
DTCVSW2 : 14:56:17 DTCQDI001I QDIO device IT710800DEV device number 0802:
14:56:17 HCPSWU2830I VSWITCH SYSTEM IT71 status is ready.
HCPSWU2830I DTCVSW2 is VSWITCH controller.
DTCVSW1 : 14:56:17 DTCOSD360I VSWITCH-OSD link added for IT71C200DEV
DTCVSW2 : 14:56:17 DTCQDI007I   Enabled for QDIO data transfers
DTCVSW2 : 14:56:17 DTCOSD238I ToOsd: IPv4 multicast support enabled for IT710800DEV
DTCVSW2 : 14:56:17 DTCOSD319I ProcessSetArpCache: Supported for device IT710800DEV
DTCVSW2 : 14:56:17 DTCOSD341I Obtained MAC address 00096B1A0582 for device IT710800DEV
DTCVSW2 : 14:56:17 DTCOSD234I ToOsd: TCPIP host is set as the PRIMARY router for port IT71 for IPv4
DTCVSW2 : 14:56:17 DTCOSD238I ToOsd: IPv6 multicast support enabled for IT710800DEV
DTCVSW2 : 14:56:21 DTCOSD246I VSWITCH-OSD device IT710800DEV: Assigned IPv4 address 192.168.71.97
DTCVSW2 : 14:56:21 DTCOSD246I VSWITCH-OSD device IT710800DEV: Assigned IPv4 address 192.168.71.93
```

Failback Actions and Messages: Repair or replace the failing cable attached to the OSA. We found it interesting that no messages were reported on the z/VM operator console when the cable was reattached to the OSA. However, the VSWTICH controller did activate the OSA and make it available for backup as noted by the Q VSWITCH output below.

```
Q VSWITCH IT71 DETAILS
15:00:01 VSWITCH SYSTEM IT71     Type: VSWITCH Connected: 1    Maxconn: INFINITE
15:00:01   PERSISTENT  RESTRICTED    PRIROUTER                 Accounting: OFF
15:00:01   VLAN Unaware
15:00:01   State: Ready
15:00:01   IPTimeout: 5         QueueStorage: 8
15:00:01   Portname: IT71       RDEV: 0800 Controller: DTCVSW2  VDEV:  0800
15:00:01   Portname: UNASSIGNED RDEV: C200 Controller: DTCVSW1  VDEV:  C200 BACKUP
```

In order to verify that the backup devices were truly available and working, we detached OSA devices 0800-0802 from the VSWITCH controller DTCVSW2. Failover took place to the C200 devices and work continued to flow through the VSWITCH as expected.

```
DET 0800-0802 DTCVSW2
DTCVSW2 : 0800-0802 DETACHED BY OPERATOR
15:02:05 0800-0802 DETACHED DTCVSW2
Ready; T=0.01/0.01 15:02:05
DTCVSW2 : 15:02:05 DTCOSD082E VSWITCH-OSD shutting down:
DTCVSW2 : 15:02:05 DTCPRI385I    Device IT710800DEV:
DTCVSW2 : 15:02:05 DTCPRI386I       Type: VSWITCH-OSD, Status: Ready
DTCVSW2 : 15:02:05 DTCPRI387I       Envelope queue size: 0
DTCVSW2 : 15:02:05 DTCPRI388I       Address: 0800
DTCVSW2 : 15:02:05 DTCQDI001I QDIO device IT710800DEV device number 0802:
DTCVSW2 : 15:02:05 DTCQDI007I   Disable for QDIO data transfers
DTCVSW2 : 15:02:05 DTCOSD361I VSWITCH-OSD link removed for IT710800DEV
15:02:05 HCPSWU2830I VSWITCH SYSTEM IT71 status is devices attached.
HCPSWU2830I DTCVSW2 is VSWITCH controller.
15:02:05 HCPSWU2830I VSWITCH SYSTEM IT71 status is in error recovery.
HCPSWU2830I DTCVSW1 is new VSWITCH controller.
DTCVSW1 : 15:02:05 DTCOSD080I VSWITCH-OSD initializing:
DTCVSW1 : 15:02:05 DTCPRI385I  Device IT71C200DEV:
DTCVSW1 : 15:02:05 DTCPRI386I     Type: VSWITCH-OSD, Status: Not started
DTCVSW1 : 15:02:05 DTCPRI387I     Envelope queue size: 0
DTCVSW1 : 15:02:05 DTCPRI388I     Address: C200
DTCVSW1 : 15:02:05 DTCQDI001I QDIO device IT71C200DEV device number C202:
DTCVSW1 : 15:02:05 DTCQDI007I   Enabled for QDIO data transfers
DTCVSW1 : 15:02:05 DTCOSD238I ToOsd: IPv4 multicast support enabled for IT71C200DEV
DTCVSW1 : 15:02:05 DTCOSD319I ProcessSetArpCache: Supported for device IT71C200DEV
DTCVSW1 : 15:02:05 DTCOSD341I Obtained MAC address 00096B1AA569 for device IT71C200DEV
DTCVSW1 : 15:02:05 DTCOSD234I ToOsd: TCPIP host is set as the PRIMARY router for port   for IPv4
DTCVSW1 : 15:02:05 DTCOSD238I ToOsd: IPv6 multicast support enabled for IT71C200DEV
DTCVSW2 : 15:02:05 DTCOSD360I VSWITCH-OSD link added for IT710800DEV
15:02:05 HCPSWU2830I VSWITCH SYSTEM IT71 status is ready.
HCPSWU2830I DTCVSW1 is VSWITCH controller.
DTCVSW1 : 15:02:09 DTCOSD246I VSWITCH-OSD device IT71C200DEV: Assigned IPv4 address 192.168.71.93
DTCVSW1 : 15:02:10 DTCOSD246I VSWITCH-OSD device IT71C200DEV: Assigned IPv4 address 192.168.71.97
```

Recovering from a network adapter failure: Channel Bonding

Description

Ideally, if one network adapter fails, then another should transparently take over for it. Channel bonding is one way to provide this function. Channel bonding joins multiple network adapters together to make a single, logically bonded interface. If one adapter within the bond fails, a surviving adapter will take over. It requires the network adapters both be connected to the same LAN and be running in layer 2 mode (because it utilizes MAC addresses). If the adapters are not connected to the same LAN then another approach, such as the layer 3 VIPA technique with a dynamic routing protocol, should be used instead.

Channel bonding makes use of the Linux bonding driver. This driver is standard in both Red Hat RHEL 4 and SUSE SLES 9 and later distributions. To pick up the latest fixes, the minimum recommended levels of each are RHEL 4 update 5, and SLES 9 service pack 3 update 1 (2.6.5-7.283-s390x). The most recently available version of OSA firmware should be installed as well.

There are several operational modes available for channel bonding. The ones supported for Linux on System z are: active-backup, balance-xor, balance-rr, and broadcast.

Unlike a layer-3 approach for VIPA, configuring an active/backup network path arrangement via channel bonding is transparent to the LAN infrastructure. There is no need to do any special router configuration, or to use a dynamic routing protocol to handle path failover. However, in active-backup mode, only one OSA port is being actively used -- the other is not utilized until the primary port fails. The balance-xor and balance-rr modes enable load balancing across the bonded adapters, though these approaches typically requires some configuration of directly-connected external routers to group the appropriate ports together (depending on the router this may be called an etherchannel or a trunk group) and to define a transmit policy.

Which mode makes the most sense to use (active-backup or load balancing) depends on whether you are running in a single switch or multi-switch network environment. Load balancing may make the most sense in a single-switch environment, as it will provide both high availability and maximized bandwidth across all bonded OSA ports. In a multi-switch environment, the active-backup configuration is the sensible approach for maximizing availability.

The OSA ports being bonded can be either dedicated to a particular LPAR, or shared between LPARs. With regard to sharing of OSA ports and layer2/3 mode, there once was a restriction that while two LPARs could share an OSD chpid (OSA port) with one LPAR using the port in layer 3 mode and the other using it in layer 2 mode, they could not communicate with each other over that shared OSA port. That is no longer the case. As of OSA support delivered May 26, 2006, there is no longer such a restriction.

Channel bonding is particularly useful when Linux is deployed in native LPARs, rather than under z/VM. This is because when running under z/VM, network adapter resilience that is transparent to external routers can be obtained by simply defining redundant OSA ports for a VSWITCH, as described in the prior section. However, as of z/VM 5.2, VSWITCH doesn't permit balancing across multiple network cards. So, unlike the load-balancing channel bonding modes, the bandwidth available for a given VSWITCH is limited to that of a single OSA adapter (note that IBM has announced VSWITCH support for IEEE Standard 802.3ad Link Aggregation, which

will enable all OSA-Express2 features that are associated with a virtual switch to be grouped and used as a single ″fat pipe″, helping to increase bandwidth and provide near-seamless failover. It will be available with z/VM 5.3). If desired, channel bonding will also work with Linux guests running under z/VM. Indeed, that is how we tested it in our environment.

In addition, VLANs can be configured on top of an OSA channel bonded interface. The reference documentation has more information on considerations for this mode of operation.

For more information, including details on the various bonding modes and when each is most appropriate, see Documentation/networking/bonding.txt in the Linux source tree, or visit

`http://sourceforce.net/projects/bonding`

Preparation

Setting up channel bonding involves defining MAC addresses for each network interface, loading the bonding driver, and then connecting the interfaces to the bonding driver. For this test, devices from two OSA ports were directly attached to a single Linux guest (even though the Linux guest was running under z/VM, we were not going through a VSWITCH for these connections). The OSA device range C300 - C302 was used for eth1, and C400 - C402 for eth2 (eth0 was already in use). MAC addresses were chosen which echo that numbering. Figure 33 on page 177 shows our channel bonding configuration.

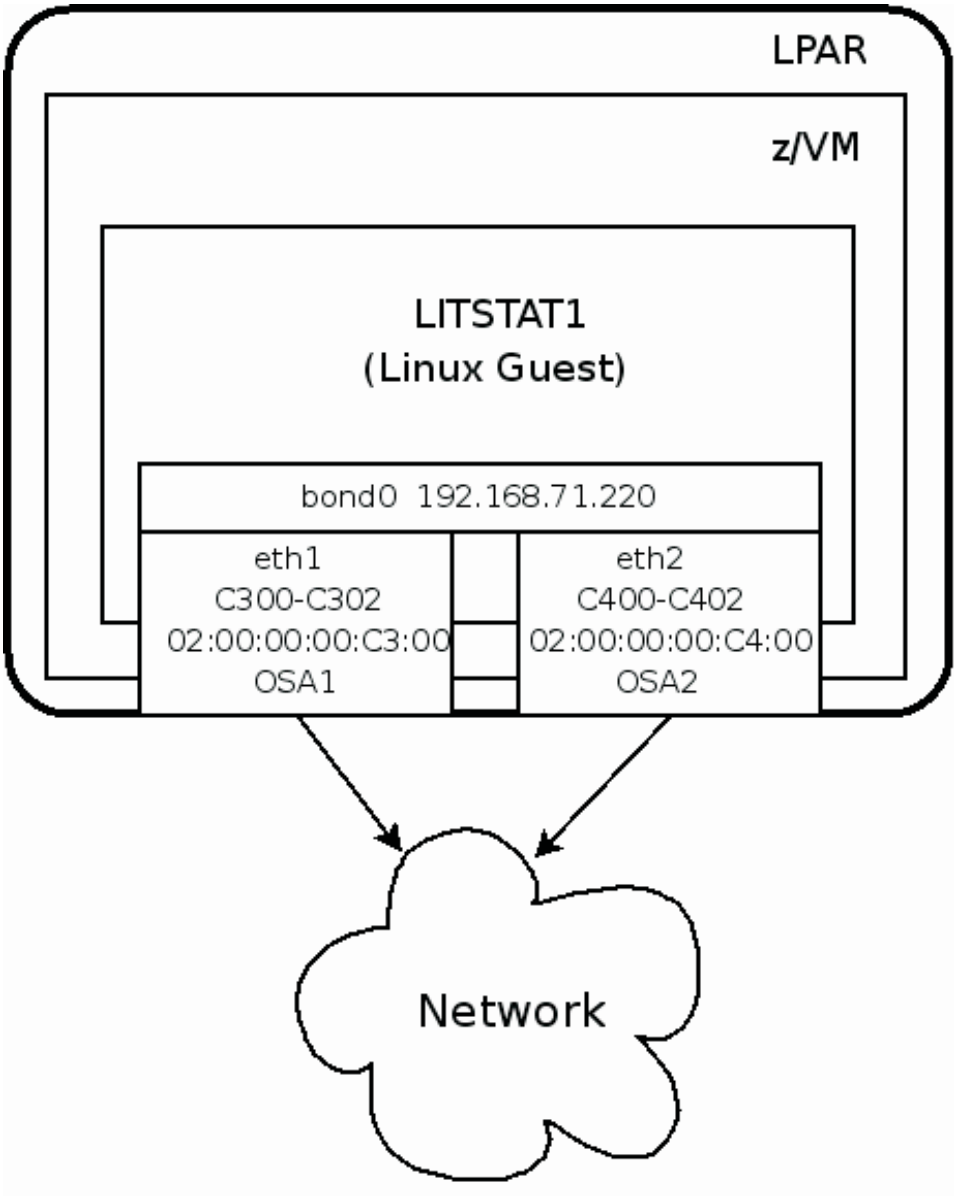


Figure 33. Configuration for OSA channel bonding

You can configure bonding dynamically on a live system through operator commands (in which case it will persist only until the system is rebooted), or permanently through the configuration files appropriate for your distribution. We'll look at examples of each from our environment.

Note: If this is the first time you've used a given OSA port from this LPAR, make sure each device is set to layer2 mode before bringing it online (see "Gotcha #2 – OSA Layer2 setup" on page 170 for more information):

```
# echo 1 >/sys/bus/ccwgroup/drivers/qeth/0.0.c300/layer2
# echo 1 >/sys/bus/ccwgroup/drivers/qeth/0.0.c400/layer2
```

Setup through operator commands: The steps below show how we initially set up OSA channel bonding in our environment.

Note: Before starting you may wish to clear the kernel message ring buffer, so that channel bonding messages from the kernel can be more readily viewed via the *dmesg* command. To do this, issue this command:

```
# dmesg –c
```

1. Added MAC addresses to eth1 and eth2:

   ```
   # ifconfig eth1 hw ether 02:00:00:00:c3:00
   # ifconfig eth2 hw ether 02:00:00:00:c4:00
   ```

2. Loaded the bonding module with the miimon option. The miimon option enables the detection of link failures. The value specified is the monitoring frequency, in milliseconds. The use_carrier option says to rely on the device driver to maintain its state with netif_carrier_on/off (which is the default):

   ```
   # modprobe bonding mode=active-backup miimon=100 use_carrier=1
   ```

   Another useful option that could have been be specified when loading the bonding driver is “primary=eth1” which would have told the bonding driver that whenever eth1 is up, make it the active interface. This would be useful, for example, if eth1 was a higher-throughput interface than eth2. That wasn’t the case for us, so we did not specify this option.

3. Brought up the bonding device bond0:

   ```
   # ifconfig bond0 192.168.71.220 netmask 255.255.255.0
   ```

4. Connected eth1 and eth2 to bond0. This completed the channel bonding configuration:

   ```
   # ifenslave bond0 eth1 eth2
   ```

5. Checked for any error messages issued by the bonding driver:

   ```
   # dmesg  OR  tail -f  /var/log/messages
   qeth: MAC address 02:00:00:00:c3:00 successfully registered on device eth1
   bonding: Warning: failed to get speed/duplex from eth1, speed forced to 100Mbps, duplex forced to Full.
   bonding: bond0: making interface eth1 the new active one.
   bonding: bond0: enslaving eth1 as an active interface with an up link.
   qeth: MAC address 02:00:00:00:c3:00 successfully registered on device eth2
   bonding: Warning: failed to get speed/duplex from eth2, speed forced to 100Mbps, duplex forced to Full.
   bonding: bond0: enslaving eth2 as a backup interface with an up link.
   eth1: no IPv6 routers present
   eth2: no IPv6 routers present
   ```

   The warning messages shown here are of no concern. The bonding module tries to use the ethtool package to get the speed and duplex mode of the OSA device, but ethtool isn’t supported for OSA so the module puts out this message. The ″speed forced to 100Mbps, duplex forced to Full″ portion of the message is also of no concern. It’s referring to some internal default values that get set for the bonding module’s processing – this has nothing to do with the actual speed provided through the interface and doesn’t affect its operation.

6. Displayed the resulting configuration. Note that the bond0, eth1 and eth2 interfaces all show the same MAC address (that of eth1 in this example):

   ```
   # ifconfig
   bond0     Link encap:Ethernet  HWaddr 02:00:00:00:C3:00
             inet addr:192.168.71.220  Bcast:192.168.71.255 Mask:255.255.255.0
             inet6 addr: fe80::200:ff:fe00:0/64 Scope:Link
             UP BROADCAST RUNNING MASTER MULTICAST  MTU:1500  Metric:1
             RX packets:21 errors:0 dropped:0 overruns:0 frame:0
             TX packets:13 errors:0 dropped:0 overruns:0 carrier:0
             collisions:0 txqueuelen:0
             RX bytes:1120 (1.0 Kb)  TX bytes:1378 (1.3 Kb)


   eth0      Link encap:Ethernet  HWaddr 02:00:00:00:00:17
             inet addr:192.168.71.149  Bcast:192.168.71.255 Mask:255.255.255.0
             inet6 addr: fe80::200:0:300:17/64 Scope:Link
             UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
             RX packets:2979 errors:0 dropped:0 overruns:0 frame:0
             TX packets:1723 errors:0 dropped:0 overruns:0 carrier:0
             collisions:0 txqueuelen:1000
             RX bytes:230859 (225.4 Kb)  TX bytes:255465 (249.4 Kb)
   ```

```
eth1      Link encap:Ethernet  HWaddr 02:00:00:00:C3:00
          inet addr:192.168.71.220  Bcast:192.168.71.255 Mask:255.255.255.0
          inet6 addr: fe80::200:0:0:c300/64 Scope:Link
          UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
          RX packets:10 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:526 (526.0 b)  TX bytes:726 (726.0 b)

eth2      Link encap:Ethernet  HWaddr 02:00:00:00:C3:00
          inet addr:192.168.71.220  Bcast:192.168.71.255 Mask:255.255.255.0
          inet6 addr: fe80::200:0:0:c300/64 Scope:Link
          UP BROADCAST RUNNING NOARP SLAVE MULTICAST  MTU:1500  Metric:1
          RX packets:11 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000

          RX bytes:594 (594.0 b)  TX bytes:652 (652.0 b)

# cat /proc/qeth

devices                   CHPID interface  cardtype       port chksum prio-q'ing rtr4 rtr6 fsz   cnt
------------------------- ----- ---------- -------------- ---- ------ ---------- ---- ---- ----- -----
0.0.0700/0.0.0701/0.0.0702 x00   eth0       GuestLAN QDIO  0    sw     always_q_2 no   no   64k   16
0.0.e000/0.0.e001/0.0.e002 xE0   hsi0       HiperSockets   0    sw     always_q_2 no   no   16k   16

0.0.c300/0.0.c301/0.0.c302 x26   eth1       OSD_1000       0    sw     always_q_2 no   no   64k   16
0.0.c400/0.0.c401/0.0.c402 x27   eth2       OSD_1000       0    sw     always_q_2 no   no   64k   16
```

7. Displayed bonding status information. This display shows the bonding mode (active/backup, load balancing, etc), the MII status, which interface is the currently active slave, each interface's defined MAC address, and other useful information:

```
# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v2.6.5 (November 4, 2005)

Bonding Mode: fault-tolerance (active-backup)
Primary Slave: None
Currently Active Slave: eth1
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0

Slave Interface: eth1
MII Status: up
Link Failure Count: 0
Permanent HW addr: 02:00:00:00:c3:00

Slave Interface: eth2
MII Status: up
Link Failure Count: 0
Permanent HW addr: 02:00:00:00:c4:00
```

8. Ensured the individual network adapters, or slaves, did not have TCP/IP routes of their own defined to the host. If such routes are defined, and they precede the route of the bond0 interface (the master), they must be removed or routing will get confused:

```
# route -n or netstat -nr
```

At this point the bond0 interface was ready for use.

Setup through the configuration file: Once we had completed initial testing with channel bonding, we wanted to permanently configure it in our environment so it would persist across reboots. Here are the steps we took for our SLES 9 system:

1. Created the appropriate hwcfg files. Here is an example for our C300-2 device:

```
/etc/sysconfig/hardware/hwcfg-qeth-bus-ccw-0.0.c300

#!/bin/sh
#
# hwcfg-qeth-bus-ccw-0.0.c300
```

```
#
# Hardware configuration for a qeth device at 0.0.c300
#

STARTMODE="auto"
MODULE="qeth_mod"
MODULE_OPTIONS=""
MODULE_UNLOAD="yes"


# Scripts to be called for the various events.
SCRIPTUP="hwup-ccw"
SCRIPTUP_ccw="hwup-ccw"
SCRIPTUP_ccwgroup="hwup-qeth"
SCRIPTDOWN="hwdown-ccw"

# CCW_CHAN_IDS sets the channel IDs for this device
# The first ID will be used as the group ID
CCW_CHAN_IDS="0.0.c300 0.0.c301 0.0.c302"

# CCW_CHAN_NUM set the number of channels for this device
# Always 3 for an qeth device
CCW_CHAN_NUM=3

# Enable qeth layer 2 support
QETH_LAYER2_SUPPORT=1
```

2. Created the appropriate ifcfg files for each OSA interface. Here is an example for our C300-2 device:

```
/etc/sysconfig/network/ifcfg-qeth-bus-ccw-0.0.c300

BOOTPROTO="static"
UNIQUE=""
STARTMODE="onboot"
_nm_name="qeth-bus-ccw-0.0.c300"
LLADDR="02:00:00:00:c3:00"
PERSISTENT_NAME="en1"
```

   Note the use of the PERSISTENT_NAME parameter is not required for bonding, we used it for the convenience of separating the names of interfaces being used for bonding from other interfaces (which use default names such as “eth0”). Without the use of this parameter, the default naming of our non-bonded interfaces on this Linux system changed during reboot because they started up in a different order (eth0 became eth2).

3. Created an ifcfg file for the bonding interface. Here’s what ours looked like:

```
/etc/sysconfig/network/ifcfg-bond0

BOOTPROTO="static"
STARTMODE="onboot"
IPADDR="192.168.71.220"
MTU="1500"
NETMASK="255.255.255.0"
NETWORK="192.168.71.0"
BROADCAST="192.168.71.255"
REMOTE_IPADDR=""
BONDING_MASTER="yes"
BONDING_MODULE_OPTS="mode=active-backup miimon=100 use_carrier=1"
BONDING_SLAVE0="qeth-bus-ccw-0.0.c300"
BONDING_SLAVE1="qeth-bus-ccw-0.0.c400"
```

That’s it. We then rebooted the system and issued the same status display commands as in the prior example to ensure everything had come up correctly. A workload was then started to drive HTTP traffic to an Apache Web server over the bonded interface.

Network adapter failure forced by pulling cable in the OSA port

With the setup done as described previously under “Setup via Operator Commands,” a failure was forced by pulling the cable into the OSA port for eth1 (chpid 26, virtual device C300).

Indicators and Failover: Messages from the bonding driver are sent into the kernel ring buffer, which ordinarily can be seen in the syslog, or by issuing the dmesg command. Here’s what we saw:

```
qeth: Link failure on eth1 (CHPID 0x26) - there is a network problem or someone
      pulled the cable or disabled the port.
bonding: bond0: link status definitely down for interface eth1, disabling it
bonding: bond0: making interface eth2 the new active one.
```

We issued various display commands. First, notice that /proc/net/bonding/bond0 shows the active slave is now eth2. The MII Status for eth1 is “down,” and its link failure count has increased to 1:

```
# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v2.6.5 (November 4, 2005)

Bonding Mode: fault-tolerance (active-backup)
Primary Slave: None
Currently Active Slave: eth2
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0

Slave Interface: eth1
MII Status: down
Link Failure Count: 1
Permanent HW addr: 02:00:00:00:c3:00

Slave Interface: eth2
MII Status: up
Link Failure Count: 0
Permanent HW addr: 02:00:00:00:c4:00
```

Then, notice that /proc/qeth is showing eth1 as +++ LAN OFFLINE +++

```
# cat /proc/qeth
devices                    CHPID interface  cardtype       port chksum prio-q'ing rtr4 rtr6 fsz   cnt
-------------------------- ----- ---------- -------------- ---- ------ ---------- ---- ---- ----- -----
0.0.0700/0.0.0701/0.0.0702 x00   eth0       GuestLAN QDIO  0    sw     always_q_2 no   no   64k   16
0.0.e000/0.0.e001/0.0.e002 xE0   hsi0       HiperSockets   0    sw     always_q_2 no   no   16k   16
0.0.c300/0.0.c301/0.0.c302 x26   eth1       OSD_1000       0
+++ LAN OFFLINE +++
0.0.c400/0.0.c401/0.0.c402 x27   eth2       OSD_1000       0    sw     always_q_2 no   no   64k   16
```

Failback Actions and Messages: We re-plugged the cable. Eth1 was recovered and put back into use as a backup by the bonding driver. Here’s what we saw when we issued *dmesg*:

```
qeth: Link reestablished on eth1 (CHPID 0x26). Scheduling IP address reset.
qeth: Recovery of device 0.0.c300 started ...
qeth: Device 0.0.c300/0.0.c301/0.0.c302 is a OSD Express card (level: 0631)
with link type OSD_1000 (portname: )
qeth: MAC address 02:00:00:00:c3:00 successfully registered on device eth1
bonding: bond0: link status definitely up for interface eth1.
qeth: Device 0.0.c300 successfully recovered!
```

We again issued various display commands. First, notice that the MII Status for eth1 is now “up,” but the currently active slave is still eth2. That is expected, because we did not specify a primary interface when loading the bonding driver.

```
# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v2.6.5 (November 4, 2005)

Bonding Mode: fault-tolerance (active-backup)
Primary Slave: None
Currently Active Slave: eth2
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0

Slave Interface: eth1
MII Status: up
Link Failure Count: 1
Permanent HW addr: 02:00:00:00:c3:00

Slave Interface: eth2
MII Status: up
Link Failure Count: 0
Permanent HW addr: 02:00:00:00:c4:00
```

Then, notice that eth1 is no longer shown by /proc/qeth to be +++ LAN OFFLINE +++

```
# cat /proc/qeth
devices                   CHPID interface  cardtype       port chksum prio-q'ing rtr4 rtr6 fsz   cnt
------------------------- ----- ---------- -------------- ---- ------ ---------- ---- ---- ----- -----
0.0.0700/0.0.0701/0.0.0702 x00   eth0       GuestLAN QDIO  0    sw     always_q_2 no   no   64k   16
0.0.e000/0.0.e001/0.0.e002 xE0   hsi0       HiperSockets   0    sw     always_q_2 no   no   16k   16
0.0.c300/0.0.c301/0.0.c302 x26   eth1       OSD_1000       0    sw     always_q_2 no   no   64k   16
0.0.c400/0.0.c401/0.0.c402 x27   eth2       OSD_1000       0    sw     always_q_2 no   no   64k   16
```

To complete the test we then repeated the process, this time pulling the cable for device eth2, and saw eth1 become the active interface as expected. Finally, we re-plugged the cable, and saw eth2 resume its role as the backup interface.

Recovering from a SCSI path failure

Description

Multipathing increases system stability and resilience. Multipathing tools can be used when SCSI disks are attached to achieve high availability and load balancing. We used LVM2 for multipathing with our FCP SCSI device - making one path fail and ensuring one of the others can be used to continue accessing the device without knowledge to the application.

We tested two types of failures – adapter failure and chipid failure. We discuss results from SUSE Enterprise Linux Server 10. But we performed the same tests on RedHat Enterprise Linux 4.4, and the results remained the same. Workload remained running without any errors, access to disk was uninterrupted.

Preparation

The following are the details of the SCSI disks defined to our test images:

Hostname: LITSMLTP (SUSE Enterprise Linux Server 10)

```
SCSI Addresses = a41a & a51a
SCSI LUNS = 571a (chpid 92) & 571b (chpid 93)
WWPN = 5005076300c1afc4 & 5005076300cdafc4
```

For details on implementing FCP on Linux/z, please reference this RedBook:

http://www.redbooks.ibm.com/abstracts/sg246344.html?Open

Enabling multipathing on our servers: Below are the steps we took to enable multipathing on our servers:

1. Loaded the multipath module

```
# modprobe dm-multipath
```

2. Ran multipath – to view what was recognized:

```
# multipath
...
1IBM_2105_71A29228IBM,2105800
[size=18G][features=1 queue_if_no_path][hwhandler=0]
\_ round-robin 0 [prio=0][active]
 \_ 0:0:0:0   sda   8:0   [active][undef]
 \_ 1:0:0:0   sdc   8:32  [active][undef]
1IBM_2105_71B29228IBM,2105800
[size=18G][features=1 queue_if_no_path][hwhandler=0]
\_ round-robin 0 [prio=0][active]
 \_ 0:0:0:1   sdb   8:16  [active][undef]
 \_ 1:0:0:1   sdd   8:48  [active][undef]
```

3. We next set up LVM2 to use the multipathed disks.

   Before we could access the multipathed devices, we had to make the following changes to */etc/lvm/lvm2.conf*, so that the devices can be accessed. In the example below, the first line indicates the original definition that was commented out, and the second line indicates the new definition:

```
##  filter = [ "r|/dev/.*/by-path/.*|", "r|/dev/.*/by-id/.*|", "a/.*/" ]
    filter = [ "r|/dev/.*/by-path/.*|", "r|/dev/.*/by-id/.*|", "a/.*/","a|/dev/dm.*|" ]
```

   The string ″a|/dev/dm.*|″ introduced those devices to LVM2.

4. Declared the disks as ″physical volume″ with pvcreate:

```
# pvcreate /dev/sda1 /dev/sdc1
```

5. Merged the just created physical volumes into our volume group (mfst_vg) with vgcreate:

```
# vgcreate mfst_vg /dev/sda1 /dev/sdc1
```

6. Created Logical Volume:

```
# lvcreate -L 18G -n mfst_lv mfst_vg
```

7. Created ext3 FileSystem:

```
# mkfs.ext3 -b 4096 /dev/mfst_vg/mfst_lv
```

8. To start the multipathing configuration, we issued:

```
# multipath
```

We used a workload called MFST (Mega File System Thrasher) to generate I/O to the target the SCSI multipathed devices.

CHPID failure

To simulate a CHIPID failure, we set a CHIPID (logically) offline from Linux so that disk access continues along the other CHIPID.

```
litsmltp:/sys/devices/css0/chp0.92 # echo offline > status
```

Indicators and Failover: The MFST workload remained running with the loss of one path to the device. We checked on the availability of the device:

```
litsmltp:/sys/devices/css0/0.0.0000/0.0.a41a # cat availability
good
```

Failback Actions and Messages: We enabled the CHPID back online and the workload remained running and no errors were thrown:

```
litsmltp:/sys/devices/css0/chp0.92 # echo online > status
```

Adapter failure

To simulate an adapter failure, we took the adapter offline:

```
litsmltp:/sys/devices/css0/0.0.0000/0.0.a41a # echo 0 > online
```

Indicators and Failover: Checked that the adapter is really offline:

```
litsmltp:/sys/devices/css0/0.0.0000/0.0.a41a # cat online
0
```

The MFST workload remained running with the loss of the adapter.

We checked on the availability of the device:

```
litsmltp:/sys/devices/css0/0.0.0000/0.0.a41a # cat availability
good
```

Failback Actions and Messages: We enabled the adapter back online and the workload remained running:

```
litsmltp:/sys/devices/css0/0.0.0000/0.0.a41a # echo 1 > online
litsmltp:/sys/devices/css0/0.0.0000/0.0.a41a # cat online
1
```

The above examples were results from SLES 10. We performed the same tests on RHEL 4.4, and the results remained the same. Workload remained running without any errors, access to disk was uninterrupted.

Recovering from a DASD failure

Description

One of our focus areas for this recovery test was to strengthen our infrastructure availability, and resilient DASD plays a key role in this area.

To setup for failure resilient DASD, we decided to take advantage of IBM's GDPS/PPRC (Geographically Dispersed Parallel Sysplex/Peer-to-Peer Remote Copy) Multiplatform Resiliency for zSeries offering. In this section, we will first provide a brief introduction to the technology, talk about how we used it in our testing, and finally discuss the failover activities.

Note: The GDPS family is comprised of a number of service offerings and the associated technical documentations are only available on a licensed basis. If you are interested in further exploring a potential deployment of GDPS/PPRC Multiplatform Resiliency for zSeries, please contact your IBM representative.

The following descriptions of GDPS and related technologies are taken from the IBM GDPS website. For a more comprehensive description of the complete offering, including sample continuous availability and disaster recovery scenarios, please refer to the RedBook SG-246374: GDPS Family – An Introduction to Concepts and Capabilities. The March 2007 draft edition of the book reflects the latest changes and new capabilities of GDPS 3.3 and GDPS 3.4, and is available at

`http://www.redbooks.ibm.com/redpieces/pdfs/sg246374.pdf.`

GDPS IBM Geographically Dispersed Parallel Sysplex (GDPS®) is a multi-site or single-site end to end application availability solution that provides the capability to manage remote copy configuration and storage subsystems (including IBM TotalStorage Enterprise Storage Server), to automate Parallel Sysplex operation tasks and perform failure recovery from a single point of control.

GDPS helps automate recovery procedures for planned and unplanned outages to provide near-continuous availability and disaster recovery capability.

HyperSwap

HyperSwap, a software technology that can seamlessly substitute Metro Mirror (peer-to-peer remote copy, or PPRC), secondary devices for PPRC primary devices, is managed through GDPS. As its name suggests, HyperSwap is designed to swap a large number of devices and to do it fast so there is minimal impact to application availability with disruptions measured in seconds rather than hours. The HyperSwap function may be performed even if the primary disk subsystem is not operational, so you can survive a primary disk subsystem—or even a complete site—failure without recycling your systems. In business terms, that means continuous access to data if such a failure were to occur.

GDPS/PPRC Multiplatform Resiliency

To reduce IT costs and complexity, many enterprises are consolidating open servers into Linux guests running under z/VM, or native zLinux systems. GDPS/PPRC Multiplatform Resilience is an offering to support these environments. Starting with GDPS/PPRC v3.1, the ability to manage Linux guests running on xDR was provided. The latest version, GDPS/PPRC v3.4, provides the ability to manage the Linux guests running on xDR (Extended Disaster Recovery) disks, as well as native zLinux systems running on xDR disks.

GDPS/PPRC is able to exploit HyperSwap to provide continuous availability for Linux guests and native zLinux systems across a planned or unplanned outage of the primary disk subsystems.

Preparation

Software and DASD used:

Software levels:

Table 12. Software levels for failure resilient DASD

Product	Maintenance Level
GDPS V3.3	All service through the end of December 2006 not already marked RSU. PE resolution and HIPER/Security/Integrity/Pervasive PTFs and their associated requisites and supersedes through the end of February 2007.
Tivoli Systems Automation for Multiplatforms V2.2	Fix Pack 1
z/VM V5.2	Service Level 0602 (64-bit) – this contains all the APARs needed for z/VM to do the HyperSwap operation.
SUSE LINUX Enterprise Server 9	Service Pack 3

xDR DASD: We used 2 Enterprise Storage Servers 800 (Shark). Microcode level is 2.4.4.37 (R2i.8b051205a).

Planning for GDPS/PPRC multiplatform resiliency: A single Logical Subsystem (LSS) on the xDR DASD was provided to us by the IBM Consolidated Service Test (CST) team. Only one z/VM LPAR can be defined to use an LSS.

In addition to the LSS, the CST team also had a fully functional GDPS v3.3 environment setup for z/OS. We were able to work with them to utilize their GDPS v3.3 to manage our Linux guests. Figure 34 shows our GDPS/PPRC Multiplatform Resiliency configuration.

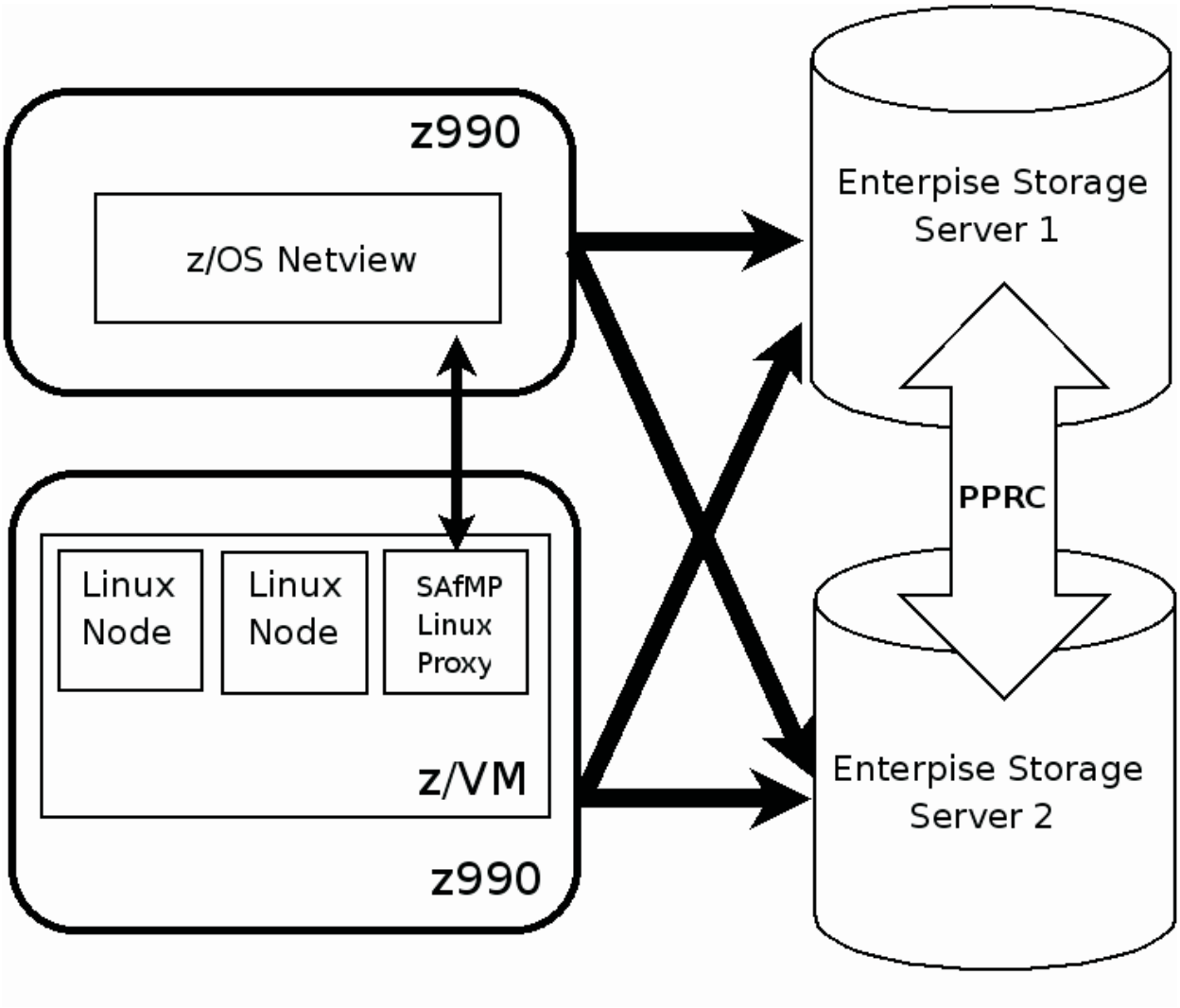


Figure 34. Configuration of GDPS/PPRC Multiplatform Resiliency

For full support information, including which versions of Linux and z/VM are supported, please refer to the licensed document: *GDPS/PPRC V3R3 Installation and Customization Guide (ZG24-6703-09)*.

Below is the information we needed in order to use CST's xDR DASD:

- Switch port numbers – our z990 system was already cabled into the xDR sharks, we needed the port numbers so that we could update the IODF's (I/O Definition) with the LSS (Logical Subsystem) information.
- LSS information for site1 DASD (normally primary) and site 2 DASD (normally secondary), for the IODF:
  - WWNN
  - SSID
  - LSS

- Serial Number
- The device numbers for site 1 and site 2

Depending on your setup, at least one Linux guest needs to communicate over TCP/IP with NetView on z/OS. In Figure 34 on page 186, the Linux proxy guest functions as the middleman communicating with NetView on z/OS as well as z/VM. Below is the information we needed so that we could communicate with CST's NetView:

- Hostname and IP address of the system that NetView is running on.
- Port number of the NetView Event Receiver.

Similarly, the CST team needed some information from us to define our systems to GDPS:

- z/VM System Identifier - as defined by SYSTEM_IDENTIFICATION in the SYSTEM CONFIG file. Instead of the default, we used the System_Identifier:

```
/* System_Identifier VMI  */
System_Identifier  2084  18B52A VMI
```

- Linux cluster name – as defined on the Linux proxy guest when you setup Tivoli Systems Automation for Multiplatforms (see the Configuration section below)
- CEC name and LPAR name for our z/VM

Note: In GDPS, every LSS needs at least one volume (called the utility volume) that is not mirrored and not used by any application (including the operating system). The CST team had defined a few volumes for this purpose, so in our string of disk, we could not use the first 8 volumes as they were not mirrored. We used only the 9th onwards.

GDPS/PPRC Multiplatform Resiliency provides the following support for Linux on System z guests:

- Recovering from a Linux on System z node failure
- Recovering from a Linux on System z cluster failure
- Disk subsystem maintenance
- Disk subsystem failure

We designed our test for Linux disk subsystem maintenance and disk subsystem failure, to verify that our DASD I/O workload runs continuously even after DASD has failed (where secondary DASD would be automatically HyperSwapped for usage). Please note that GDPS doesn't support SCSI DASD yet.

Configuration: To test the scenario, we needed to have one Linux guest on the xDR DASD that would run the DASD I/O workload while we simulated a DASD failure. We needed the Linux proxy guest, running under the same z/VM as the Linux guest but not installed on xDR DASD, to monitor the logs for DASD failure, communicate with GDPS and z/VM, and manage the HyperSwap and failover.

Instead of building a Linux system on the xDR DASD, we cloned the disks of an existing Linux guest, litdat01, to the primary disks on the xDR DASD. The built-in Shark PPRC function enabled the disks to be automatically mirrored to the secondary disks.

To utilize the PPRC volumes on our Linux guests we needed to change the MDISK statements we were using in the z/VM directory to DEDICATE statements with the VOLID parameter. See sample directory entry below. This was the only change required in z/VM to utilize the PPRC volumes.

```
USER LITDAT01 xxxxxxxx 1536M 2048M G
   INCLUDE LIN191
   CPU 0
   CPU 1
   IPL CMS PARM AUTOCR
   MACHINE ESA 3
   NICDEF 0920 TYPE QDIO LAN SYSTEM PRVV71L2
   MDISK 0200 FB-512 V-DISK 2048000 MR
******** DASD Prior to using PPRC Volumes ****
*  MDISK 0201 3390 DEVNO 5030 MR ALL ALL AL
*  MDISK 0202 3390 DEVNO 3624 MR ALL ALL AL
*  MDISK 0203 3390 DEVNO 3625 MR ALL ALL AL
*  MDISK 0204 3390 DEVNO 3312 MWV ALL ALL AL
********* DASD Using PPRC VOLUMES ************
   DEDICATE 0201 VOLID LX5030
   DEDICATE 0202 VOLID LX3624
   DEDICATE 0203 VOLID LX3625
   DEDICATE 0204 VOLID LX4E17
```

We followed the steps outlined in Appendix H of the licensed document: *GDPS/PPRC V3R3 Installation and Customization Guide (ZG24-6703-09)* to configure the Linux proxy guest in z/VM, to setup Tivoli Systems Automation for Multiplatforms on the proxy system, and to configure GDPS/PPRC Multiplatform Resiliency on the proxy system.

The CST team made changes to GDPS, such as GEOPLEX OPTIONS and GEOPARM policy, and those are also described in the licensed document.

One item worth noting is that, on the Linux proxy guest, for the DASD error reporting daemons, *erpd* and *sseprd*, you need to use the ones that come with Tivoli Systems Automation for Multiplatforms v2.2. This is to avoid a problem with *erpd* where it would consume a lot of CPU in a loop issuing these messages:

```
erpd: Could not read from file /dev/vmlogrdr; device=-1; errno=0010
```

This problem has been reported to the GDPS team and is currently being addressed by the GDPS team.

Verification: When everything is all set on the Linux proxy side (heartbeat and the DASD error reporting daemon are running), and all set on the GDPS side; on GDPS turn the GEOPLEX OPTION XDR to YES, so that the disks defined in the GEOPARM definitions will be managed, and GDPS will communicate with Tivoli Systems Automation for Multiplatforms to manage the Linux disks.

The primary xDR disks for our Linux guest were 4E10, 4E11, 4E12, and 4E17. The secondary xDR disks that were mirrored were 6E10, 6E11, 6E12, and 6E17.

To verify that your Linux guest is using the primary disks, issue a vmcp command from your Linux:

```
litdat01:~/mfst2003 # vmcp q dasd
DASD 0190 3390 VMIIPL R/O        107 CYL ON DASD  830B SUBCHANNEL = 000C
DASD 0191 3390 VM3017 R/O          2 CYL ON DASD  3017 SUBCHANNEL = 0010
DASD 0196 3390 VM3130 R/O       3338 CYL ON DASD  3130 SUBCHANNEL = 000F
DASD 019D 3390 VMIIPL R/O        160 CYL ON DASD  830B SUBCHANNEL = 000D
DASD 019E 3390 VMIIPL R/O        350 CYL ON DASD  830B SUBCHANNEL = 000E
DASD 0200 9336 (VDSK) R/W    2048000 BLK ON DASD  VDSK SUBCHANNEL = 0003
DASD 0201 ON DASD  4E10 R/W LX5030 SUBCHANNEL = 0004
DASD 0202 ON DASD  4E11 R/W LX3624 SUBCHANNEL = 0005
DASD 0203 ON DASD  4E12 R/W LX3625 SUBCHANNEL = 0006
DASD 0204 ON DASD  4E17 R/W XX4E17 SUBCHANNEL = 0007
```

Issuing the CP Q DASD from the z/VM guest will display also display this information.

To verify your installation, on the Linux proxy, check /var/log/messages for messages like the following (CSKEIP is the hostname of GDPS NetView). These messages indicate that the proxy is communicating with NetView, and GPDS is getting HyperSwap ready and monitoring the DASD.

You should see messages like these when the Linux proxy is communicating with GDPS:

```
Feb  6 10:52:21 litsprxy xdr: erpd: Sending EIF event to
GDPS (config file=/etc/Tivoli/tec/ksys1.conf, CSKEIP.xx.xx.xx@4429)
Feb  6 10:52:21 litsprxy xdr: erpd: Sending EIF event class XDR_EVENT_ERP_INIT:
Feb  6 10:52:21 litsprxy xdr: erpd: ====================================
Feb  6 10:52:21 litsprxy xdr: erpd: 0:CN=proxy_domain
Feb  6 10:52:21 litsprxy xdr: erpd: 1:NodeName=litsprxy
Feb  6 10:52:21 litsprxy xdr: erpd: 2:IP=x.xx.xx.187
Feb  6 10:52:21 litsprxy xdr: erpd: 3:UserID=LITSPRXY
Feb  6 10:52:21 litsprxy xdr: erpd: 4:Proxy=True
Feb  6 10:52:21 litsprxy xdr: erpd: 5:VERSION=1
Feb  6 10:52:21 litsprxy xdr: erpd: ====================================
Feb  6 10:52:21 litsprxy xdr: erpd: Event String: '
Feb  6 10:52:21 litsprxy xdr: erpd:
 XDR_EVENT_ERP_INIT;CN=proxy_domain;NodeName=litsprxy;IP=x.xx.xx.187;UserID=LITSPRXY;
Proxy=True;VERSION=1;END;
Feb  6 10:52:21 litsprxy xdr: erpd: '
Feb  6 10:52:21 litsprxy xdr: erpd: Sending EIF event to GDPS
(config file=/etc/Tivoli/tec/ksys1.conf, CSKEIP.xx.xx.xx@4429)
```

Here you see GDPS prepping for hyperswap:

```
initialization finished.
Feb  6 10:52:22 litsprxy in.rexecd[3435]: connect from x.xx.xx.95 (x.xx.xx.95)
Feb  6 10:52:22 litsprxy in.rexecd[3435]: connect from CSKEIP.xx.xx.xx
Feb  6 10:52:22 litsprxy in.rexecd[3435]: login from CSKEIP.xx.xx.xx as xdrusr
Feb  6 10:52:22 litsprxy logger: xdr.hyperswap: cluster name = proxy_domain
Feb  6 10:52:22 litsprxy logger: xdr.hyperswap: node name    = litsprxy
Feb  6 10:52:22 litsprxy logger: xdr.hyperswap: tcp/ip       = x.xx.xx.187
Feb  6 10:52:22 litsprxy logger: xdr.hyperswap: ignore comment: #devices of package 1
Feb  6 10:52:22 litsprxy logger: xdr.hyperswap: execute: hcp VARY ON 6e08-6e8f
Feb  6 10:52:22 litsprxy logger: xdr.hyperswap: ignore comment: #devices of package 1
Feb  6 10:52:22 litsprxy logger: xdr.hyperswap: execute: hcp hyperswap enable 4E08.136
Feb  6 10:52:22 litsprxy logger: xdr.hyperswap: calling sserpd to kill erpd
```

You should see messages like these, indicating that DASD logs are being monitored:

```
Feb  6 10:52:22 litsprxy xdr: erpd: Woken up from sleep mode
Feb  6 10:52:22 litsprxy xdr: erpd: read PPRC device file
Feb  6 10:52:22 litsprxy xdr: erpd: Read pprc triple, primary=4E08, secondary=6E08, RANGE=136
Feb  6 10:52:22 litsprxy kernel: vmlogrdr: recording command: RECORDING EREP PURGE  QID *
Feb  6 10:52:22 litsprxy sserpd[3465]: pid of erpd is 1929, successfully sent SIGUSR1
Feb  6 10:52:22 litsprxy kernel: vmlogrdr: recording response: Command complete
Feb  6 10:52:22 litsprxy kernel: vmlogrdr: recording command: RECORDING EREP ON  QID *
Feb  6 10:52:22 litsprxy kernel: vmlogrdr: recording response: Command complete
Feb  6 10:52:22 litsprxy logger: xdr.hyperswap: hyperswap command processing completed
```

Before the failure event, we started the DASD I/O workload on the Linux guest, litdat01, doing reads and writes to stress the xDR DASD.

Planned and unplanned DASD outages

The CST team conducts two types of outages per month which we were able to piggyback on.

Planned takeover

GDPS is a NetView/System Automations application and it provides a set of

verbs that an installation can use to write scripts. The CST team has a script that executes the appropriate verbs to perform a planned HyperSwap (which results in the dynamic flipping of the PPRC mirroring relationship; if the disk in site1 was primary and site2 was secondary, after a HyperSwap, the disk in site2 is now primary and site1 is secondary).

Unplanned outage

An unplanned HyperSwap is one where GDPS automatically performs the HyperSwap in response to an error condition (usually referred to as a trigger in the GDPS books). The CST team has several methods for triggering an unplanned HyperSwap, and the one they use most often is powering down the primary disk subsystem.

Indicators and Failover: On the Linux proxy, we saw the following messages in /var/log/messages after the failure:

```
Feb  6 10:55:50 litsprxy xdr: erpd: Detected OBRREC
Feb  6 10:55:50 litsprxy xdr: erpd: Analysing SNS data
Feb  6 10:55:50 litsprxy xdr: erpd: F0
Feb  6 10:55:50 litsprxy xdr: erpd: 1D detected
Feb  6 10:55:50 litsprxy xdr: erpd: not IEA491E (2)
```

Then we saw messages that indicated the HyperSwap activities:

```
Feb  6 10:55:50 litsprxy logger: xdr.hyperswap: execute: hcp hyperswap quiesce  4E08.136 6E08.136
Feb  6 10:56:01 litsprxy logger: xdr.hyperswap: execute: hcp hyperswap swap 4E08.136 6E08
Feb  6 10:56:02 litsprxy logger: xdr.hyperswap: execute: hcp hyperswap resume 4E08.136 6E08.136
Feb  6 10:57:21 litsprxy kernel: vmlogrdr: recording response: HCPCRC8058I User LITSPRXY has purged
00005556 records from the *L
Feb  6 10:58:05 litsprxy logger: xdr.hyperswap: execute: hcp hyperswap disable 6E08.136
Feb  6 10:58:05 litsprxy logger: xdr.hyperswap: execute: hcp VARY OFF 4e08-4e8f
Feb  6 10:58:05 litsprxy kernel: vmlogrdr: recording response: HCPCRC8058I User LITSPRXY has purged
00002532 records from the *LOGREC recording queue.
Feb  6 10:58:05 litsprxy logger: xdr.hyperswap: execute: hcp hyperswap disable 6E08.136
Feb  6 10:58:05 litsprxy logger: xdr.hyperswap: execute: hcp VARY OFF 4e08-4e8f
Feb  6 10:58:06 litsprxy logger: xdr.hyperswap: execute: hcp hyperswap disable 6E08.136
Feb  6 10:58:06 litsprxy logger: xdr.hyperswap: execute: hcp VARY OFF 4e08-4e8f
Feb  6 11:00:20 litsprxy logger: xdr.hyperswap: execute: hcp VARY ON 4e08-4e8f
Feb  6 11:00:20 litsprxy logger: xdr.hyperswap: execute: hcp hyperswap enable 6E08.136
```

You will see messages interspersed throughout indicating that the HyperSwap commands have completed successfully:

```
Feb  6 10:56:01 litsprxy logger: xdr.hyperswap: hyperswap command processing completed
```

On the Linux system, litdat01, we queried the DASD to verify they switched over to the secondary DASD:

```
litdat01:~/mfst2003 # vmcp q dasd
...
DASD 0201 ON DASD  6E10 R/W LX5030 SUBCHANNEL = 0004
DASD 0202 ON DASD  6E11 R/W LX3624 SUBCHANNEL = 0005
DASD 0203 ON DASD  6E12 R/W LX3625 SUBCHANNEL = 0006
DASD 0204 ON DASD  6E17 R/W 0X0204 SUBCHANNEL = 0007
```

The workload continued to run without any interruptions and no errors were reported.

Failback Actions and Messages: The CST team recovered the site 1 DASD. We did not have to do anything on z/VM or the Linux proxy.

The Linux proxy logs were quite extensive; we have highlighted some of the key messages that occurred in /var/log/messages during the failback:

```
Feb  6 11:16:05 litsprxy logger: xdr.hyperswap: execute: hcp hyperswap quiesce  4E08.136 6E08.136
Feb  6 11:16:17 litsprxy logger: xdr.hyperswap: execute: hcp hyperswap swap 6E08.136 4E08
Feb  6 11:16:17 litsprxy logger: xdr.hyperswap: execute: hcp hyperswap resume 4E08.136 6E08.136
Feb  6 11:17:22 litsprxy kernel: vmlogrdr: recording response: HCPCRC8058I User LITSPRXY has purged
```

```
00004590 records from the *LOGREC recording queue.
Feb  6 11:17:49 litsprxy logger: xdr.hyperswap: execute: hcp hyperswap disable 4E08.136
Feb  6 11:17:49 litsprxy logger: xdr.hyperswap: execute: hcp VARY OFF 6e08-6e8f
Feb  6 11:17:51 litsprxy kernel: vmlogrdr: recording response: HCPCRC8058I User LITSPRXY has purged
00001900 records from the *LOGREC recording queue.
Feb  6 11:17:51 litsprxy logger: xdr.hyperswap: execute: hcp hyperswap disable 4E08.136
Feb  6 11:17:52 litsprxy logger: xdr.hyperswap: execute: hcp VARY OFF 6e08-6e8f
Feb  6 11:17:53 litsprxy logger: xdr.hyperswap: execute: hcp hyperswap disable 4E08.136
Feb  6 11:17:53 litsprxy logger: xdr.hyperswap: execute: hcp VARY OFF 6e08-6e8f
Feb  6 11:18:49 litsprxy logger: xdr.hyperswap: execute: hcp hyperswap disable 4E08.136
Feb  6 11:18:49 litsprxy logger: xdr.hyperswap: execute: hcp VARY OFF 6e08-6e8f
Feb  6 11:20:07 litsprxy logger: xdr.hyperswap: execute: hcp VARY ON 6e08-6e8f
Feb  6 11:20:08 litsprxy logger: xdr.hyperswap: execute: hcp hyperswap enable 4E08.136
```

On the Linux system, we queried the DASD to verify that the system switched back to the primary DASD:

```
litdat01:~/mfst2003 # vmcp q dasd
...
DASD 0201 ON DASD  4E10 R/W LX5030 SUBCHANNEL = 0004
DASD 0202 ON DASD  4E11 R/W LX3624 SUBCHANNEL = 0005
DASD 0203 ON DASD  4E12 R/W LX3625 SUBCHANNEL = 0006
DASD 0204 ON DASD  4E17 R/W XX4E17 SUBCHANNEL = 0007
```

The workload continued to run without any errors.

GDPS/PPRC Gotcha's

erpd and sserpd: On the Linux proxy guest, for the erpd and sseprd daemons, you need to use the ones that come with Tivoli Systems Automation for Multiplatforms v2.2, in order to avoid a problem with erpd where it would consume a lot of CPU in a loop issuing these messages "erpd: Could not read from file */dev/vmlogrdr; device=-1; errno=0010".* This problem has been reported to the GDPS team and is currently being addressed by the GDPS team.

PAV (Parallel Access Volumes): When we initially setup our environment to use PPRC, we copied the selected set of DASD using DDR copying everything to the targeted PPRC volumes. When the copy completed we varied offline the original volumes preventing duplicate volume labels in z/VM. However, we failed to realize that the original volumes had PAV definitions in the IODF. We only took the BASE addresses offline. This created a number of problems for GDPS/PPRC now that z/VM was able to see the PAV devices and the Linux guest was unable to use them. This triggered a failure to our surprise. Once we varied off the PAV volumes, PPRC worked perfectly.

Recovering from a Hardware Crypto failure

Description

System z Peripheral Component Interconnect (PCI) cryptographic cards can accelerate asymmetric cryptographic operations for Linux on System z. These PCI cryptographic cards (PCI Cryptographic Coprocessor, PCI Cryptographic Accelerator, PCI Extended Cryptographic Coprocessor, Crypto Express2) are commonly used to provide leading-edge performance of the complex Rivest-Shamir-Adelman (RSA) cryptographic operations used in the SSL protocol. Six cryptographic cards were made available to our VM LPAR on the z990. As the administrator user in z/VM, the following CP command was used to display the six cards:

```
CP Q CRYPTO AP
AP 00 PCICA Queue 08 is installed
AP 01 PCICA Queue 08 is installed
AP 02 CEX2C Queue 08 is superseded by PCICA
```

```
AP 03 CEX2C Queue 08 is superseded by PCICA
AP 04 PCIXCC Queue 08 is superseded by PCICA
AP 05 PCIXCC Queue 08 is superseded by PCICA
```

All of our Linux guests had access to the cards via their user directory profiles that contain the CRYPTO APVIRT statement. APVIRT allows sharing of the crypto cards so that all the guests can use them for cryptographic acceleration.

If there are multiple adjunct processors (AP) types available to z/VM, like in our z/VM, z/VM chooses the “best” AP type for acceleration to show to the Linux guest. The prioritized list is:

1. CEX2A (Crypto Express2 Accelerator mode)
2. PCICA (PCI Cryptographic Accelerator)
3. CEX2C (Crypto Express2 Coprocessor mode)
4. PCIXCC (Extended Cryptographic Coprocessor)
5. PCICC (PCI Cryptographic Coprocessor)

In our case, z/VM “selected” the PCICA card for the guests, as indicated by the “superseded by” statements of the other cards in the above sample output. Once a type is selected, z/VM will route crypto requests from the guest to however many queues/cards of that type are available. In our environment, z/VM load balanced requests among the two PCICA cards that were available.

We configured our Apache web servers to use the available hardware cryptography for Secure Sockets Layer (SSL) handshake acceleration. Apache uses openSSL which includes the ibmca engine that provides openSSL applications access to IBM crypto hardware via libICA. libICA is the IBM eServer Cryptographic Accelerator Device Interface Library for Linux and provides a generalized abstraction to the ICA device driver interface. Linux uses the z90crypt module as the generic character device driver for a cryptographic device. See Figure 35 for how everything works together.

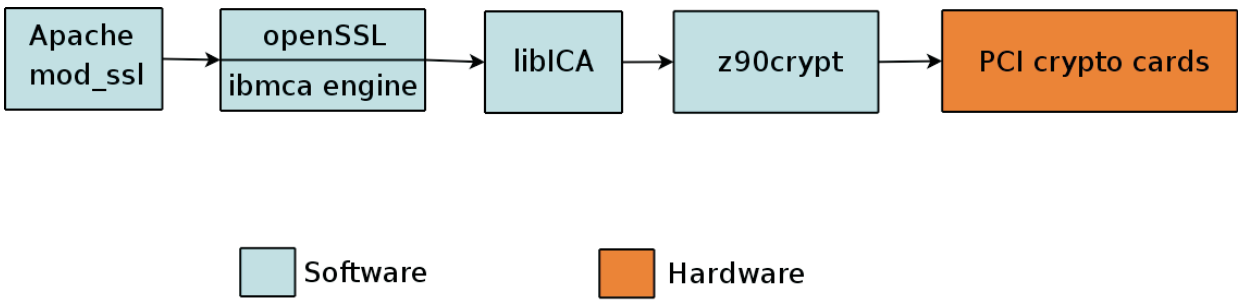


Figure 35. Apache using PCI crypto cards

For the failover test, we verified that our workload ran uninterrupted after one of the PCICA cards that it was using for cryptographic acceleration had failed. We simulated the failure by varying the CHIPID offline on the Service Element. We then varied the second PCICA card offline, and the workload started using software for the SSL handshakes.

For the failback/recovery test, we varied both PCICA cards online and verified that the workload started to use them again without interruptions.

Preparation

As mentioned in the description above, all of our Linux guests had access to the cryptographic cards via their user directory profiles that contain the CRYPTO APVIRT statement. On the z990 z/VM LPAR we had two Apache servers running on Linux, litstat1 and litstat2. By logging onto the guest VM console, you can query which virtual crypto AP and queue numbers z/VM has assigned to the guest by issuing CP command Q V CRYPTO.

```
CP Q V CRYPTO
No CAM or DAC Crypto Facilities defined
AP 15 PCICA Queue 04 shared
```

Alternately, if you have the Linux module vmcp installed, you can see the crypto information from the Linux command line by issuing the following:

```
# vmcp q crypto
No CAM or DAC Crypto Facilities defined
AP 15 PCICA Queue 04 shared
```

Our Apache servers were version 2.0.49 running on SUSE LINUX Enterprise Server 9 Service Pack 3.

We used the following steps to configure the Apache servers to use the available PCICA crypto cards:

1. Since we had a 32-bit version of Apache running on 64-bit system, we needed to install the following openSSL packages that came with the distribution:
   - openssl-32bit-9-200511222035
   - openssl-0.9.7d-15.21
2. We also installed the following versions of libICA, that also came with the distro:
   - libica-32bit-9-200511222035
   - libica-1.3.6rc2-0.5
3. Support for PCI Crypto cards on Linux for System z has been dependent on levels of z90crypt and z/VM. Refer to *Linux on zSeries Device Drivers, Features, and Commands* -

   `http://download.boulder.ibm.com/ibmdl/pub/software/dw/linux390/docu/l26bdd02.pdf`

   for more information on what is supported and where.

   For our distribution, SUSE LINUX Enterprise Server 9 SP3, the level of z90crypt is 1.3.3 and it has support for all the crypto cards: CEX2A, CEX2C, PCIXCC, PCICA and PCICC. Our z/VM version, 5.2, also supports the above crypto cards.
4. To load the z90crypt module, we used the z90crypt init script provided by the libica package:

   ```
   # rcz90crypt start
   Loading z90crypt module done
   ```
5. Now you can see details of the card:

   ```
   # cat /proc/driver/z90crypt

   z90crypt version: 1.3.3
   Cryptographic domain: 12
   Total device count: 1
   PCICA count: 1
   PCICC count: 0
   PCIXCC MCL2 count: 0
   PCIXCC MCL3 count: 0
   CEX2C count: 0
   ```

```
CEX2A count: 0
requestq count: 0
pendingq count: 0
Total open handles: 3


Online devices: 1=PCICA 2=PCICC 3=PCIXCC(MCL2) 4=PCIXCC(MCL3) 5=CEX2C 6=CEX2A
           0000000001000000 0000000000000000 0000000000000000 0000000000000000


Waiting work element counts
           0000000000000000 0000000000000000 0000000000000000 0000000000000000


Per-device successfully completed request counts
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 000014D3 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

6. The number of open handles in the above step indicates how many SSL sessions are active. Every SLES9 SP3 SSH session open a handle. Apache that's been configured to use hardware crypto, that's running with SSL, also opens a handle.
7. Next, we updated Apache's ssl.conf file to use ibmca by adding the following:

   ```
   SSLCryptoDevice ibmca
   ```

   and updated the following parameter to indicate the suite of encryption ciphers to use with the crypto device:

   ```
   SSLCipherSuite AES128-SHA
   ```
8. We generated self-signed certificates using openSSL and defined those in Apache's ssl.conf as well:

   ```
   SSLCertificateFile /usr/local/apache2.0.49_susessl7c/keys/litstat2.crt
   SSLCertificateKeyFile /usr/local/apache2.0.49_susessl7c/keys/litstat2.key
   ```
9. We ensured the server name was correctly listed in ssl.conf:

   ```
   ServerName litstat2.ltic.pok.ibm.com:443
   ```
10. Started Apache with SSL:

    ```
    litstat2:/usr/local/apache2.0.49_susessl7c/bin # ./apachectl startssl
    ```
11. 11. After it had started, we did another cat /proc/drivers/z90crypt command to see that the number of open handles had increased by one.
12. To verify the crypto cards were being used, we conducted a https request to the Apache server and did another cat /proc/drivers/z90crypt to see that the counter incremented:

    ```
    Per-device successfully completed request count
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
        00000000 000014D4 00000000 00000000 00000000 00000000 00000000 00000000
        00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    ```

Now, we were ready to conduct the failover test.

Simulate crypto failure by varying crypto CHIPID offline

With the BookStore workload running, generating SSL handshake with every request, we varied one of the PCICA CHIPIDS offline via the Service Element. The CHIPID immediately went to "Stopped" state and turned red.

Indicators and Failover: The workload was still going, completely uninterrupted. We checked the counter with cat /proc/drivers/z90crypt periodically and noticed that it was still steadily increasing. At this point the other PCICA card was being used.

As the z/VM administrator, a query crypto indicated that one of the PCICA cards was in "deconfigured" state:

```
CP Q CRYPTO AP
AP 00 PCICA Queue 08 is installed
AP 01 PCICA Queue 08 is deconfigured
AP 02 CEX2C Queue 08 is superseded by PCICA
AP 03 CEX2C Queue 08 is superseded by PCICA
AP 04 PCIXCC Queue 08 is superseded by PCICA
AP 05 PCIXCC Queue 08 is superseded by PCICA
```

Varying the second crypto CHIPID offline: We then varied the second, and our last, PCICA CHIPID, offline.

Indicators and Failover: The workload still continued to run, although its performance dropped drastically, because now it was using software to handle every SSL handshake. We checked the counter with cat /proc/drivers/z90crypt and noticed that the counter has stopped incrementing.

At this point, as the z/VM administrator, a query crypto indicated that both PCICA cards were in "deconfigured" state:

```
CP Q CRYPTO AP
AP 00 PCICA Queue 08 is deconfigured
AP 01 PCICA Queue 08 is deconfigured
AP 02 CEX2C Queue 08 is superseded by PCICA
AP 03 CEX2C Queue 08 is superseded by PCICA
AP 04 PCIXCC Queue 08 is superseded by PCICA
AP 05 PCIXCC Queue 08 is superseded by PCICA
```

Failback Actions and Messages: Through the Service Element, we varied both cards back online. The workload's performance immediately picked up again, and another cat /proc/drivers/z90crypt indicated that Apache was indeed using the PCICA cards because the counter started to increment again.

At this point, as the z/VM administrator, a query crypto indicated that one of the PCICA cards were back to "installed" state:

```
CP Q CRYPTO AP
AP 00 PCICA Queue 08 is installed
AP 01 PCICA Queue 08 is deconfigured
AP 02 CEX2C Queue 08 is superseded by PCICA
AP 03 CEX2C Queue 08 is superseded by PCICA
AP 04 PCIXCC Queue 08 is superseded by PCICA
AP 05 PCIXCC Queue 08 is superseded by PCICA
```

VM console failure

Description

When running Linux in a z/VM environment, you may find at times you need to access an active Linux guest from a 3270 console. We were interested is seeing what happens to the running Linux system when connectivity to the console is lost. This could occur from a number of conditions, failure of the emulation program used to access the 3270 console, a network outage (TCP/IP, SNA) or a failure of the communications controller used by z/VM.

Preparation

We started a TN3270 session and logged on to a running Linux guest on z/VM. Logged on to a Linux user (root in this case) from the TN3270 emulated session. From the root user, started an application. We used TOP for our tests.

TN3270 emulation program failure

We closed the TN3270 emulator window to simulate a failure of the TN3270 emulation program. Connectivity to the Linux Guest was lost.

Indicators and Failover: On the z/VM Operator Console a message was displayed that the user was disconnected from the console.

```
10:23:56 GRAF L0003 DISCONNECT LTIC0002 USERS = 48    FORCED BY SYSTEM
```

The Linux guest continued to run in disconnected state. All programs that were running from the TN3270 when it was lost continued to run. We have verified this with the w command from a ssh session to the same Linux guest. You can see that the ttyS0 session is still active running the top program.

```
LTIC0002:~ # w
 11:28:30 up 11 days, 21:41,  2 users,  load average: 0.00, 0.00, 0.00
USER     TTY        LOGIN@   IDLE   JCPU   PCPU WHAT
root     ttyS0     11:12    4:38   0.41s  0.31s top
root     pts/0     11:28    0.00s  0.07s  0.00s w
```

Failback Actions and Messages: Determine the cause of console failure.

We restarted the TN3270 emulation session and logged on to the z/VM guest. The following message was displayed on the z/VM operator console.

```
10:31:07 GRAF L0005 RECONNECT  LTIC0002 USERS = 48    FROM 10.1.2.99
```

Once the TN3270 session is re-established, the Linux guest will be in the same state as when the failure occurred. In our case, the top command was still running as it nothing happened.

Notes:

1. If SET RUN ON is in the PROFILE EXEC for the guest, when the TN3270 session is re-established the guest will be RUNNING. If SET RUN ON was not set, the default is OFF, the guest will be in CP READ when the session is re-established and you will need to enter B for begin to start the guest running again. We recommend placing SET RUN ON in the PROFILE EXEC for each Linux guest.
2. In our environment, the FEATURES Disconnect_timeout statement in the SYSTEM CONFIG to set OFF. This disables the automatic logoff of the virtual machine that is forcibly disconnected. See the *z/VM V5R2.0 CP Planning and Administration Guide* at:

   http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/HCSGOB11/CCONTENTS?SHELF=HCSH2A80&DN=SC24-6083-03&DT=20060516094207

   for more information on the SYSTEM CONFIG FEATURES statement.

   Example:

   ```
   SYSTEM CONFIG
   Features Disconnect_timeout OFF   /* Disable auto logoff of virt    */
                       /* machines forcibly disconnected */
   ```

Recovery from a software server failure

In this section, we discuss what happens when software servers fail.

For z/VM, we discuss RACF and TCP/IP failures.

For Linux, we discuss what happens when the Linux systems fail while running the following software products that have been configured with high availability: WebSphere Application Server Network Deployment Edge Component Load Balancer, TAMe WebSEAL, LDAP, LVS Directors, Apache, and WebSphere Application Server.

Recovering from a RACF failure

Description

We were interested in finding out what effect a RACF failure would have on a running Linux guest. Requests to the RACF security manager are generally made only when the Linux guest is logged on. These requests include: authorization to link DASD devices, access to SPOOL and PRINT, VSWITCH authorization if controlled by RACF. Generally Linux guests will not make any additional request to the RACF security Manager once the guest has been logged on. However, a small window does exist while the RACF security manager is down, if there are requests for:

1. Linking or defining additional devices. If the Linux guest attempts to link additional DASD or define new z/VM devices while the security manager is down the request will fail and the following message will appear on the Linux 3270 console:

   ```
   HCPLNM6525E The External Security Manager is unavailable.
   ```

2. Logging onto a Linux z/VM guest from the 3270 console. If an attempt is made to logon to the z/VM Linux guest while RACF is down it will fail with the following message:

   ```
   HCPLNM6525E The External Security Manager is unavailable.
   ```

Once RACF is back online these requests will be successful.

Preparation

We set up z/VM to use RACF as the External Security Manager.

Using the CP FORCE RACFVM command to simulate a RACF failure

While Linux systems were running on z/VM, we forced the RACF service machine running with the CP FORCE RACFVM command to simulate a RACF failure.

Indicators and Failover: On the z/VM Operator Console a message was displayed that RACF was no longer communicating with CP.

```
15:13:33 HCPRPI036E CP/RACF communication path broken to RACFVM
```

There were no messages reported by the Linux system that RACF had failed. The Linux guests all continued to run normally.

Failback Actions and Messages: Determine the cause of the RACF failure and restart the RACF external security manager. Once RACF had completed initialization all external security manager issues returned to full function.

Recovering from a z/VM TCP/IP failure

Description

Our z/VM system has three TCP/IP stacks running. The main stack, named TCPIP, is used for communication with the z/VM system. This is the stack used when using FTP, TELNET (TN3270) and so on, to or from a non-Linux guest or z/VM itself. The additional two stacks, DTCVSW1 and DTCVSW2, are used for the VSWITCH controllers. In this section, we are only referencing what happens when the main TCPIP stack fails. We covered the failure of the VSWITCH controller stacks in “Loss of a VSWITCH controller” on page 172.

Preparation

We had configured z/VM to used TCPIP for TELNET(TN3270), FTP, and so on. Our Linux guests run independently of the main TCPIP stack. They use network devices defined to a VSWITCH or have dedicate OSA devices. This completely isolates the Linux guests from the main TCPIP stack running on z/VM.

Using the CP FORCE TCPIP command to simulate a TCP/IP failure

From the operator console on z/VM we forced the TCPIP stack, *CP FORCE TCPIP*. We issued this from the Operating System Messages interface on the HMC’s daily panel so that we would not loose connectivity to the Operator console. If we were using a traditional TCPIP emulator our connection would have dropped when we forced TCPIP.

Indicators and Failover: The z/VM Operator console displayed the following messages indicating that TCP/IP terminated.

All Linux systems continued to run and communications with the guests continued as normal with one exception. Any guest that was logged on through a TELNET TN3270 session was dropped. However, as we discussed above in the “VM Console Failure” section, once connectivity to the TN3270 session was re-established all work-in-progress was be in the same state it was when the session was dropped.

```
12:56:29 USER DSC   LOGOFF AS  TCPIP    USERS = 44    FORCED BY OPERATOR
PORTMAP : TCPIP    severed IUCV path.
PORTMAP : svc_run: - select failed: IUCV error (EIBMIUCVERR)
PORTMAP : run_svc returned unexpectedly
....
```

Failback Actions and Messages: Determine the cause of the TCPIP failure and restart TCPIP stack.

To bring it back we used a 3270 console that did not need TCPIP to connect to z/VM. The Operating System Messages interface on the HMC’s daily panel is one access point available. Others include a SNA attached terminal, PVM session or the OSA-Express Integrated Console Controller.

The following message appeared on the operator’s console once TCPIP initialization had completed:

```
TCPIP   : 12:56:46 DTCIPI023I TCP-IP initialization complete.
```

Recovering from a Load Balancer failure

Description

In order to make Tivoli Access Manager for e-business WebSEAL highly available, we needed to front end WebSEAL with WebSphere Application Server Network Deployment Edge Component Load Balancer. WebSEAL needed to communicate with LDAP to determine authorization and authentication for the web applications, so LDAP needed to be highly available as well. In the end, we had two Load Balancer servers, two WebSEAL servers, and two LDAP servers. These clustered nodes were split across two different z/VM LPARS on two different CECs. Figure 36 depicts the relationship between these components in our environment:

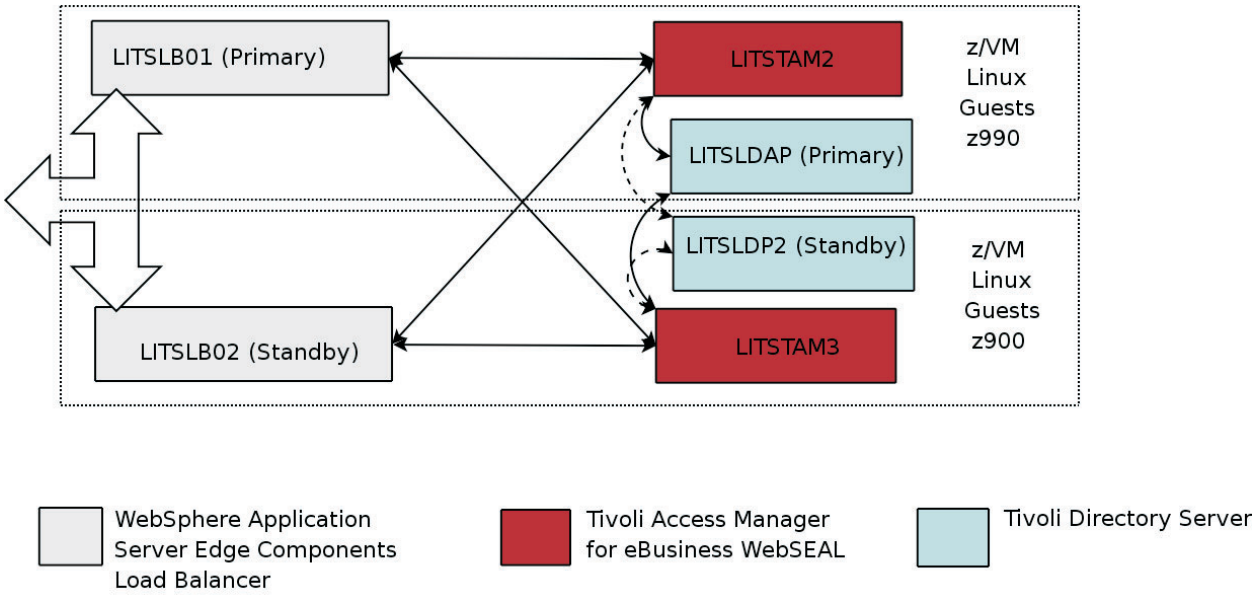


Figure 36. Highly available Load Balancer, WebSEAL, and LDAP

In this section, we discuss the updated Load Balancer configuration and what happens when a Load Balancer fails. In the next two related sections, “Recovering from WebSEAL failure” and “Recovering from LDAP failure”, we discuss what happens when WebSEAL fails and when LDAP fails.

Preparation

If you are familiar with our last test report, you will remember that Load Balancer was used to make Apache highly available. This time, we used the same two Load Balancers but updated them to make WebSEAL highly available.

In our previous test report we detailed what was needed to make a highly available Load Balancer cluster. To improve upon that, we first moved one Load Balancer from our pair onto a separate z/VM located on another CEC as indicated by Figure 36 above.

Due to our virtual networking infrastructure, we had to revise our forwarding method on the Load Balancer from MAC to NAT forwarding. Here, we discuss our new configurations.

Setting up the primary dispatcher: On the primary dispatcher – LITSLB02, we setup the following script to load the configurations, including commands to setup NAT forwarding. The comments start with "#" and explain the purpose of each section:

```
# start the server and executor and set log levels
dsserver
dscontrol executor start
dscontrol executor set clientgateway 192.168.74.100
dscontrol set loglevel 1

# add cluster address
dscontrol cluster add 192.168.74.150
dscontrol cluster set 192.168.74.150 proportions 49 50 1 0
dscontrol executor configure 192.168.74.150
dscontrol executor configure 192.168.74.152

# setup port number information on cluster
dscontrol port add 192.168.74.150:80 method nat reset no
dscontrol port add 192.168.74.150:443 method nat reset no

# add WebSEAL servers to the cluster
dscontrol server add 192.168.74.150:80:192.168.74.112 router 192.168.74.112 returnaddress 192.168.74.152
dscontrol server add 192.168.74.150:443:192.168.74.112 router 192.168.74.112 returnaddress 192.168.74.152
dscontrol server add 192.168.74.150:80:192.168.74.113 router 192.168.74.113 returnaddress 192.168.74.152
dscontrol server add 192.168.74.150:443:192.168.74.113 router 192.168.74.113 returnaddress 192.168.74.152

# start the manager and advisor
dscontrol manager start manager.log 10004
dscontrol advisor start Http 80 Http_80.log
dscontrol advisor start ssl 443 ssl_443.log
dscontrol advisor connecttimeout ssl 443 9
dscontrol advisor receivetimeout ssl 443 9

# setup heartbeat for HA
# first IP is litslb02, second IP is litslb01
dscontrol highavailability heartbeat add 192.168.74.135 192.168.74.99

# number of seconds that the executor uses to timeout HA heartbeats
dscontrol executor set hatimeout 3

# see if the dispatcher has network availability
dscontrol highavailability reach add 192.168.74.251

# set this dispatcher as the primary dispatcher
dscontrol highavailability backup add primary auto 9123

# print the status to screen
dscontrol highavailability status
```

Setting up the backup dispatcher: On the backup dispatcher – LITSLB01, we setup the following script to load the configurations, including commands to setup NAT forwarding. The comments start with "#" and explain the purpose of each section:

```
# start the server and executor and set log levels
dsserver
dscontrol executor start
dscontrol executor set clientgateway 192.168.74.100
dscontrol set loglevel 1

# add cluster address
dscontrol cluster add 192.168.74.150
dscontrol cluster set 192.168.74.150 proportions 49 50 1 0
dscontrol executor configure 192.168.74.150
dscontrol executor configure 192.168.74.151

# setup port number information on cluster
dscontrol port add 192.168.74.150:80 method nat reset no
dscontrol port add 192.168.74.150:443 method nat reset no

# add WebSEAL servers to the cluster
dscontrol server add 192.168.74.150:80:192.168.74.112 router 192.168.74.112 returnaddress 192.168.74.151
dscontrol server add 192.168.74.150:443:192.168.74.112 router 192.168.74.112 returnaddress 192.168.74.151
dscontrol server add 192.168.74.150:80:192.168.74.113 router 192.168.74.113 returnaddress 192.168.74.151
dscontrol server add 192.168.74.150:443:192.168.74.113 router 192.168.74.113 returnaddress 192.168.74.151

# start the manager and advisor
dscontrol manager start manager.log 10004
dscontrol advisor start Http 80 Http_80.log
dscontrol advisor start ssl 443 ssl_443.log
```

```
dscontrol advisor connecttimeout ssl 443 9
dscontrol advisor receivetimeout ssl 443 9

# setup heartbeat for HA
# first IP is litslb01, second IP is litslb02
dscontrol highavailability heartbeat add 192.168.74.135 192.168.74.99

# number of seconds that the executor uses to timeout HA heartbeats
dscontrol executor set hatimeout 3

# see if the dispatcher has network availability
dscontrol highavailability reach add 192.168.74.251

# set this dispatcher as the primary dispatcher
dscontrol highavailability backup add backup auto 9123

# print the status to screen
dscontrol highavailability status
```

Setting up scripts on both servers: The last change involved modifying the three scripts used for LB high availability – goActive, goInOp and goStandby. We defined the new cluster address to 192.168.74.150 in the following files:

- /opt/ibm/edge/lb/servers/bin/goActive
- /opt/ibm/edge/lb/servers/bin/goInOp
- /opt/ibm/edge/lb/servers/bin/goStandby

Verifying HA status: Once the scripts were run, we executed a dscontrol highavailability status command on both servers to verify their HA status.

On the primary Load Balancer:

```
litslb02:~ # dscontrol highavailability status

High Availability Status:
-------------------------
Role ................. Primary
Recovery strategy .... Auto
State ................ Active
Sub-state ............ Synchronized
Primary host ......... 192.168.74.135
Port ................. 9123
Preferred target ..... 192.168.74.99

Heartbeat Status:
-----------------
Count ................ 1
Source/destination ... 192.168.74.135/192.168.74.99

Reachability Status:
--------------------
Count ................ 1
Address .............. 192.168.74.251 reachable
```

On the backup Load Balancer:

```
litslb01:~ # dscontrol highavailability status

High Availability Status:
-------------------------
Role ................. Backup
Recovery strategy .... Auto
State ................ Standby
Sub-state ............ Synchronized
Primary host ......... 192.168.74.135
Port ................. 9123
Preferred target ..... 192.168.74.135

Heartbeat Status:
-----------------
```

```
Count ................ 1
Source/destination ... 192.168.74.99/192.168.74.135

Reachability Status:
--------------------
Count ................ 1
Address .............. 192.168.74.251 reachable
```

Stopping the primary Load Balancer

We started a workload. For a description of the workload flow, please refer to the list following Figure 32 on page 164. The steps involved in this failure are 2 and 3.

To simulate a Load Balancer failure, we halted the primary Load Balancer.

Indicators and Failover: The backup Load Balancer became "Active" and started to picked up the work shortly after the primary failed:

```
litslb01:~ # dscontrol highavailability status

High Availability Status:
-------------------------
Role ................. Backup
Recovery strategy .... Auto
State ................ Active
Sub-state ............ Not Synchronized
Primary host ......... 192.168.74.135
Port ................. 9123
Preferred target ..... n/a

Heartbeat Status:
-----------------
Count ................ 1
Source/destination ... 192.168.74.99/192.168.74.135

Reachability Status:
--------------------
Count ................ 1
Address .............. 192.168.74.251 reachable
```

It has picked up the cluster IP address, 192.168.74.150:

```
litslb01:/opt/ibm/edge/lb/servers/bin # ifconfig
eth0      Link encap:Ethernet  HWaddr 02:00:00:00:00:0A
          inet addr:192.168.74.99  Bcast:192.168.74.255  Mask:255.255.255.0
          inet6 addr: fe80::200:0:400:a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1492  Metric:1
          RX packets:45166401 errors:0 dropped:0 overruns:0 frame:0
          TX packets:45691777 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3695806907 (3524.5 Mb)  TX bytes:431813846 (411.8 Mb)

eth0:1    Link encap:Ethernet  HWaddr 02:00:00:00:00:0A
          inet addr:192.168.74.150  Bcast:192.168.74.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1492  Metric:1

eth0:2    Link encap:Ethernet  HWaddr 02:00:00:00:00:0A
          inet addr:192.168.74.151  Bcast:192.168.74.255  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:686175 errors:0 dropped:0 overruns:0 frame:0
          TX packets:686175 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:48467975 (46.2 Mb)  TX bytes:48467975 (46.2 Mb)
```

This message appeared on the primary Load Balancer's log, */opt/ibm/edge/lb/servers/logs/lb.log*:

```
Sun Apr 15 16:37:13 EDT 2007 LB just ran goInOp.
```

This message appeared on the backup's log, */opt/ibm/edge/lb/servers/logs/lb.log*:

```
Sun Apr 15 17:41:09 EDT 2007 LB just ran goActive.
```

The workload received a few "HTTP Timeout" errors, and picked up in a few seconds and started working the way it was before.

Failback Actions and Messages: We brought back the primary Load Balancer by starting the Linux system. The script we setup for Load Balancer ran on boot and as soon as the dispatcher was setup, the primary Load Balancer took over as "Active":

```
litslb02:~ # dscontrol highavailability status
High Availability Status:
-------------------------
Role ................. Primary
Recovery strategy .... Auto
State ................ Active
Sub-state ............ Synchronized
Primary host ......... 192.168.74.135
Port ................. 9123
Preferred target ..... 192.168.74.99

Heartbeat Status:
-----------------
Count ................ 1
Source/destination ... 192.168.74.135/192.168.74.99

Reachability Status:
--------------------
Count ................ 1
Address .............. 192.168.74.251 reachable
```

The primary Load Balancer picked up the cluster address:

```
litslb02:~ # ifconfig
eth0      Link encap:Ethernet  HWaddr 02:00:00:00:00:0C
          inet addr:192.168.74.135  Bcast:192.168.74.255  Mask:255.255.255.0
          inet6 addr: fe80::200:0:100:c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1492  Metric:1
          RX packets:2482252 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2633483 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:199375351 (190.1 Mb)  TX bytes:276805701 (263.9 Mb)

eth0:1    Link encap:Ethernet  HWaddr 02:00:00:00:00:0C
          inet addr:192.168.74.150  Bcast:192.168.74.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1492  Metric:1

eth0:2    Link encap:Ethernet  HWaddr 02:00:00:00:00:0C
          inet addr:192.168.74.152  Bcast:192.168.74.255  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:419369 errors:0 dropped:0 overruns:0 frame:0
          TX packets:419369 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:27037937 (25.7 Mb)  TX bytes:27037937 (25.7 Mb)
```

This message appeared on *litslb02:/opt/ibm/edge/lb/servers/logs/lb.log*:

```
Sun Apr 15 19:13:31 EDT 2007 LB just ran goActive.
```

The backup Load Balancer indicated that it was on Standby now:

```
litslb01:/opt/ibm/edge/lb/servers/bin # dscontrol highavailability status

High Availability Status:
-------------------------
Role ................. Backup
Recovery strategy .... Auto
State ................ Standby
Sub-state ............ Synchronized
Primary host ......... 192.168.74.135
Port ................. 9123
Preferred target ..... 192.168.74.135

Heartbeat Status:
-----------------
Count ................ 1
Source/destination ... 192.168.74.99/192.168.74.135

Reachability Status:
--------------------
Count ................ 1
Address .............. 192.168.74.251 reachable
```

This message appeared on *litslb01:/opt/ibm/edge/lb/servers/logs/lb.log*:

```
Sun Apr 15 20:17:04 EDT 2007 LB just ran goStandby.
```

The workload ran continuously without any interruptions.

Load Balancer Gotcha: If you have a non-working “Reach address” defined on the highavailability configuration of your Load Balancers, failback will not work. We found this out because originally we had a reach address of 192.168.74.100, the IP address of our firewall. However, the Load Balancer’s couldn’t “reach” that IP address because ping was disabled on the firewall. We saw “*192.168.74.100 nonreachable*“ messages when we executed *dscontrol highavailability* status. When we brought back the primary Load Balancer after the failover, the primary wouldn’t automatically become “Active”; the backup was still “Active”. This wouldn’t cause a problem in the workflow, but it wasn’t the way failback was supposed to work. Once we changed the “Reach address” to something the Load Balancers could ping, failback worked flawlessly, making the recovery process automatic and painless.

Recovering from a WebSEAL failure

Description

Now that Load Balancer was set up with the new NAT forwarding method to pass incoming requests to our WebSEAL servers, we needed to complete the configuration for our two WebSEAL servers – LITSTAM2 and LITSTAM3. Please refer to Figure 36 on page 199 for configuration of WebSEAL.

Preparation

If you remember our last test report, we only had one WebSEAL server. This time we cloned the existing system to make the second WebSEAL server. We created the following WebSEAL instance for the new server:

```
litstam3-WebSeal-webseald-litstam3.ltic.pok.ibm.com
```

The instance name for the existing WebSEAL was:

```
litstam2-WebSeal-webseald-litstam2.ltic.pok.ibm.com
```

Under both instances, we had the following junctions:

- /tradessl – defined to the Apache service IP at port 443
- /bookstoressl – defined to the Apache service IP at port 443
- /tradetcp – defined to the Apache service IP at port 80
- /bookstoretcp – defined to the Apache service IP at port 80

LITSLB01 and LITSTAM2 resided under the same z/VM LPAR on our z990 CEC. LITSLB02 and LITSTAM3 resided under the same z/VM LPAR on our z900 CEC.

We first had to create a new object to be shared by both WebSEALs through *pdadmin*:

```
pdadmin sec_master> object create \ /WebSEAL/websealcluster.ltic.pok.ibm.com none 1 ispolicyattachable yes
```

To check to see that it was created, we ran:

```
pdadmin sec_master> object list /WebSEAL  /WebSEAL/litstam2.ltic.pok.ibm.com-litstam2-WebSeal
/WebSEAL/litstam3.ltic.pok.ibm.com-litstam3-WebSeal
/WebSEAL/littam20-WebSeal1
/WebSEAL/sharedRoot
/WebSEAL/websealcluster.ltic.pok.ibm.com
```

Then, we made the following changes in each of the WebSEAL configuration files on -

- LITSTAM2 - /opt/pdweb/etc/webseald-litstam2-WebSeal.conf:

  ```
  server-name = websealcluster.ltic.pok.ibm.com
  web-host-name = litstam2.ltic.pok.ibm.com
  ```

- LITSTAM3 - /opt/pdweb/etc/webseald-litstam3-WebSeal.conf:

  ```
  server-name = websealcluster.ltic.pok.ibm.com
  web-host-name = litstam3.ltic.pok.ibm.com
  ```

Because we were using the NAT forwarding method to load balance between the two WebSEALs, on the WebSEAL systems LITSTAM2 and LITSTAM3, we had to make their default routes point to the cluster IP 192.168.74.150, which corresponded to one of the Load Balancers. We also had to remove the route to access systems on the same LAN.

We ran the following three commands to add the new default route, remove the old one, and remove the route to access systems on the same LAN:

```
# /sbin/route add default gw 192.168.74.150
# /sbin/route del default gw 192.168.74.100
# /sbin/route del -net 192.168.74.0/24 eth0
```

The WebSEALs routing table looked like this:

```
litstam2:~ # route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.71.0    litrout74_vip.l 255.255.255.0   UG    0      0        0 eth0
link-local      *               255.255.0.0     U     0      0        0 eth0
loopback        *               255.0.0.0       U     0      0        0 lo
default         192.168.74.150  0.0.0.0         UG    0      0        0 eth0
```

We started the WebSEALs as usual with *pdweb* start.

To make sure WebSEAL and the correct routes were started on boot, we made the following init script, called */etc/init.d/pdwebstart*:

```
/usr/bin/pdweb start
/sbin/route add default gw 192.168.74.150
/sbin/route del default gw 192.168.74.100
/sbin/route del -net 192.168.74.0/24 eth0
```

We made the following script to run when the systems were shutting down, called */etc/init.d/pdwebstop*:

```
/usr/bin/pdweb stop
```

The two scripts were then symlinked to the appropriate runlevels:

```
litstam2:/etc/init.d/rc3.d # ln -s ../pdwebstart S590pd
litstam2:/etc/init.d/rc3.d # ln -s ../pdwebstop K01pd
```

We started a workload, and checked that they were being spread to both WebSEALs with the Load Balancer command *dscontrol* server report.

```
litslb02:/opt/ibm/edge/lb/servers/logs/dispatcher # dscontrol server report ::
Cluster: 192.168.74.150 Port: 80
--------------------------------------------------------------------------------
| Server         |  CPS |    KBPS   |    Total   |  Active |  FINed  |  Comp   |
--------------------------------------------------------------------------------
| 192.168.74.113 |    0 |         0 |        202 |       0 |       0 |     202 |
| 192.168.74.112 |    0 |         0 |        190 |       0 |       0 |     190 |
```

From the above output you can see that for LITSTAM3, there were 202 total connections, and for LITSTAM2, there were 190 total connections.

Stopping a WebSEAL server

With a workload running, we shut down LITSTAM2.

Indicators and Failover: Work continued to flow without any interruptions.

On the Load Balancer, we noticed that work was going to LITSTAM3:

```
litslb02:/opt/ibm/edge/lb/servers/logs/dispatcher # dscontrol server report ::
Cluster: 192.168.74.150 Port: 80
--------------------------------------------------------------------------------
| Server         |  CPS |    KBPS   |    Total   |  Active |  FINed  |  Comp   |
--------------------------------------------------------------------------------
| 192.168.74.113 |    0 |         0 |       1346 |       0 |       0 |    1346 |
| 192.168.74.112 |    0 |         0 |        646 |       0 |       0 |     646 |
```

A few minutes later, we did another server report, and noticed that the number for LITSTAM2 did not increase, while the total connections for LITSTAM3 increased:

```
litslb02:/opt/ibm/edge/lb/servers/logs/dispatcher # dscontrol server report ::
Cluster: 192.168.74.150 Port: 80
--------------------------------------------------------------------------------
| Server         |  CPS |    KBPS   |    Total   |  Active |  FINed  |  Comp   |
--------------------------------------------------------------------------------
| 192.168.74.113 |    0 |         0 |       1530 |       0 |     184 |    1346 |
| 192.168.74.112 |    0 |         0 |        646 |       0 |       0 |     646 |
--------------------------------------------------------------------------------
```

Failback Actions and Messages: We restarted LITSTAM2, and work continued to flow without interruptions.

On the Load Balancer we noticed that total connections for LITSTAM2 started to increase again:

```
litslb02:/opt/ibm/edge/lb/servers/logs/dispatcher # dscontrol server report ::
Cluster: 192.168.74.150 Port: 80
--------------------------------------------------------------------------------
| Server         |  CPS |    KBPS   |    Total   |  Active |  FINed  |  Comp   |
```

```
-------------------------------------------------------------------------------
| 192.168.74.113 |   55 |          0 |        1928 |        0 |      398 |     1530 |
| 192.168.74.112 |   41 |          0 |         862 |        1 |      215 |      646 |
-------------------------------------------------------------------------------
```

Recovering from a LDAP failure

Description

IBM Tivoli Directory Server v6.0 maintained the user registry for the web authorization and authentication services that IBM Tivoli Access Manager for e-business (TAMe) provided for our applications. Every time an end user attempted a new connection to one of our web applications, the user had to provide a user ID and password to TAMe WebSEAL. The ID and password were then authenticated, and a TAMe credential built for authorization checking, using the IBM Tivoli Directory Server user registry. We wanted to ensure that the user registry was highly available and recoverable without interruptions from an LDAP or Linux system failure.

For more information on how IBM Tivoli Directory Server is used in a TAMe and WebSEAL environment; please refer to the following technical paper:

`http://www.ibm.com/systems/z/os/linux/utilities/pdf/lu26st00.pdf`

In order to provide user registry high availability, we setup a peer-to-peer replication scheme for our LDAP server instance.

Peer-to-peer LDAP replication means that there can be several servers acting as masters for directory information, with each master responsible for updating other master servers and replica servers.

For this setup, we configured two identical LDAP server instances that were master servers and that replicated with each other. Please refer back to Figure 36 on page 199 for how LDAP was used in our environment. The two LDAP servers resided on two separate Linux systems. For further elimination of single-point-of-failure, we put the two Linux systems under different z/VM LPARs on different CECs.

We then configured TAMe and WebSEALs to use two LDAP replicas for the user registry.

Recovering from an LDAP failure is fairly straightforward – the infrastructure must ensure that authorization and authentication operations can still run continuously during the failover and the failback of the LDAP server.

Preparation

Keep in mind that before this test, we had one LDAP server and it had the following properties:

- IBM Tivoli Directory Server v6
- Server instance tiodb
- Installed on SUSE LINUX Enterprise Server 9, Service Pack 3
- Already configured for WebSEAL and TAMe
- Communication with WebSEAL and TAMe done via the secure port 636
- Already had end user authorization and authentication information for our applications

For how to complete the above steps, please refer to the Information Center at:

`http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?toc=/com.ibm.itame.doc/toc.xml.`

Creating a second LDAP master server: To setup two LDAP servers in peer-to-peer replication mode, we first needed another LDAP server. These are the steps we followed to create another LDAP server. Please note that the way we did it was fast but it may not be the best especially if you have a lot of server specific updates, or something else installed on the server besides LDAP. There are other ways of creating another LDAP server, such as installing one from scratch. Here, we'll talk about how we created litsldp2. For the installation of our first LDAP server, litsldap, please refer to Chapter 24 of *zSeries Platform Test Report*.

1. We chose the cloning method to create the second LDAP server, litsldp2. In z/VM, we used FLASHCOPY to copy the disks of the existing LDAP server, litsldap, to the target disks of litsldp2.
2. Now we had two completely identical LDAP servers. We updated the new LDAP server to reflect the new hostname and IP address.
3. We tried to start LDAP on the new server, and noticed an error:

```
GLPRDB001E Error code -1 from function:" SQLConnect " ldapdb2b .
10/09/2006 16:31:05     0   0   SQL6048N  A communication error occurred during
START or STOP DATABASE MANAGER processing.
SQL1032N  No start database manager command was issued.  SQLSTATE=57019
GLPRDB004E The LDAP directory server is unable to connect to the database.
GLPSRV064E Failed to initialize be_config.
GLPSRV040E Error encountered. Server starting in configuration only mode.
```

4. Stopped the server so we could debug the problem.

```
# idsslapd -I tiodb -k
```

5. To fix the DB2 problem, we changed the hostname in the db2nodes.cfg file for each of our DB2 instances to reflect litsldp2. The two files edited were:
   - */home/tiodb/sqllib/db2nodes.cfg*
   - */home/db2inst1/sqllib/db2nodes.cfg*
6. To complete the process, we ran the update instance command:

```
litsldp2:/opt/IBM/db2/V8.1/instance # ./db2iupdt db2inst1
DBI1070I Program db2iupdt completed successfully.

litsldp2:/opt/IBM/db2/V8.1/instance # ./db2iupdt tiodb
DBI1070I Program db2iupdt completed successfully.
```

7. We used the secure port for communication between LDAP and WebSEAL and TAMe. Hence we needed to generate a new SSL key database for litsldp2. We named it key2.kdb and the certificate name cert_ldap2. This was to distinguish them from the original LDAP server, litsldap.

```
litsldp2:/usr/ldap/etc # gsk7cmd -keydb -create -db key2.kdb -pw XXXXXX -type cms -expire 999 -stash
litsldp2:/usr/ldap/etc # gsk7cmd -cert -create -db key2.kdb -pw XXXXXX -size 1024 -dn
"CN=litsldp2.ltic.pok.ibm.com,O=ibm,C=US" -label cert_ldap2
```

8. Updated the LDAP configuration file, /home/tiodb/idsslapd-

```
tiodb/etc/ibmslapd.conf, to reflect the new SSL key database:

ibm-slapdSslCertificate: cert_ldap2
ibm-slapdSslKeyDatabase: /usr/ldap/etc/key.kdb
```

9. Now the server specific information had been updated, LDAP was started successfully with the following command:

```
# idsslapd -I tiodb
```

Enabling peer-to-peer LDAP replication: There are two ways of enabling peer-to-peer LDAP replication: via the command line or using the Web Administration Tool. We chose to use the Web Administration Tool since it was easier to use. To install the Web Administration Tool, you need a WebSphere Application Server. We installed it into two existing WebSphere Application Server instances so that we could administer one LDAP server per instance. To install and

setup the Web Administration Tool, we followed the Installation and Configuration guide located in the IBM Tivoli Directory Server Info Center:

```
http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?toc=/com.ibm.IBMDS.doc/toc.xml
```

In the Administration Guide, there is also a section on setting up and administering Replication. The section that we referred to is called “Setting up a simple topology with peer replication”. We recommend that you follow the guide to setup your peer-to-peer LDAP replication. Here, we just want to add the few things that we did that weren’t mentioned in the setup guide.

Before you start the replication process, we recommend that you backup all your registry information on your LDAP server. We used the db2ldif command to export all our entries from our registry:

```
litsldap:/home/tiodb/idsslapd-tiodb/etc # idsdb2ldif -o litsldap.save
litsldp2:/home/tiodb/idsslapd-tiodb/etc # idsdb2ldif -o litsldp2.save
```

If anything goes awry, you can always revert back to the original state with the *ldif2db* command.

We started executing the Web Administration tasks on the original LDAP server, litsldap. We expanded the Replication management category in the navigation area and selected Manage topology. To see the topologies listed in the Replication subtrees table, we first had to add the subtrees by clicking on the “Add subtree” button in the menu section of the table. We browsed and selected the following three subtrees:

1. CN=IBMPOLICIES
2. SECAUTHORITY=DEFAULT
3. o=ibm,c=us

SECAUTHORITY=DEFAULT is the subtree that holds the TAMe-specific data, or “meta-data”. *o=IBM,c=us* is the subtree that holds the user and group entries for our installation. TAMe uses both of these subtrees. You might have other subtrees in your environment that you’d like to replicate.

For each of the three subtrees, we performed the tasks documented in the setup guide.

Here are the options we used during the Add Master task:

- We chose to use the port 389 and not use SSL encryption for the replication.
- Did not create the server as a gateway server.
- We added customized credentials for each subtree and used those when creating the master server. For example, to add a customized credential for the SECAUTHORITY=DEFAULT subtree, we navigated to Replication management -> Manage credentials. We then selected the SECAUTHORITY=DEFAULT subtree, and clicked Add. After specifying credential name, bind ID, and bind password, clicked finish to complete the process.
- Back in the Add master -> Select credential panel, we selected the one we created for the subtree
- In the Add master -> Additional tab, we chose to Add credential information on consumer, which is litsldp2, and added the consumer admin Bind ID and password information.

During the Add master process, the Web Admin tool reported some messages during the process of adding a master:

```
Failed to create topology on consumer litsldp2:389.
You need to synchronize the topology on new server.
 and
Server litsldp2:389 has been added to the replication topology.
However, data must be synchronized in order to fully initialize the new server.
```

The messages are saying that you have to manually add the replication information that you just created to the other LDAP server, the one you are replicating with. To do that, we dumped the registry from *litsldap* and then imported it into litsldp2.

1. We dumped the registry from litsldap:

   ```
   litsldap:/home/tiodb/idsslapd-tiodb/etc # idsdb2ldif -o litsldap.ldif
   ```

2. We copied the litsldap.ldif file over to litsldp2. We stopped the tiodb server instance on litsldp2:

   ```
   litsldp2:/home/tiodb/idsslapd-tiodb/etc # ibmslapd –I tiodb –k
   ```

3. We loaded the ldif on litsldp2:

   ```
   litsldp2:/home/tiodb/idsslapd-tiodb/etc # idsldif2db -r no -i ~/litsldap.ldif
   ```

On litsldp2, we restarted the tiodb instance with *ibmslapd –I tiodb*. We also restarted the LDAP instance on litsldap.

You should make sure that the same credentials for the subtrees that you are replicating exist on all your LDAP masters. In order for them to replicate, they use the credentials to authorize each other. If they are missing on a master, you need to add them via the Replication Management -> Manage Credentials panel on the Web Admin Tool. If your credentials are not correct, you will see a message similar to the following message in your LDAP instance log:

```
GLPRPL036E Error Can't contact LDAP server occurred for replica
'CN=LITSLDP2:389,CN=LITSLDAP.LTIC.POK.IBM.COM:389,IBM-REPLICAGROUP=DEFAULT,DC=IBM,DC=COM': bind failed
using masterDn 'cn=ibmcomcredentials'
```

Verifying the replication: To verify that replication was ready, on the Web Admin Tool for litsldp2 (see Figure 37 on page 211), we navigated to Replication Management -> Manage Queues, and saw that the states for all replicated subtrees were “Ready”. Then on the Web Admin Tool for the other LDAP master, litsldap, we navigated to the same area, and saw that the states were also “Ready”. Sometimes you have to “Suspend” the replica and then “Resume” it to see the “Ready” state.

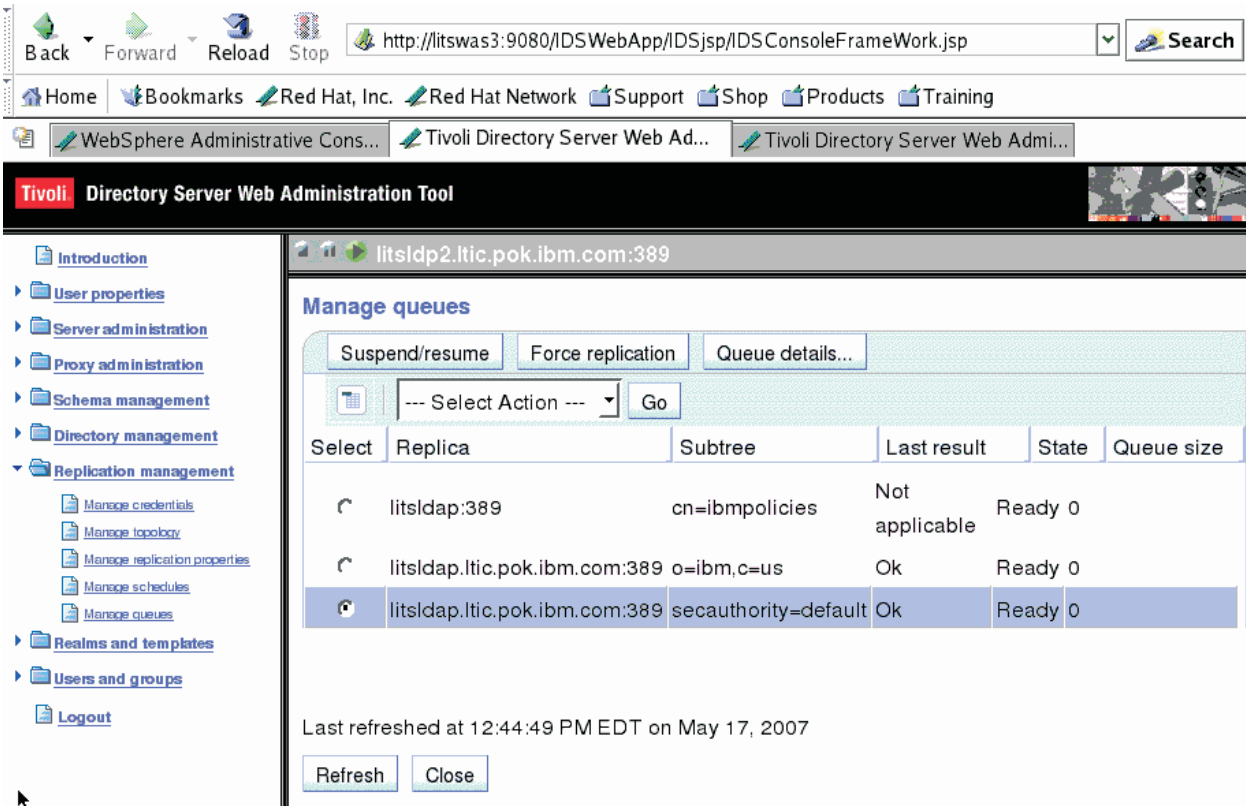


Figure 37. LDAP Web Admin Tool

In the instance log file, located in */home/tiodb/idsslapd-tiodb/logs/ibmslapd.log* for us, indicated that replication was established for the subtree SECAUTHORITY=DEFAULT (similar messages were seen for the other two replicated subtrees), and that a supplier was found and any changes would be replicated immediately:

```
04/13/2007 10:32:38 AM GLPRPL029I Established connection for replica
'CN=LITSLDP2.LTIC.POK.IBM.COM:389,CN=LITSLDAP.LTIC.POK.IBM.COM:389,IBM-REPLICAGROUP=DEFAULT,SECAUTHORITY=DEFAULT'
on host 'litsldp2.ltic.pok.ibm.com' port 389.

04/13/2007 10:32:38 AM GLPRPL022W Replica 'CN=LITSLDP2.LTIC.POK.IBM.COM:389,CN=LITSLDAP.LTIC.POK.IBM.COM:389,IBM-
REPLICAGROUP=DEFAULT,SECAUTHORITY=DEFAULT' missing schedule DN, all changes will be replicated immediately.

04/13/2007 11:23:27 AM GLPRPL073I Client on connection 81 on 192.168.71.187 has bound as a supplier.
```

Configuring TAMe and WebSEAL to use two LDAP replicas: Now that we had two LDAP masters, we must let the TAMe policy server and WebSEAL know about them. In the previous section we created two WebSEAL servers so we had to make the same changes on both WebSEALs.

We added the following lines in the LDAP configuration file for TAMe and the two WebSEAL servers. On TAMe and WebSEAL, the LDAP configuration file is located in */opt/PolicyDirector/etc/ldap.conf*:

```
replica = litsldp2,636,readwrite,9
replica = litsldap,636,readwrite,5
```

These two lines indicate that both replicas; litsldp2 and litsldap, are readable and writeable. In order to avoid write conflicts between the two masters, we configured litsldp2 with a weight of 9, and litsldap with a weight of 5. The one with the higher

weight will always be used first, essentially serving as the primary LDAP server. The one with the smaller weight will serve as the backup in case communication with the primary server is broken.

There was one more task we had to complete before they could communicate with the new LDAP master, litsldp2. We had to import the new LDAP master's SSL key into the policy server and WebSEAL's key databases. To do this, we copied the key database of the new LDAP master, key2.kdb, over to TAMe and WebSEAL. Then we used gsk7ikm to import the litsldp2 key, with the label cert_ldap2, from key2.kdb into its own key database.

To verify that they can communicate with both LDAP servers, we ran some simple *pdadmin* commands from TAMe and WebSEAL, then we stopped LDAP on the primary master, litsldp2, and we continued to be able to execute pdadmin commands. With a *netstat –at*, we noticed communication to litsldap through the ldaps port:

```
tcp        0      0 littam02.ltic.pok:37757 litsldap.ltic.pok:ldaps ESTABLISHED
```

This indicated that the SSL key import worked as well.

We started a BookStore workload from end to end. Please refer to the About Our Environment section for the transaction details of the workload. The areas we were monitoring in the flow were steps 4 – 6 in the transaction flow.

Stopping the primary LDAP server

The failure event was to stop the primary LDAP server, litsldp2, by halting the Linux system.

Indicators and Failover: After we stopped the primary LDAP server, litsldp2, in the WebSEAL log, */var/pdweb/log/msg__webseald-litstam2-WebSeal.log*, we noticed that it recognized the litsldp2 failure and recovered to the backup server litsldap:

```
2007-04-13-16:21:21.078-04:00I----- 0x38AD50C0 webseald WARNING wiv general LDAPClient.cpp 61 0x488a6bc0
DPWIV0192W   LDAP server litsldp2 has failed
2007-04-13-16:21:23.126-04:00I----- 0x38AD50C1 webseald WARNING wiv general LDAPClient.cpp 64 0x4b4ffbc0
DPWIV0193W   LDAP server litsldap has recovered
```

On WebSEAL, a netstat –at also indicated that it was communicating with the backup server now:

```
litstam2:/var/pdweb/log # netstat -at | grep litsldap
tcp        0      0 litstam2.ltic.pok:37688 litsldap.ltic.pok:ldaps ESTABLISHED
tcp        0      0 litstam2.ltic.pok:54631 litsldap.ltic.pok:ldaps ESTABLISHED
tcp        0      0 litstam2.ltic.pok:54606 litsldap.ltic.pok:ldaps ESTABLISHED
tcp        0      0 litstam2.ltic.pok:54585 litsldap.ltic.pok:ldaps ESTABLISHED
tcp        0      0 litstam2.ltic.pok:54555 litsldap.ltic.pok:ldaps ESTABLISHED
tcp        0      0 litstam2.ltic.pok:54559 litsldap.ltic.pok:ldaps ESTABLISHED
tcp        0      0 litstam2.ltic.pok:54667 litsldap.ltic.pok:ldaps ESTABLISHED
tcp        0      0 litstam2.ltic.pok:54431 litsldap.ltic.pok:ldaps ESTABLISHED
tcp        0      0 litstam2.ltic.pok:60901 litsldap.ltic.pok:ldaps ESTABLISHED
```

There were no interruptions in the BookStore workload – everything continued to run as they were.

Failback Actions and Messages: We rebooted the primary LDAP server, litsldp2, and the LDAP server was started on boot. Immediately WebSEAL recognized that it had recovered. Message from */var/pdweb/log/msg__webseald-litstam2-WebSeal.log*:

```
2007-04-13-16:27:52.625-04:00I----- 0x38AD50C1 webseald WARNING wiv general LDAPClient.cpp 64 0x488a6bc0
DPWIV0193W   LDAP server litsldp2 has recovered
```

A *netstat –at* on WebSEAL now indicated connections to both litsldp2 and litsldap:

```
litstam2:/var/pdweb/log # netstat -at | grep litsld
tcp        0      0 litstam2.ltic.pok:39802 litsldp2.ltic.pok:ldaps ESTABLISHED
tcp        0      0 litstam2.ltic.pok:39500 litsldp2.ltic.pok:ldaps ESTABLISHED
tcp        0      0 litstam2.ltic.pok:39167 litsldp2.ltic.pok:ldaps ESTABLISHED
tcp        0      0 litstam2.ltic.pok:37688 litsldap.ltic.pok:ldaps ESTABLISHED
tcp        1      0 litstam2.ltic.pok:44916 litsldp2.ltic.pok:ldaps CLOSE_WAIT
tcp        1      0 litstam2.ltic.pok:44921 litsldp2.ltic.pok:ldaps CLOSE_WAIT
tcp        1      0 litstam2.ltic.pok:44974 litsldp2.ltic.pok:ldaps CLOSE_WAIT
tcp        0      0 litstam2.ltic.pok:54631 litsldap.ltic.pok:ldaps ESTABLISHED
tcp        0      0 litstam2.ltic.pok:54606 litsldap.ltic.pok:ldaps ESTABLISHED
tcp        0      0 litstam2.ltic.pok:54585 litsldap.ltic.pok:ldaps ESTABLISHED
tcp        0      0 litstam2.ltic.pok:60901 litsldap.ltic.pok:ldaps ESTABLISHED
```

On litsldap, the *tiodb* instance log, */home/tiodb/idsslapd-tiodb/logs/ibmslapd.log*, had messages indicating that it had re-established connections with the supplier, litsldp2:

```
04/13/2007 04:27:51 PM GLPRPL073I Client on connection 176 on 192.168.71.187 has bound as a supplier.
04/13/2007 04:27:51 PM GLPRPL073I Client on connection 177 on 192.168.71.187 has bound as a supplier.
04/13/2007 04:27:51 PM GLPRPL073I Client on connection 176 on 192.168.71.187 has bound as a supplier.
04/13/2007 04:27:51 PM GLPRPL073I Client on connection 177 on 192.168.71.187 has bound as a supplier.
```

Recovering from an Apache failure

Description

With the goal of modifying our HA Apache web server cluster to span z/VM instances, native Linux LPARs, and multiple CECs, we set out to reconfigure our existing HA (within a single z/VM instance) cluster from the 2006 test report. In the time since the construction of our last reference architecture, version 2 of the Linux Virtual Server Heartbeat technology had been released. Interestingly it would prove necessary to update to this new release as it resolves a number of issues that would have otherwise prevented us from spanning CECs and VM instances. If you recall our last deployment, we used only 2 Linux virtual server instances and 2 Apache servers. Those 4 systems should be considered the absolute minimum number of machines necessary for an HA architecture.

Yet, the whole architecture still relied on a single z/VM instance. Although the architecture could be adapted to an LPAR only deployment using a rather straightforward conversion process, the test team felt it was necessary to ensure our new reference architecture was capable of high availability under a more heterogeneous environment. Our new design must consider the use of more systems, LPAR only deployments (as requested by some of our readers) and of course, spanning multiple physical machines. Unfortunately the older versions of HA shipped with SLES9 used in our previous testing, did not support more than 2 nodes.

Our new design for a more robust open source Apache web server stack uses 6 apache servers (though 3 would be sufficient) as well as 3 virtual server directors. Some of the inquisitive readers may wonder why we doubled the apache servers if it is not necessary. Quite simply the major reason is that it allowed us to drive higher workload throughputs during our testing in order to simulate larger customer deployments. In addition, at one point early in the testing and design phase, we had constructed the system with additional LVS directors on each of the z/VM instances such that our full HA reference architecture from the last report was maintained even within a single z/VM instance! You may of course chose to do that in your own deployment, but we eventually limited the number of directors simply to save time and administrative work. The architecture presented should scale to many more

directors and backend apache servers as resources permit, but configurations larger than our depiction above have not been deployed in our labs at the time of this writing.

Figure 38 depicts our implementation using the Linux Virtual Server components in addition to the linux-ha.org components. All servers used in the testing of this Scenario were SUSE Linux Enterprise Server 10 and the appropriate generally available software packages shipped with the release.

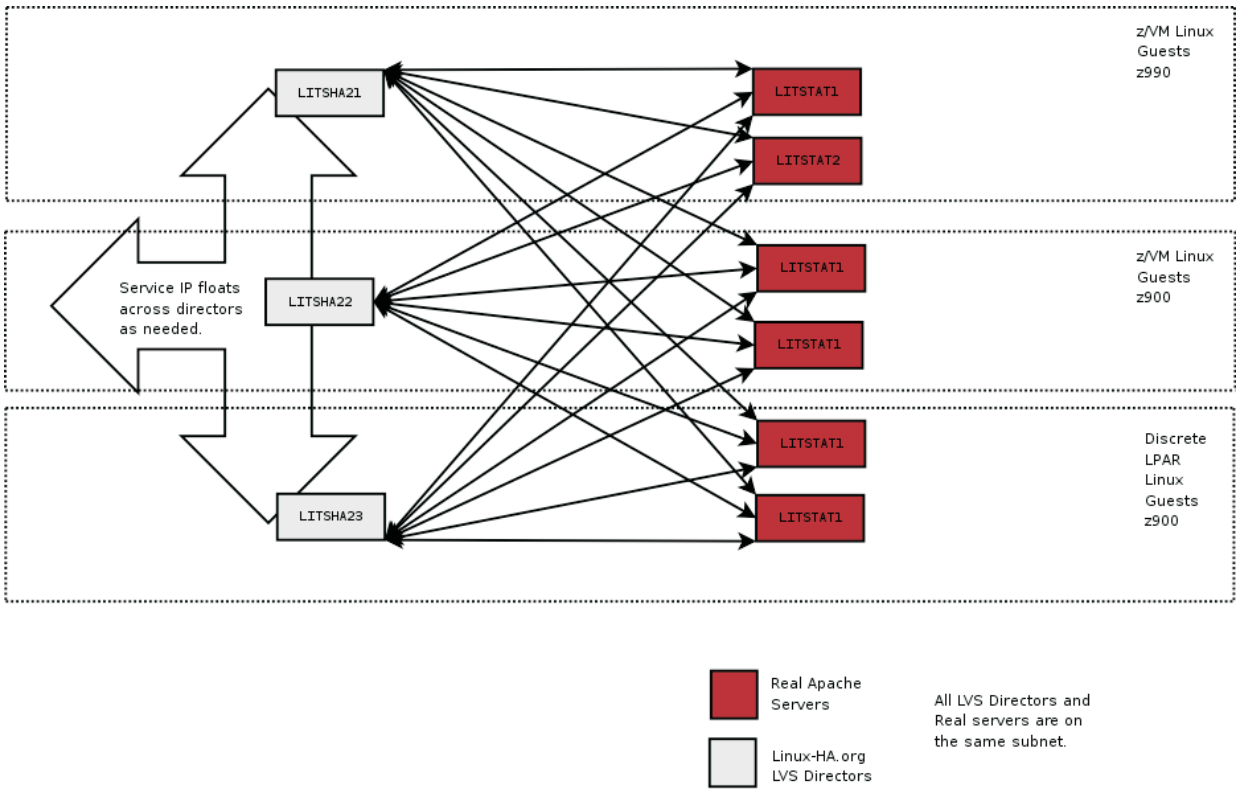


Figure 38. Linux Virtual Servers and Apache

Note that the workload progresses in from the left side of the image working towards the right. The floating resource address for this cluster will reside on one of the LVS director instances at any given time. The service address may be moved manually through a new graphical configuration utility, or is more commonly self managing depending on the state of the LVS directors. Should any director become ineligible (loss of connectivity, software failure etc) the service address will be relocated automatically to an eligible director.

This floating service address spans 2 discrete z/VM instances on 2 separate CECs, as well as a set of native LPAR instances. With our configuration each LVS director is able to forward packets to any real apache web server regardless of physical location or proximity to the active director providing the floating service address. As with our last reference architecture, we will demonstrate how each of the LVS directors can actively monitor the apache servers in order to ensure requests are only sent to operational backend servers.

With our new more robust implementation, we have successfully failed entire zVMs and LPARs with no interruption of service to the consumers of the services enabled on the floating service address (typically http and https web requests).

Terminology:

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 multiple 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 using the Heartbeat software from the Linux-HA project. 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, we used the Mon package. 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 providing the HA service. It is vital to have more than one RealServer providing the service you wish to make HA. In our environment we implemented six RealServers, but adding more is 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 it were 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 back end infrastructure. Each specific method has advantages and disadvantages. LVS-NAT operates on a director server by grabbing incoming packets, that are for configuration specified ports, and reconstructing them dynamically. The director does not handle the packets itself, but rather relays them on to the realservers. The packets are reconstructed with the destination address set to the address of some realserver from the cluster. The packet is then placed back on the network for delivery to the realserver. The realserver is unaware that anything has gone on. As far as the realserver is concerned, it simply received a request directly from the outside world. The replies from the realserver are then sent back to the director where they are rewritten have the source address of the floating IP address that clients are pointed at.

Using the LVS-NAT approach means the realservers require simple TCPIP functionality. The other modes of LVS operation, namely LVS-DR and LVS-Tun

require more complex networking concepts. The major benefit behind our choice for LVS-NAT is that very little alteration is required to the configuration of the realservers. In fact, the hardest part is remembering to set the routing statements properly.

Preparation

Step 1: Building RealServer images: We first started by making our 6 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 listening to port 80 and 443 (for http and https packets respectively) 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 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 command will return 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
```

You can make this permanent by creating a resource script and including it in your configuration. Or, you can edit */etc/sysctl.conf* and put it there.

From an outside system you can *nmap* to make sure the virtual IP has http open. The real address should only show the ports that were open before:

```
[root@host ~]$ 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
```

Be aware that some organizations frown on the use of port scanning tools as *nmap*. Make sure your organization approves before using it.

Next we pointed web browsers 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.

Step 2: Installing and configuring Heartbeat on the directors: Next we moved on to construct the 3 LVS director instances needed for our architecture. We chose to do a fresh installation of SUSE Linux Enterprise Server 10 on each of the LVS Directors. During the initial installation we were sure to select the high availability packages relating to heartbeat, ipvsadm, and mon. If you have an existing installation, you can always use the YAST package manager to add these packages

after your base installation. It is strongly recommended that you add each of the RealServers to the /etc/hosts file. This will ensure there is no DNS related delay 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 $REAL_SERVER_IP_1
$> ping -c 1 $REAL_SERVER_IP_2
$> ping -c 1 $REAL_SERVER_IP_3
$> ping -c 1 $REAL_SERVER_IP_4
$> ping -c 1 $REAL_SERVER_IP_5
$> ping -c 1 $REAL_SERVER_IP_6
```

Configuring Heartbeat Version 2 on SLES10 is quite a bit different than it was for Heartbeat Version 1 on SLES9. Linux-ha.org/v2 provides the documentation that explains how to implement the new Cluster Information Base (CIB), which is XML format. With Heartbeat Version 1, files *haresources*, *ha.cf* and *authkeys* in the */etc/ha.d/* directory were used. The haresources file can be used to generate the new cib.xml file for Version 2 so that's what we did.

We took the ha.cf file from our SLES9 system and added the bottom 3 lines (respawn, pingd and crm) for Version 2. The respawn directive is used to specify a program to run and monitor while it runs. If this program exits with anything other than exit code 100, it will be automatically restarted. The first parameter is the user id to run the program under, and the second parameter is the program to run. The –m parameter sets the attribute pingd to 100 times the number of ping nodes reachable from the current machine. The –d parameter says to delay 5 seconds before modifying the pingd attribute in the CIB. The ping directive is given to declare the PingNode to Heartbeat. The crm directive specifies whether Heartbeat should run the 1.x-style cluster manager or 2.x-style cluster manager that supports more than 2 nodes.

This file should be identical on all the directors. It is absolutely vital that you set the permissions appropriately such that the hacluster daemon can read the file. Failure to do so will cause a slew of warnings in your log files that may be difficult to debug.

```
logfacility daemon
node litsha22
node litsha23
node litsha21
keepalive 3
warntime 5
deadtime 10
ucast eth1 192.168.68.201
ucast eth1 192.168.68.202
ucast eth1 192.168.68.203
respawn hacluster /usr/lib/heartbeat/pingd -m 100 -d 5s
ping litrout71_vip
crm yes
```

For a release 1-style Heartbeat cluster, the haresources file specifies the node name and networking information (floating IP, associated interface and broadcast). This file remained unchanged:

```
litsha21 192.168.71.205/24/eth0/192.168.71.255
```

We only used this file to generate the cib.xml file.

The authkeys file specifies a shared secret allowing directors to communicate with one another. This file also remained unchanged:

```
auth 1
1 sha1 ca0e08148801f55794b23461eb4106db
```

The next few steps will show you the process to convert the version 1 haresources file to the new version 2 XML based configuration format (cib.xml). Though it should be possible to simply copy and use the configuration file we present as a starting point, we strongly suggest following along to tailor the configuration for your deployment.

The actual conversion of file formats is done by issuing the following command:

```
python /usr/lib64/heartbeat/haresources2cib.py /etc/ha.d/haresources > /var/lib/heartbeat/crm/test.xml
```

The following was generated and placed in to file */var/lib/heartbeat/crm/test.xml*.

```
<cib admin_epoch="0" have_quorum="true" num_peers="3" cib_feature_revision="1.3" generated="true"
 ccm_transition="7" dc_uuid="114f3ad1-f18a-4bec-9f01-7ecc4d820f6c" epoch="280" num_updates="5205" cib-last-written=
"Tue Apr  3 16:03:33 2007">
   <configuration>
     <crm_config>
       <cluster_property_set id="cib-bootstrap-options">
         <attributes>
           <nvpair id="cib-bootstrap-options-symmetric_cluster" name="symmetric_cluster" value="true"/>
           <nvpair id="cib-bootstrap-options-no_quorum_policy" name="no_quorum_policy" value="stop"/>
           <nvpair id="cib-bootstrap-options-default_resource_stickiness" name="default_resource_stickiness"
 value="0"/>
           <nvpair id="cib-bootstrap-options-stonith_enabled" name="stonith_enabled" value="false"/>
           <nvpair id="cib-bootstrap-options-stop_orphan_resources" name="stop_orphan_resources" value="true"/>
           <nvpair id="cib-bootstrap-options-stop_orphan_actions" name="stop_orphan_actions" value="true"/>
           <nvpair id="cib-bootstrap-options-remove_after_stop" name="remove_after_stop" value="false"/>
           <nvpair id="cib-bootstrap-options-transition_idle_timeout" name="transition_idle_timeout" value="5min"/>
           <nvpair id="cib-bootstrap-options-is_managed_default" name="is_managed_default" value="true"/>
         </attributes>
       </cluster_property_set>
     </crm_config>
     <nodes>
       <node uname="litsha21" type="normal" id="01ca9c3e-8876-4db5-ba33-a25cd46b72b3">
         <instance_attributes id="standby-01ca9c3e-8876-4db5-ba33-a25cd46b72b3">
           <attributes>
             <nvpair name="standby" id="standby-01ca9c3e-8876-4db5-ba33-a25cd46b72b3" value="off"/>
           </attributes>
         </instance_attributes>
       </node>
       <node uname="litsha23" type="normal" id="dc9a784f-3325-4268-93af-96d2ab651eac">
         <instance_attributes id="standby-dc9a784f-3325-4268-93af-96d2ab651eac">
           <attributes>
             <nvpair name="standby" id="standby-dc9a784f-3325-4268-93af-96d2ab651eac" value="off"/>
           </attributes>
         </instance_attributes>
       </node>
       <node uname="litsha22" type="normal" id="114f3ad1-f18a-4bec-9f01-7ecc4d820f6c">
         <instance_attributes id="standby-114f3ad1-f18a-4bec-9f01-7ecc4d820f6c">
           <attributes>
             <nvpair name="standby" id="standby-114f3ad1-f18a-4bec-9f01-7ecc4d820f6c" value="off"/>
           </attributes>
         </instance_attributes>
       </node>
     </nodes>
     <resources>
       <primitive class="ocf" provider="heartbeat" type="IPaddr" id="IPaddr_1">
         <operations>
           <op id="IPaddr_1_mon" interval="5s" name="monitor" timeout="5s"/>
         </operations>
         <instance_attributes id="IPaddr_1_inst_attr">
           <attributes>
             <nvpair id="IPaddr_1_attr_0" name="ip" value="192.168.71.205"/>
             <nvpair id="IPaddr_1_attr_1" name="netmask" value="24"/>
             <nvpair id="IPaddr_1_attr_2" name="nic" value="eth0"/>
             <nvpair id="IPaddr_1_attr_3" name="broadcast" value="192.168.71.255"/>
           </attributes>
         </instance_attributes>
       </primitive>
     </resources>
     <constraints>
       <rsc_location id="rsc_location_IPaddr_1" rsc="IPaddr_1">
         <rule id="prefered_location_IPaddr_1" score="200">
           <expression attribute="#uname" id="prefered_location_IPaddr_1_expr" operation="eq" value="litsha21"/>
         </rule>
       </rsc_location>
       <rsc_location id="my_resource:connected" rsc="IPaddr_1">
         <rule id="my_resource:connected:rule" score_attribute="pingd">
           <expression id="my_resource:connected:expr:defined" attribute="pingd" operation="defined"/>
         </rule>
```

```
      </rsc_location>
    </constraints>
  </configuration>
</cib>
```

Once this was done and we were happy with the contents, we moved test.xml to cib.xml, changed the owner to hacluster, the group to haclient and restart heartbeat.

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 other LVS Director images established quorum, and instantiated the service address on the newly elected primary node within a matter of seconds. Upon bringing the halted director image back on line, the machines reestablish quorum across all nodes, at which time the floating resource IP was transferred back. The entire process takes only seconds.

Additionally we employed the graphical configuration utility for the heartbeat process, *hb_gui* as shown in Figure 39 on page 220, to manually move the IP address around in our cluster by setting various nodes to the standby or active state. We then retried these steps numerous times, disabling various machines that were active or inactive. With our configuration policy, as long as quorum could be established (for example there are at least 2 directors online) and at least one node was eligible, the floating resource IP address remained operational. During the testing we used simple pings to ensure that no packet loss occurred and we were now confident our floating resource IP was HA.

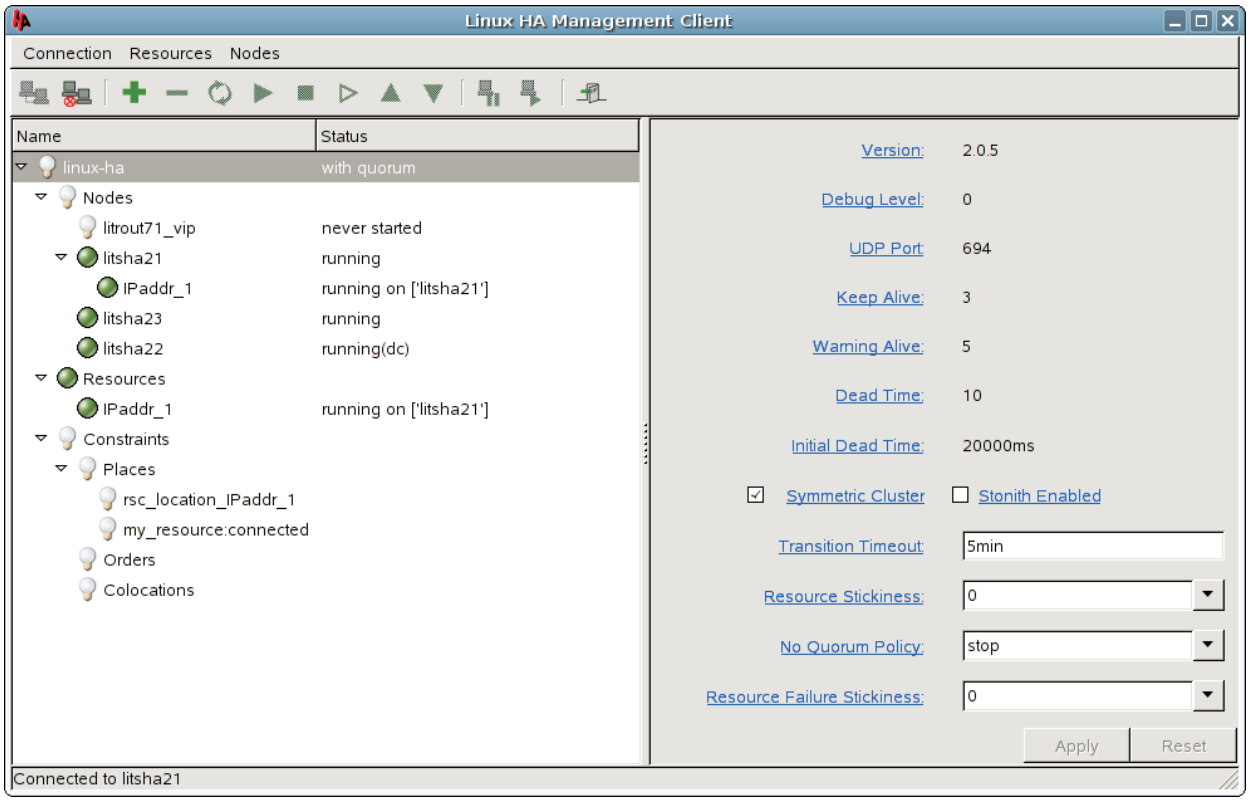


Figure 39. Graphical configuration utility for the heartbeat process, hb_gui

Note that you must log into the *hb_gui* console when you first launch the application. The credentials used are dependent on your deployment. Note how the nodes in our cluster, the litsha2* systems, are each in the running state. The system labeled litsha21 is the current active node as indicated by the addition of a resource displayed immediately below and indented (IPaddr_1).

Of other note is our choice to select ″No Quorum Policy″ to the value ″stop″. This means that any isolated node releases resources it would otherwise own. The implication of that decision is that in our deployment, at any given time 2 heartbeat nodes must be active to establish quorum (in other words a voting majority). Even if a single active, 100% operational, node loses connection to its peer systems due to network failure or simply halting both the inactive peers simultaneously, the resources will be voluntarily released.

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 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 which 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 is 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 second command would return a “1” as output to your terminal. To add this permanently, we added the following to */etc/sysconfig/sysctl*:

```
IP_FORWARD="yes"
```

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 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.

For the majority of our testing we used RR scheduling as it was easy to confirm. We also tried WRR and LC to confirm they work as expected. All reported testing was done with the RR scheduling and its variants.

To get started, we created a script to enable the ipvsadm service forwarding to the real servers used in our deployment. The script was then placed on each of our LVS directors. This script will not be necessary when the later configuration of mon is done to automatically monitor for active real servers, but it aids in testing the ipvsadm component until then. Remember to double check for proper network and http/https connectivity to each of your real servers before executing this script.

```
============================ HA_CONFIG.sh ============================
#!/bin/sh

# The virtual address on the Director which acts as a cluster address
VIRTUAL_CLUSTER_ADDRESS=192.168.71.205
REAL_SERVER_IP_1=192.168.71.220
REAL_SERVER_IP_2=192.168.71.150
REAL_SERVER_IP_3=192.168.71.121
REAL_SERVER_IP_4=192.168.71.145
REAL_SERVER_IP_5=192.168.71.185
REAL_SERVER_IP_6=192.168.71.186

#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), if not the real servers might be clever and not use
# the director as the gateway!
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

# Clear ipvsadm tables
```

```
/sbin/ipvsadm -C

# We install LVS services with ipvsadm for HTTP and HTTPS connections with RR scheduling
/sbin/ipvsadm -A -t $VIRTUAL_CLUSTER_ADDRESS:http -s rr
/sbin/ipvsadm -A -t $VIRTUAL_CLUSTER_ADDRESS:https -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
/sbin/ipvsadm -a -t $VIRTUAL_CLUSTER_ADDRESS:https -r $REAL_SERVER_IP_1:https -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
/sbin/ipvsadm -a -t $VIRTUAL_CLUSTER_ADDRESS:https -r $REAL_SERVER_IP_2:https -m -w 1

# Third Real Server
# Forward HTTP to REAL_SERVER_IP_3 using LVS-NAT (-m), with weight=1
/sbin/ipvsadm -a -t $VIRTUAL_CLUSTER_ADDRESS:http -r $REAL_SERVER_IP_3:http -m -w 1
/sbin/ipvsadm -a -t $VIRTUAL_CLUSTER_ADDRESS:https -r $REAL_SERVER_IP_3:https -m -w 1

# Fourth Real Server
# Forward HTTP to REAL_SERVER_IP_4 using LVS-NAT (-m), with weight=1
/sbin/ipvsadm -a -t $VIRTUAL_CLUSTER_ADDRESS:http -r $REAL_SERVER_IP_4:http -m -w 1
/sbin/ipvsadm -a -t $VIRTUAL_CLUSTER_ADDRESS:https -r $REAL_SERVER_IP_4:https -m -w 1

# Fifth Real Server
# Forward HTTP to REAL_SERVER_IP_5 using LVS-NAT (-m), with weight=1
/sbin/ipvsadm -a -t $VIRTUAL_CLUSTER_ADDRESS:http -r $REAL_SERVER_IP_5:http -m -w 1
/sbin/ipvsadm -a -t $VIRTUAL_CLUSTER_ADDRESS:https -r $REAL_SERVER_IP_5:https -m -w 1

# Sixth Real Server
# Forward HTTP to REAL_SERVER_IP_6 using LVS-NAT (-m), with weight=1
/sbin/ipvsadm -a -t $VIRTUAL_CLUSTER_ADDRESS:http -r $REAL_SERVER_IP_6:http -m -w 1
/sbin/ipvsadm -a -t $VIRTUAL_CLUSTER_ADDRESS:https -r $REAL_SERVER_IP_6:https -m -w 1

# We print the new ipvsadm table for inspection
echo "NEW IPVSADM TABLE:"
/sbin/ipvsadm
============================== HA_CONFIG.sh ============================
```

As you can see the script simply enables the ipvsdam services, then has virtually identical stanzas to forward web and ssl requests to each of the individual real servers. We use the ″-m″ option to specify nat, and we weight each real server equally with a weight of 1 (″-w 1″).

We have now configured each director to handle incoming web and ssl requests to the floating service IP by rewriting them and passing the work on to our RealServers in succession. But in order to get traffic back from the real servers, and do the reverse process before handing the requests back to the client who made the request, we need to alter a few of their networking settings on the directors. We need to do this because we chose to implement our LVS Directors and RealServers in a flat network topology (ie all on the same subnet). We need to perform the following steps to force the Apache response traffic back through the Directors rather than answering directly themselves:

```
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 default route of each of the RealServers to point at 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.

Normally at this point we would try to ensure that our LVS configuration would start automatically at boot, but we do not want to do that just yet! As we mentioned previously we will go one step further to do active monitoring of our real servers to keep a dynamic list of who we can send work to.

Installing and configuring Mon on the LVS Directors: We have thus far established a highly available service IP address and bound that to our pool of realserver instances. But we can not trust each of our individual apache servers to be operational at any given point in time. Since we chose RR scheduling, if any given real server becomes disabled, or ceases to respond to network traffic in a timely fashion (for any reason), 1/6th of our HTTP requests would be failures!

Thus we implement monitoring of the RealServers on each of the the LVS directors in order to dynamically add or remove them from the service pool. To do this, we used another well-known open source package called *mon*.

The *mon* solution is commonly used for monitoring LVS realnodes. *Mon* is relatively easy to configure, and is very extensible for people familiar with shell scripting. We discovered that there was essentially three main steps to get everything working: installation, service monitoring configuration, and alert creation. Yast handled the installation of mon for us automatically, so we only needed to perform the monitoring configuration, and create some alert scripts. The alert scripts are triggered when the monitors determine a realserver has gone off-line, or come back on-line.

Note: Wth heartbeat v2 installations, monitoring of realservers can be accomplished by making all the realserver services resources. Or, the Heartbeat ldirectord package can be used. We did not attempt this configuration in our lab, but may do so in future testing.

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. SLES 10 is a 64 bit Linux image, but the sample configuration as shipped 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 6 directives:

```
hostgroup litstat1 192.168.71.220 #Real Server 1
hostgroup litstat2 192.168.71.150
hostgroup litstat3 192.168.71.121
hostgroup litstat4 192.168.71.145
hostgroup litstat5 192.168.71.185
hostgroup litstat6 192.168.71.186 #Real Server 6
```

If we wanted to add additional real servers, we would simply add additional entries here.

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 real server).

```
================ /etc/mon/mon.cf  ============
#
# global options
#
cfbasedir    = /etc/mon
alertdir     = /usr/lib64/mon/alert.d
mondir       = /usr/lib64/mon/mon.d
statedir     = /var/lib/mon
logdir       = /var/log
maxprocs     = 20
histlength   = 100
historicfile = mon_history.log
randstart    = 60s

#
# authentication types:
#   getpwnam      standard Unix passwd, NOT for shadow passwords
#   shadow        Unix shadow passwords (not implemented)
#   userfile      "mon" user file
#
authtype = getpwnam

#
# downtime logging, uncomment to enable
# if the server is running, don't forget to send a reset command
# when you change this
#
#dtlogfile = downtime.log
dtlogging = yes

#
# NB:  hostgroup and watch entries are terminated with a blank line (or
# end of file).  Don't forget the blank lines between them or you lose.
#

#
# group definitions (hostnames or IP addresses)
# example:
#
# hostgroup servers www mail pop server4 server5
#
# For simplicity we monitor each each individual server as if it were a "group"
# so we add only the hostname and the ip address of an individual node for each.
```

```
hostgroup litstat1 192.168.71.220
hostgroup litstat2 192.168.71.150
hostgroup litstat3 192.168.71.121
hostgroup litstat4 192.168.71.145
hostgroup litstat5 192.168.71.185
hostgroup litstat6 192.168.71.186


#
# Now we set identical watch definitions on each of our groups. They could be customized to treat
# individual servers differently, but we have made the configurations homogeneous here-
 to match our homogeneous LVS configuration.
#

 watch litstat1
     service http
        description http check servers
        interval 6s
        monitor http.monitor -p 80 -u /index.html
        allow_empty_group
        period wd {Mon-Sun}
            alert dowem.down.alert -h
            upalert dowem.up.alert -h
            alertevery 600s
                alertafter 1

 watch litstat2
     service http
        description http check servers
        interval 6s
        monitor http.monitor -p 80 -u /index.html
        allow_empty_group
        period wd {Mon-Sun}
            alert dowem.down.alert -h
            upalert dowem.up.alert -h
            alertevery 600s
                alertafter 1

 watch litstat3
     service http
        description http check servers
        interval 6s
        monitor http.monitor -p 80 -u /index.html
        allow_empty_group
        period wd {Mon-Sun}
            alert dowem.down.alert -h
            upalert dowem.up.alert -h
            alertevery 600s
                alertafter 1

 watch litstat4
     service http
        description http check servers
        interval 6s
        monitor http.monitor -p 80 -u /index.html
        allow_empty_group
        period wd {Mon-Sun}
            alert dowem.down.alert -h
            upalert dowem.up.alert -h
            alertevery 600s
                alertafter 1

 watch litstat5
     service http
        description http check servers
        interval 6s
        monitor http.monitor -p 80 -u /index.html
        allow_empty_group
        period wd {Mon-Sun}
            alert dowem.down.alert -h
            upalert dowem.up.alert -h
            alertevery 600s
                alertafter 1


 watch litstat6
     service http
        description http check servers
```

```
        interval 6s
        monitor http.monitor -p 80 -u /index.html
        allow_empty_group
        period wd {Mon-Sun}
            alert dowem.down.alert -h
            upalert dowem.up.alert -h
            alertevery 600s
                alertafter 1

================ /etc/mon/mon.cf  ============
```

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. In our specific deployment we used the distribution default ″/usr/lib64/mon/alert.d/″ directory. The alerts are responsible for telling LVS to add or remove Apache servers from the eligibility list (by invoking the ipvsadm command as we shall see in a moment). When one of the realservers fails the http test, the dowem.down.alert will be executed by mon with several arguments automatically. Likewise when the monitors determine that a real server has come back online, the mon process executes the dowem.up.alert with the numerous arguments automatically. The names were chosen by the implementor, and you are free to alter them to suit your own custom deployments.

We saved this file, and created the alerts, using simple bash scripting, in the alertdir as follows:

```
================= dowem.up.alert ============
#! /bin/bash

#   The h arg is followed by the hostname we are interested in acting on
#   So we skip ahead to get the -h option since we don't care about the others

REALSERVER=192.168.71.205

while [ $1 != "-h" ] ;
do
        shift
done
ADDHOST=$2

# For the HTTP service
/sbin/ipvsadm -a -t $REALSERVER:http -r $ADDHOST:http -m -w 1

# For the HTTPS service
/sbin/ipvsadm -a -t $REALSERVER:https -r $ADDHOST:https -m -w 1


================= dowem.up.alert ============



================= dowem.down.alert ============
#! /bin/bash

#   The h arg is followed by the hostname we are interested in acting on
#   So we skip ahead to get the -h option since we dont care about the others

REALSERVER=192.168.71.205

while [ $1 != "-h" ] ;
do
        shift
done
```

```
BADHOST=$2


# For the HTTP service
/sbin/ipvsadm -d -t $REALSERVER:http -r $BADHOST

# For the HTTPS service
/sbin/ipvsadm -d -t $REALSERVER:https -r $BADHOST
================= dowem.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. Note that these scripts are far from perfect. Our mon setup only monitored the http port for simple web requests. As implemented in our environment, our HA is vulnerable to situations where a given real server might be operating correctly for http requests and not ssl requests. Under those circumstances we would fail to remove the offending real server from the list of https candidates. This is of course easily remedied by making more advanced alerts specifically for each type of web request in addition to enabling a second https monitor for each real server in the mon configuration file. We leave these tasks as an exercise for the reader. Note that you are not forced to use bourne shell scripting to implement your own alerts or monitors.

To ensure our monitoring was activated, we enabled and disabled the Apache process on each of our real servers in sequence observing each directors proper reaction to the events. Only when we had configured each director was properly monitoring each real server, did we use the *chkconfig* command to make sure that the mon process starts automatically at boot. The specific command used was *"chkconfig mon on"*.

With this last piece in place, we have finished our construction of a cross system highly available Web server infrastructure. In later sections we will discuss the additional configuration of a WebSphere Application Server plugin to enable these Apache servers to do more interesting SOA work.

Note: Some astute readers may wonder why we used *mon* rather than a Heartbeat service. In fact our current system implementation relies on the mon daemon, which is not monitored itself. The heartbeat project has support for such configurations, but we were unable to complete testing of those functions during this test cycle.

Failing the active LVS node

There are numerous reasons why an active node could stop functioning properly in a HA cluster, either voluntarily or involuntarily. The node could loose network connectivity to the other nodes, the heartbeat process could be stopped, environmental occurrences, just to name a few. On z/VM, we failed the active node by either issuing halt on the guest or by setting it to standby mode using the hb_gui (clean take down). Then we got a bit more aggressive (yank the plug!) by logging the guest off while active or by crashing the entire z/VM system. On LPAR, we used the HMC to reset the active node.

Indicators and Failover: There are 2 types of log file indicators available to the system administrator responsible for configuring a Linux HA heartbeat system. The log files vary depending on whether or not a system is the recipient of the floating resource IP address.

On Members of the Cluster which did not receive the Floating Resource IP Address:

```
litsha21:~ # cat  /var/log/messages
Jan 16 12:00:20 litsha21 heartbeat: [3057]: WARN: node litsha23: is dead
Jan 16 12:00:21 litsha21 cib: [3065]: info: mem_handle_event: Got an event OC_EV_MS_NOT_PRIMARY from ccm
Jan 16 12:00:21 litsha21 cib: [3065]: info: mem_handle_event: instance=13, nodes=3, new=1, lost=0, n_idx=0, new_idx=3,
 old_idx=6
Jan 16 12:00:21 litsha21 crmd: [3069]: info: mem_handle_event: Got an event OC_EV_MS_NOT_PRIMARY from ccm
Jan 16 12:00:21 litsha21 crmd: [3069]: info: mem_handle_event: instance=13, nodes=3, new=1, lost=0, n_idx=0,
new_idx=3, old_idx=6
Jan 16 12:00:21 litsha21 crmd: [3069]: info: crmd_ccm_msg_callback:callbacks.c Quorum lost after event=NOT PRIMARY (id=13)
Jan 16 12:00:21 litsha21 heartbeat: [3057]: info: Link litsha23:eth1 dead.
Jan 16 12:00:38 litsha21 ccm: [3064]: debug: quorum plugin: majority
Jan 16 12:00:38 litsha21 ccm: [3064]: debug: cluster:linux-ha, member_count=2, member_quorum_votes=200
Jan 16 12:00:38 litsha21 ccm: [3064]: debug: total_node_count=3, total_quorum_votes=300
                    ................. Truncated For Brevity .................
Jan 16 12:00:40 litsha21 crmd: [3069]: info: update_dc:utils.c Set DC to litsha21 (1.0.6)
Jan 16 12:00:41 litsha21 crmd: [3069]: info: do_state_transition:fsa.c litsha21: State transition S_INTEGRATION ->
S_FINALIZE_JOIN [ input=I_INTEGRATED cause=C_FSA_INTERNAL origin=check_join_state ]
Jan 16 12:00:41 litsha21 crmd: [3069]: info: do_state_transition:fsa.c All 2 cluster nodes responded to the join offer.
Jan 16 12:00:41 litsha21 crmd: [3069]: info: update_attrd:join_dc.c Connecting to attrd...
Jan 16 12:00:41 litsha21 cib: [3065]: info: sync_our_cib:messages.c Syncing CIB to all peers
Jan 16 12:00:41 litsha21 attrd: [3068]: info: attrd_local_callback:attrd.c Sending full refresh
                    ................. Truncated For Brevity .................
Jan 16 12:00:43 litsha21 pengine: [3112]: info: unpack_nodes:unpack.c Node litsha21 is in standby-mode
Jan 16 12:00:43 litsha21 pengine: [3112]: info: determine_online_status:unpack.c Node litsha21 is online
Jan 16 12:00:43 litsha21 pengine: [3112]: info: determine_online_status:unpack.c Node litsha22 is online
Jan 16 12:00:43 litsha21 pengine: [3112]: info: IPaddr_1        (heartbeat::ocf:IPaddr):      Stopped
Jan 16 12:00:43 litsha21 pengine: [3112]: notice: StartRsc:native.c  litsha22   Start IPaddr_1
Jan 16 12:00:43 litsha21 pengine: [3112]: notice: Recurring:native.c litsha22      IPaddr_1_monitor_5000
Jan 16 12:00:43 litsha21 pengine: [3112]: notice: stage8:stages.c Created transition graph 0.
                    ................. Truncated For Brevity .................
Jan 16 12:00:46 litsha21 mgmtd: [3070]: debug: update cib finished
Jan 16 12:00:46 litsha21 crmd: [3069]: info: do_state_transition:fsa.c litsha21: State transition S_TRANSITION_ENGINE ->
 S_IDLE [ input=I_TE_SUCCESS cause=C_IPC_MESSAGE origin=do_msg_route ]
Jan 16 12:00:46 litsha21 cib: [3118]: info: write_cib_contents:io.c Wrote version 0.53.593 of the CIB to disk (digest:
 83b00c386e8b67c42d033a4141aaef90)
```

As we see from the above listing, a roll is taken, and sufficient members for quorum are available for the vote. A vote is taken and normal operation is resumed with no further action needed.

On the member of the cluster who received the Floating Resource IP Address:

```
litsha22:~ # cat  /var/log/messages
Jan 16 12:00:06 litsha22 syslog-ng[1276]: STATS: dropped 0
Jan 16 12:01:51 litsha22 heartbeat: [3892]: WARN: node litsha23: is dead
Jan 16 12:01:51 litsha22 heartbeat: [3892]: info: Link litsha23:eth1 dead.
Jan 16 12:01:51 litsha22 cib: [3900]: info: mem_handle_event: Got an event OC_EV_MS_NOT_PRIMARY from ccm
Jan 16 12:01:51 litsha22 cib: [3900]: info: mem_handle_event: instance=13, nodes=3, new=3, lost=0, n_idx=0,
new_idx=0, old_idx=6
Jan 16 12:01:51 litsha22 crmd: [3904]: info: mem_handle_event: Got an event OC_EV_MS_NOT_PRIMARY from ccm
Jan 16 12:01:51 litsha22 crmd: [3904]: info: mem_handle_event: instance=13, nodes=3, new=3, lost=0, n_idx=0,
new_idx=0, old_idx=6
Jan 16 12:01:51 litsha22 crmd: [3904]: info: crmd_ccm_msg_callback:callbacks.c Quorum lost after event=NOT PRIMARY (id=13)
Jan 16 12:02:09 litsha22 ccm: [3899]: debug: quorum plugin: majority
Jan 16 12:02:09 litsha22 crmd: [3904]: info: do_election_count_vote:election.c Election check: vote from litsha21
Jan 16 12:02:09 litsha22 ccm: [3899]: debug: cluster:linux-ha, member_count=2, member_quorum_votes=200
Jan 16 12:02:09 litsha22 ccm: [3899]: debug: total_node_count=3, total_quorum_votes=300
Jan 16 12:02:09 litsha22 cib: [3900]: info: mem_handle_event: Got an event OC_EV_MS_INVALID from ccm
Jan 16 12:02:09 litsha22 cib: [3900]: info: mem_handle_event: no mbr_track info
Jan 16 12:02:09 litsha22 cib: [3900]: info: mem_handle_event: Got an event OC_EV_MS_NEW_MEMBERSHIP from ccm
Jan 16 12:02:09 litsha22 cib: [3900]: info: mem_handle_event: instance=14, nodes=2, new=0, lost=1, n_idx=0,
new_idx=2, old_idx=5
Jan 16 12:02:09 litsha22 cib: [3900]: info: cib_ccm_msg_callback:callbacks.c LOST: litsha23
Jan 16 12:02:09 litsha22 cib: [3900]: info: cib_ccm_msg_callback:callbacks.c PEER: litsha21
Jan 16 12:02:09 litsha22 cib: [3900]: info: cib_ccm_msg_callback:callbacks.c PEER: litsha22
                    ................. Truncated For Brevity .................
Jan 16 12:02:12 litsha22 crmd: [3904]: info: update_dc:utils.c Set DC to litsha21 (1.0.6)
Jan 16 12:02:12 litsha22 crmd: [3904]: info: do_state_transition:fsa.c litsha22: State transition S_PENDING -> S_NOT_DC
[ input=I_NOT_DC cause=C_HA_MESSAGE origin=do_cl_join_finalize_respond ]
Jan 16 12:02:12 litsha22 cib: [3900]: info: cib_diff_notify:notify.c Update (client: 3069, call:25): 0.52.585 ->
0.52.586 (ok)
                    ................. Truncated For Brevity .................
Jan 16 12:02:14 litsha22 IPaddr[3998]: INFO: /sbin/ifconfig eth0:0 192.168.71.205 netmask 255.255.255.0
broadcast 192.168.71.255
Jan 16 12:02:14 litsha22 IPaddr[3998]: INFO: Sending Gratuitous Arp for 192.168.71.205 on eth0:0 [eth0]
Jan 16 12:02:14 litsha22 IPaddr[3998]: INFO: /usr/lib64/heartbeat/send_arp -i 500 -r 10 -p
/var/run/heartbeat/rsctmp/send_arp/send_arp-192.168.71.205 eth0 192.168.71.205 auto 192.168.71.205 ffffffffffff
Jan 16 12:02:14 litsha22 crmd: [3904]: info: process_lrm_event:lrm.c LRM operation (3) start_0 on IPaddr_1 complete
Jan 16 12:02:14 litsha22 kernel: send_arp uses obsolete (PF_INET,SOCK_PACKET)
Jan 16 12:02:14 litsha22 kernel: klogd 1.4.1, ---------- state change ----------
Jan 16 12:02:14 litsha22 kernel: NET: Registered protocol family 17
Jan 16 12:02:15 litsha22 crmd: [3904]: info: do_lrm_rsc_op:lrm.c Performing op monitor on IPaddr_1 (interval=5000ms,
key=0:f9d962f0-4ed6-462d-a28d-e27b6532884c)
Jan 16 12:02:15 litsha22 cib: [3900]: info: cib_diff_notify:notify.c Update (client: 3904, call:18): 0.53.591 ->
0.53.592
 (ok)
Jan 16 12:02:15 litsha22 mgmtd: [3905]: debug: update cib finished
```

As we see from the above listing, a roll is taken, and sufficient members for quorum are available for the vote. A vote is taken followed by the *ifconfig* commands that are issued to claim the floating resource IP address.

As an additional means of indicating when a failure has occurred, a systems programmer may log into any of the cluster members and execute the *hb_gui* command. Through this method, system programmers can determine by visual inspection which system has the floating resource.

Lastly, we would be remiss if we did not take a moment to illustrate a sample log file from a no quorum situation. In our deployment if any singular node cannot communicate with either of its peers, it has lost quorum (since 2/3 is the majority in a 3 voting party). Under these situations the node considers that is has lost quorum, and invokes the no quorum policy handler. The following is an example of the log file from such an event.

No Quorum Policy Actions:

```
litsha22:~ # cat /var/log/messages
.....................
Jan 16 12:06:12 litsha22 ccm: [3899]: debug: quorum plugin: majority
Jan 16 12:06:12 litsha22 ccm: [3899]: debug: cluster:linux-ha, member_count=1, member_quorum_votes=100
Jan 16 12:06:12 litsha22 ccm: [3899]: debug: total_node_count=3, total_quorum_votes=300
                    ................... Truncated For Brevity ...................
Jan 16 12:06:12 litsha22 crmd: [3904]: info: crmd_ccm_msg_callback:callbacks.c Quorum lost after event=INVALID (id=15)
Jan 16 12:06:12 litsha22 crmd: [3904]: WARN: check_dead_member:ccm.c Our DC node (litsha21) left the cluster
                    ................... Truncated For Brevity ...................
Jan 16 12:06:14 litsha22 IPaddr[5145]: INFO: /sbin/route -n del -host 192.168.71.205
Jan 16 12:06:15 litsha22 lrmd: [1619]: info: RA output: (IPaddr_1:stop:stderr) SIOCDELRT: No such process
Jan 16 12:06:15 litsha22 IPaddr[5145]: INFO: /sbin/ifconfig eth0:0 192.168.71.205 down
Jan 16 12:06:15 litsha22 IPaddr[5145]: INFO: IP Address 192.168.71.205 released
Jan 16 12:06:15 litsha22 crmd: [3904]: info: process_lrm_event:lrm.c LRM operation (6) stop_0 on IPaddr_1 complete
Jan 16 12:06:15 litsha22 cib: [3900]: info: cib_diff_notify:notify.c Update (client: 3904, call:32): 0.54.599 ->
0.54.600 (ok)
Jan 16 12:06:15 litsha22 mgmtd: [3905]: debug: update cib finished
```

As we see from the above listing, when quorum is lost for any given node, it relinquishes any resources as a result of our no quorum policy configuration. The choice of no quorum policy is up to the systems programmer.

Failback Actions and Messages: One of the more interesting implications of a properly configured Linux HA system is that there is no real action to take on the part of a systems programmer to re-instantiate a cluster member. Simply activating the Linux instance is sufficient to let the node rejoin its peers automatically. If you have configured a primary node (ie one which is favored to gain the floating resource above all others) it will regain the floating resources automatically. Non favored systems will simply join the eligibility pool and proceed as normal.

No Quorum Policy Actions:

```
litsha22:~ # tail -f /var/log/messages
Jan 16 12:09:02 litsha22 heartbeat: [3892]: info: Heartbeat restart on node litsha21
Jan 16 12:09:02 litsha22 heartbeat: [3892]: info: Link litsha21:eth1 up.
Jan 16 12:09:02 litsha22 heartbeat: [3892]: info: Status update for node litsha21: status init
Jan 16 12:09:02 litsha22 heartbeat: [3892]: info: Status update for node litsha21: status up
Jan 16 12:09:22 litsha22 heartbeat: [3892]: debug: get_delnodelist: delnodelist=
Jan 16 12:09:22 litsha22 heartbeat: [3892]: info: Status update for node litsha21: status active
Jan 16 12:09:22 litsha22 cib: [3900]: info: cib_client_status_callback:callbacks.c Status update: Client litsha21/cib now
 has status [join]
Jan 16 12:09:23 litsha22 heartbeat: [3892]: WARN: 1 lost packet(s) for [litsha21] [36:38]
Jan 16 12:09:23 litsha22 heartbeat: [3892]: info: No pkts missing from litsha21!
Jan 16 12:09:23 litsha22 crmd: [3904]: notice: crmd_client_status_callback:callbacks.c Status update: Client litsha21/crmd
 now has status [online]
.....................
Jan 16 12:09:31 litsha22 crmd: [3904]: info: crmd_ccm_msg_callback:callbacks.c Quorum (re)attained after event=NEW
MEMBERSHIP (id=16)
Jan 16 12:09:31 litsha22 crmd: [3904]: info: ccm_event_detail:ccm.c NEW MEMBERSHIP: trans=16, nodes=2, new=1, lost=0
 n_idx=0, new_idx=2, old_idx=5
Jan 16 12:09:31 litsha22 crmd: [3904]: info: ccm_event_detail:ccm.c     CURRENT: litsha22 [nodeid=1, born=13]
Jan 16 12:09:31 litsha22 crmd: [3904]: info: ccm_event_detail:ccm.c     CURRENT: litsha21 [nodeid=0, born=16]
Jan 16 12:09:31 litsha22 crmd: [3904]: info: ccm_event_detail:ccm.c     NEW:     litsha21 [nodeid=0, born=16]
Jan 16 12:09:31 litsha22 cib: [3900]: info: cib_diff_notify:notify.c Local-only Change (client:3904, call: 35):
0.54.600 (ok)
Jan 16 12:09:31 litsha22 mgmtd: [3905]: debug: update cib finished
.....................
Jan 16 12:09:34 litsha22 crmd: [3904]: info: update_dc:utils.c Set DC to litsha22 (1.0.6)
Jan 16 12:09:35 litsha22 cib: [3900]: info: sync_our_cib:messages.c Syncing CIB to litsha21
Jan 16 12:09:35 litsha22 crmd: [3904]: info: do_state_transition:fsa.c litsha22: State transition S_INTEGRATION ->
 S_FINALIZE_JOIN [ input=I_INTEGRATED cause=C_FSA_INTERNAL origin=check_join_state ]
Jan 16 12:09:35 litsha22 crmd: [3904]: info: do_state_transition:fsa.c All 2 cluster nodes responded to the join offer.
Jan 16 12:09:35 litsha22 attrd: [3903]: info: attrd_local_callback:attrd.c Sending full refresh
Jan 16 12:09:35 litsha22 cib: [3900]: info: sync_our_cib:messages.c Syncing CIB to all peers
```

```
.....................
Jan 16 12:09:37 litsha22 tengine: [5119]: info: send_rsc_command:actions.c Initiating action 4: IPaddr_1_start_0 on litsha22
Jan 16 12:09:37 litsha22 tengine: [5119]: info: send_rsc_command:actions.c Initiating action 2: probe_complete on litsha21
Jan 16 12:09:37 litsha22 crmd: [3904]: info: do_lrm_rsc_op:lrm.c Performing op start on IPaddr_1 (interval=0ms,
 key=2:c5131d14-a9d9-400c-a4b1-60d8f5fbbcce)
Jan 16 12:09:37 litsha22 pengine: [5120]: info: process_pe_message:pengine.c Transition 2: PEngine Input stored in:
 /var/lib/heartbeat/pengine/pe-input-72.bz2
Jan 16 12:09:37 litsha22 IPaddr[5196]: INFO: /sbin/ifconfig eth0:0 192.168.71.205 netmask 255.255.255.0 broadcast
192.168.71.255
Jan 16 12:09:37 litsha22 IPaddr[5196]: INFO: Sending Gratuitous Arp for 192.168.71.205 on eth0:0 [eth0]
Jan 16 12:09:37 litsha22 IPaddr[5196]: INFO: /usr/lib64/heartbeat/send_arp -i 500 -r 10 -p
 /var/run/heartbeat/rsctmp/send_arp/send_arp-192.168.71.205 eth0 192.168.71.205 auto 192.168.71.205 ffffffffffff
Jan 16 12:09:37 litsha22 crmd: [3904]: info: process_lrm_event:lrm.c LRM operation (7) start_0 on IPaddr_1 complete
Jan 16 12:09:37 litsha22 cib: [3900]: info: cib_diff_notify:notify.c Update (client: 3904, call:46): 0.55.607 -> 0.55.608
 (ok)
Jan 16 12:09:37 litsha22 mgmtd: [3905]: debug: update cib finished
Jan 16 12:09:37 litsha22 tengine: [5119]: info: te_update_diff:callbacks.c Processing diff (cib_update):
0.55.607 -> 0.55.608
Jan 16 12:09:37 litsha22 tengine: [5119]: info: match_graph_event:events.c Action IPaddr_1_start_0 (4) confirmed
Jan 16 12:09:37 litsha22 tengine: [5119]: info: send_rsc_command:actions.c Initiating action 5: IPaddr_1_monitor_5000 on
 litsha22
Jan 16 12:09:37 litsha22 crmd: [3904]: info: do_lrm_rsc_op:lrm.c Performing op monitor on IPaddr_1 (interval=5000ms,
 key=2:c5131d14-a9d9-400c-a4b1-60d8f5fbbcce)
Jan 16 12:09:37 litsha22 cib: [5268]: info: write_cib_contents:io.c Wrote version 0.55.608 of the CIB to disk (digest:
98cb6685c25d14131c49a998dbbd0c35)
Jan 16 12:09:37 litsha22 crmd: [3904]: info: process_lrm_event:lrm.c LRM operation (8) monitor_5000 on IPaddr_1 complete
Jan 16 12:09:38 litsha22 cib: [3900]: info: cib_diff_notify:notify.c Update (client: 3904, call:47): 0.55.608 ->
0.55.609 (ok)
Jan 16 12:09:38 litsha22 mgmtd: [3905]: debug: update cib finished
```

As we see from the above listing, quorum is re-established. When quorum is re-established, a vote is performed and litsha22 becomes the active node with the floating resource.

Recovering from a WebSphere Application Server failure

Description

Recovering from WebSphere Application Server Failure was covered in our last test report, *zSeries Platform Test Report*. Our recovery results were the same as detailed in the last test report, except for one Gotcha that we discovered with the WebSphere Application Server Plug-in.

First, let us give you an overview of our expanded setup. We updated our WebSphere Application Server configuration by consolidating our WebSphere Application Servers into a 6 server cluster, and then spreading the servers across 2 different z/VM LPARs and 2 native Linux LPARs.

Table 13 shows the details of our WebSphere Application Server configuration.

Table 13. Our WebSphere Application Server configuration

Hostname	WebSphere version	Linux distro	LPAR
litrwas1	6.0.2.5	RedHat Enterprise Linux 4 Update 4	z/VM LPAR VMI on z990
litrwas1	6.0.2.1	SUSE LINUX Enterprise Server 9 SP 3	z/VM LPAR VMI on z990
litrwas2	6.0.2.5	RedHat Enterprise Linux 4 Update 4	z/VM LPAR VMK on z900
litrwas2	6.0.2.1	SUSE LINUX Enterprise Server 9 SP 3	z/VM LPAR VMK on z900
litrwas3	6.0.2.5	RedHat Enterprise Linux 4 Update 4	Linux LPAR LIT3 on z900
litrwas3	6.0.2.1	SUSE LINUX Enterprise Server 9 SP 3	Linux LPAR LIT4 on z900

Two J2EE applications, BookStore and Trade6, ran on their own clustered WebSphere Application Server instances. The Tivoli Directory Server Web Admin Tool was installed on two different server instances.

WebSphere Application Server Plug-in Gotcha: During our testing we noticed that whenever we ran a failover test where we halted one or more WebSphere Application Server systems, the workload would only flow intermittently. On every third attempt or so, the http or https request would time out.

After debugging, we found that the WebSphere Application Server Web server Plug-in by default performs a blocking connect, in which the plug-in sits until an operating system times out (which could be as long as 2 minutes depending on the platform) and allows the plug-in to mark the server unavailable. We would notice messages like these in the plug-in log file, */opt/IBM/WebSphere/Plugins/logs/litstat5/http_plugin.log*:

```
[Fri Mar 30 14:27:19 2007] 00002d03 40402aa0 - ERROR: ws_server: serverSetFailoverStatus: Marking
litswas2_BookStore_litswas2 down
[Fri Mar 30 14:27:19 2007] 00002d03 40402aa0 - STATS: ws_server: serverSetFailoverStatus:
Server litswas2_BookStore_litswas2 : pendingConnections 0 failedConnections 9 affinityConnections 0
totalConnections 0.
[Fri Mar 30 14:27:19 2007] 00002d03 40402aa0 - ERROR: ws_common: websphereHandleRequest:
Failed to execute the transaction to 'litswas2_BookStore_litswas2'on host 'litswas2.ltic.pok.ibm.com';
will try another one
```

We could tell that it was detecting the failed WebSphere Application Server instances, but it would wait until the OS timed out before marking the server unavailable and redirecting the request to the next available one. The wait would cause a timeout for the end user who was performing the http or https request, and that was why we were noticing the intermittent workflow.

The parameter that defined the timeout value is “ConnectTimeout” in the plugin-cfg.xml file that is generated by WebSphere Application Server and propagated to the Web server. We changed this value from the default of 0, to 2 seconds for every server instance;

```
<Server CloneID="11mpp68iq" ConnectTimeout="2" ExtendedHandshake="false"
LoadBalanceWeight="2" MaxConnections="-1" Name="litrwas_BookStore_litrwas1"
ServerIOTimeout="0" WaitForContinue="false">
```

Then we restarted Apache to pick up the latest Plug-in configuration, and the intermittent problem was gone. We were getting a page every time we attempt a http or https request. You may want to tweak this and possibly other parameters further to obtain optimal results.

We could tell when the plug-in was trying the failed servers (litrwas2 and litswas2, in our case) the request was responded to at a slower rate because the plug-in waited for the timeout then marked the server unavailable and moved the request to the next available server.

Note: The plug-in by default is set to recheck on the failed servers every 60 seconds to see if they came back, we did not change this value.

Recovering from a DB2 UDB failure

Description

In our previous report, *zSeries Platform Test Report*, we wrote about recovering from DB2 UDB failure in the section titled “Implementing HA Reference Architectures: WebSphere with DB2 on Linux”. This time, our setup of DB2 UDB and TSAM to provide a highly available computing environment using DB2 HADR

involved a small update – making it even more highly available. In our original test, the DB2 primary and standby servers resided on the same z/VM. We have now moved the standby server to another z/VM located on another CEC. Setup remained the same – please reference our December 2006 Test report for details on setup. Our recovery results were the same as detailed in the last test report, except for one Gotcha which we discuss below.

DB2 HADR Gotcha: We tested and verified DB2 HADR failover by shutting down the z/VM guest. In our initial tests, when shutting down the z/VM guest hosting the primary node, the standby node would never take control to become the primary node. Instead, we captured these errors in the db2diag.log:

```
2007-04-03-18.13.46.846915-240 I65939A435         LEVEL: Severe
PID     : 30117                TID  : 2199083423808PROC : db2hadrs (HADRDB) 0
INSTANCE: db2instp             NODE : 000         DB   : HADRDB
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduAcceptEvent, probe:20280
MESSAGE : Failed to connect to primary. rc:
DATA #1 : Hexdump, 4 bytes
0x000003FFFFFFB8D8 : 810F 0019                                    ....

2007-04-03-18.13.46.847267-240 I66375A371         LEVEL: Severe
PID     : 30117                TID  : 2199083423808PROC : db2hadrs (HADRDB) 0
INSTANCE: db2instp             NODE : 000         DB   : HADRDB
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduAcceptEvent, probe:20280
RETCODE : ZRC=0x810F0019=-2129723367=SQLO_CONN_REFUSED "Connection refused"
```

We determined that an odd behavior occurred when shutting down the z/VM hosting the primary node, the OS on the primary node would restart itself. As a result, rlogin was not being restarted upon boot causing the standby node to believe the primary node was still active and available. We made the change so that the rlogin daemon is started upon reboot. When we performed the z/VM shutdown again, failover occurred as expected onto the standby node.

Recovering from a DB2 z/OS failure

Description

Our DB2 z/OS Data sharing group configuration did not change from the previous test report. Our WebSphere Application Server JDBC configuration also remained the same. For Recovering from a DB2 z/OS Failure, please refer to the December 2006 Test Report, *zSeries Platform Test Report*, to the section titled “Implementing HA Reference Architecture: WebSphere with DB2 database on z/OS”.

Recovery from a z/VM, LPAR, Linux failure

Figure 40 on page 233 depicts our environment for this round of recovery testing. As you can see, we’ve expanded our infrastructure availability by splitting our critical software and applications across different z/VM LPARs and native Linux LPARs. In this chapter, we discuss recovering from Linux, LPAR and z/VM failures.

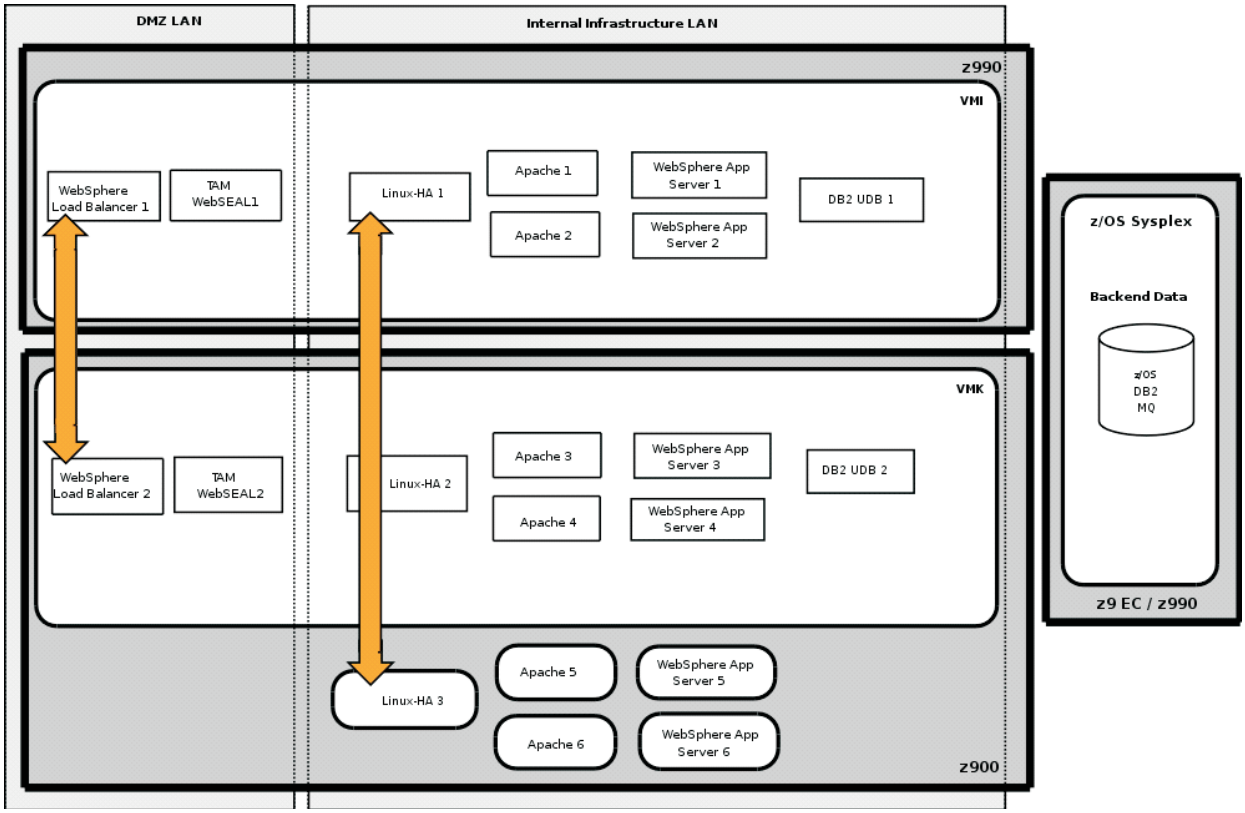


Figure 40. Final high availability networking picture

Recovering from a Linux and LPAR Failure

For all of our Linux software server recoveries, we tested them by halting the Linux systems, simulating a Linux failure. In the cases of LVS Director, Apache, and WebSphere Application Server, we also tested the scenarios by halting the Linux LPARs that they were installed on. The procedure and results are covered in “Recovery from a software server failure” on page 197.

Recovering from a z/VM Failure

Description

To test recovering from z/VM failure, we picked the VMK LPAR to fail. The following Linux systems resided on VMK:

- litslb02 – Load Balancer
- litstam3 – WebSEAL
- litsha21 – Linux Virtual Server Director
- litstat3 – Apache Web server
- litstat4 – Apache Web server
- litrwas2 – WebSphere Application Server
- litswas2 – WebSphere Application Server
- litdat02 – DB2 UDB
- litsldp2 – LDAP server

Preparation

In order to conduct a meaningful z/VM failure, we first had to make sure that the workloads would flow through Linux systems on VMK. This way we would be able to verify that when we conduct the failure event, the workloads were still running.

We ensured that the following servers on VMK were the “active” primary servers:

- litslb02 – Load Balancer
- litsha21 – Linux Virtual Server Director
- litdat02 – DB2 UDB
- litsldp2 – LDAP server

WebSEAL, Apache, and WebSphere Application Server were load balanced so litstam3, litstat3, litstat4, litrwas2 and litswas2 would receive work during the failover.

We tested the following scenarios before “pulling the plug” on VMK:

1. Tested recovery of litrwas2 and litswas2
2. Tested recovery of litstat3 and litstat4
3. Tested recovery of litsha21
4. Tested recovery of litstam3
5. Tested recovery of litsldp2
6. Tested recovery of litslb02

For detailed information on the recovery of any of the above systems, please refer to “Recovery from a software server failure” on page 197.

Next, we made sure the time was in sync on all of our Linux systems. The easiest way to ensure this is with a time server.

We also gathered the locations of all our software logs, found in Table 14, so that we could monitor them during the failover and failback. We cleared the logs so we could start fresh.

Table 14. Software logs and locations

Logs	Location
Load Balancer	*/opt/ibm/edge/lb/servers/logs/dispatcher/*log*
WebSEAL	*/var/pdweb/log/msg*log /var/pdweb/www-litstam*-WebSeal/log/*log**
LDAP	*/home/tiodb/idsslapd-tiodb/logs/*log*
LVS Directors	
Apaches	*/usr/local/apache2.0.49_susessl7c/logs/*log*
WebSphere Application Server Plug-ins	*/opt/IBM/WebSphere/Plugins/logs/litstat*/*log*
WebSphere Application Server	*/opt/IBM/WebSphere/AppServer/profiles/litswas*/logs/BookStore_lit*/*log /opt/IBM/WebSphere/AppServer/profiles/litswas*/logs/TradeLIT*/*log /opt/IBM/WebSphere/AppServer/profiles/RHELDmgr/logs/dmgr/*log*
DB2 z/OS	*/opt/IBM/db2/V8.1/logs/db2zos/pool**
DB2 UDB	*/misc/home*/db2inst*/sqllib/db2dump/*log*

We started the BookStore and Trade6 workloads.

For a typical new transaction, the flow to access both applications was:

1. Client initiated application request from the “outside” world.
2. The request was handled by the firewall and passed to the TAMe WebSEAL cluster address.
3. WebSphere Application Server Network Deployment Edge Component Load Balancer handled spraying the request to a member of the TAMe WebSEAL cluster.
4. WebSEAL then asked the end user for authentication and authorization information.
5. The end user input this information.
6. WebSEAL checked against the LDAP user registry for authentication and authorization. If OK, then WebSEAL passed the request to the Apache cluster address.*
7. The LVS Director handled spraying the request to an available Apache server.
8. The WebSphere Application Server Plug-in that was installed on the Apache server, would transfer the request to an available member in the WebSphere Application Server cluster.
9. WebSphere Application Server fulfilled the request. If the request involved a transaction to DB2, then WebSphere Application Server used the JDBC Type 4 driver to pass the request onto DB2. 1. For Trade, the request went to the DB2 UDB cluster on Linux. 2. For BookStore, the request went to DB2 z/OS data sharing group and shared Message Queue, depending on the type of transaction.
10. Once the request was fulfilled, the response was bubbled back to the client.

Please note that WebSEAL itself has the capability to load balance among Apache Web servers. If you are configuring Apache HA, you can do so without the Linux Virtual Server Director layer. Because we are a test team, we chose to have WebSEAL go to the Apache cluster address so that we could test Linux-ha.org’s Linux Virtual Server and heartbeat components.

Failing z/VM

We used the following z/VM EXEC to simulate a z/VM failure on VMK:

```
KILLVM EXEC:
       /* */
       say;say;say
       say "ABOUT TO KILL VM - ARE YOU SURE YOU WANT TO DO THIS (Y/N)"
       pull yn
       if yn='Y'
       then DO
          say "kill it"
          'CP SET DUMP OFF'
          'STORE HR1B0 0 0'
       END
       else SAY "No action taken"
```

By running KILLVM, we basically killed VMK by storing 0’s in the PSW. This method caused VM to attempt a restart, but since the WARM start data was invalid, we were prompted to enter ″GO″ before the system starts IPLing. We entered ″STOP″ to prevent VM from IPLing.

Indicators and Failover: VMK came down hard, bringing down with it all the Linux guests in 10 seconds. The following guests were dead:

- litslb02 – Load Balancer
- litstam3 – WebSEAL
- litsha21 – Linux Virtual Server Director
- litstat3 – Apache Web server
- litstat4 – Apache Web server
- litrwas2 – WebSphere Application Server
- litswas2 – WebSphere Application Server
- litdat02 – DB2 UDB
- litsldp2 – LDAP server

The workloads died for about 30 seconds with “HTTP Timeout” error messages. Then they both picked up to comparable throughputs as before the failure event.

On the Load Balancer, we noticed that litslb01, the standby server had now become “Active”:

```
litslb01:~ # dscontrol highavailability status

High Availability Status:
-------------------------
Role ................. Backup
Recovery strategy .... Auto
State ................ Active
Sub-state ............ Not Synchronized
Primary host ......... 192.168.74.135
Port ................. 9123
Preferred target ..... n/a

Heartbeat Status:
-----------------
Count ................ 1
Source/destination ... 192.168.74.99/192.168.74.135

Reachability Status:
--------------------
Count ................ 1
Address .............. 192.168.74.251 reachable
```

We could see that work was being spread to litstam3, the surviving WebSEAL node on the other VM LPAR:

```
litslb01:~ # dscontrol server report ::
Cluster: 192.168.74.150 Port: 80
--------------------------------------------------------------------------------
| Server         |  CPS |    KBPS   |    Total   |  Active |  FINed  |  Comp   |
--------------------------------------------------------------------------------
| 192.168.74.113 |    0 |         0 |        656 |       0 |       0 |    646 |
| 192.168.74.112 |    0 |         0 |       2762 |     110 |       0 |     2652 |
--------------------------------------------------------------------------------
```

On litstam3, the WebSEAL instance log indicated that it was recovering to use litsldap now:

```
2007-04-13-16:21:21.078-04:00I----- 0x38AD50C0 webseald WARNING wiv general LDAPClient.cpp 61 0x488a6bc0
DPWIV0192W   LDAP server litsldp2 has failed
2007-04-13-16:21:23.126-04:00I----- 0x38AD50C1 webseald WARNING wiv general LDAPClient.cpp 64 0x4b4ffbc0
DPWIV0193W   LDAP server litsldap has recovered
```

On the hb_gui, Linux Virtual Server graphical configuration utility for the heartbeat process, we noticed that litsah22, the Director on Linux LPAR, held the service IP address. This meant that work was flowing through that director to get to Apache.

In the WebSphere Application Server Plug-in logs, we could see that it was marking litrwas2 and litswas2 as failed and redirecting traffic to the remaining WebSphere Application Servers:

```
[Mon Apr  2 09:51:24 2007] 000035ca 40402aa0 - ERROR: ws_common: websphereGetStream: Connect timeout fired
[Mon Apr  2 09:51:24 2007] 000035ca 40402aa0 - ERROR: ws_common: websphereExecute: Failed to create the stream
[Mon Apr  2 09:51:24 2007] 000035ca 40402aa0 - ERROR: ws_server: serverSetFailoverStatus: Marking litswas2_TradeLITSWAS2 down
[Mon Apr  2 09:51:24 2007] 000035ca 40402aa0 - STATS: ws_server: serverSetFailoverStatus: Server litswas2_TradeLITSWAS2 :
 pendingConnections 0 failedConnections 3 affinityConnections 0 totalConnections 0.
[Mon Apr  2 09:51:24 2007] 000035ca 40402aa0 - ERROR: ws_common: websphereHandleRequest: Failed to execute the transaction
 to 'litswas2_TradeLITSWAS2'on host 'litswas2.ltic.pok.ibm.com'; will try another one
[Mon Apr  2 09:51:24 2007] 000035ca 40402aa0 - STATS: ws_server_group: serverGroupCheckServerStatus: Checking status of
 litswas3_TradeLITSWAS3, ignoreWeights 0, markedDown 0, retryNow 0, wlbAllows 1 reachedMaxConnectionsLimit 0
[Mon Apr  2 09:51:25 2007] 000035ca 40402aa0 - STATS: ws_server: serverSetFailoverStatus: Server litswas3_TradeLITSWAS3 :
pendingConnections 0 failedConnections 1 affinityConnections 0 totalConnections 2.
```

On litdat01, we could see that it had taken over as the primary DB2 UDB:

```
# db2 get snapshot for all on hadrdb
HADR Status
  Role                    = Primary
  State                   = Peer
  Synchronization mode    = Sync
  Connection status       = Connected, 04/12/2007 15:21:21.867232
  Heartbeats missed       = 0
  Local host              = 192.168.71.104
  Local service           = 60014
  Remote host             = 192.168.71.117
  Remote service          = 60014
  Remote instance         = db2insts
  timeout(seconds)        = 120
  Primary log position(file, page, LSN) = S0000019.LOG, 0, 00000000055F0000
  Standby log position(file, page, LSN) = S0000019.LOG, 0, 00000000055F0000
  Log gap running average(bytes) = 0
```

Failback Actions and Messages: After 10 minutes of stable conditions, we IPLed VMK. The systems came up in the order defined in AUTOLOG2:

```
/*********************************************************************/
/* Sample nodeid EXEC for Integration Test.                          */
/*********************************************************************/
/* ensure that the network is up before           */
/* xautologging systems, sleep for 2 minutes      */
  'CP SLEEP 2 MIN'
  'XAUTOLOG LITSDNS2'
  'XAUTOLOG LITSLDP2'
  'XAUTOLOG LITDAT02'
  'XAUTOLOG LITRWAS2'
  'XAUTOLOG LITSWAS2'
  'CP SLEEP 5 MIN'
  'XAUTOLOG LITSTAT3'
  'XAUTOLOG LITSTAT4'
  'CP SLEEP 1 MIN'
  'XAUTOLOG LITSHA21'
  'XAUTOLOG LITSTAM3'
  'CP SLEEP 1 MIN'
  'XAUTOLOG LITSLB02'
exit
```

After the systems came up, no interruptions were noticed from the workloads perspective.

On the Load Balancer, we noticed that litslb02, the primary server had now become “Active”:

```
litslb02:~ # dscontrol highavailability status
High Availability Status:
-------------------------
Role ................. Primary
Recovery strategy .... Auto
State ................ Active
```

```
Sub-state ............ Synchronized
Primary host ......... 192.168.74.135
Port ................. 9123
Preferred target ..... 192.168.74.99

Heartbeat Status:
-----------------
Count ................ 1
Source/destination ... 192.168.74.135/192.168.74.99

Reachability Status:
--------------------
Count ................ 1
Address .............. 192.168.74.251 reachable
```

We noticed that work was being spread to both litstam2 and litstam3.

On the WebSEAL servers, we noticed in their instance logs that litsldp2 has recovered:

```
DPWIV0193W   LDAP server litsldp2 has recovered
```

On the hb_gui, Linux Virtual Server graphical configuration utility for the heartbeat process, we noticed that litsah21, the Director on VMK, held the service IP address. This meant that work was flowing through that director to get to Apache.

In the WebSphere Application Server Plug-in logs, we didn't notice anymore errors reporting that WebSphere Application Server instances were unavailable. On the WebSphere Application Server Network Deployment Manager administration console, we also noticed that all the WebSphere Application Sever instanced were in "Running" state.

On litdat02, we could see that it had taken over as the primary DB2 UDB:

```
# db2 get snapshot for all on hadrdb
HADR Status
  Role                   = Primary
  State                  = Peer
  Synchronization mode   = Sync
  Connection status      = Connected, 04/12/2007 15:20:05.005974
  Heartbeats missed      = 0
  Local host             = 192.168.71.117
  Local service          = 60014
  Remote host            = 192.168.71.104
  Remote service         = 60014
  Remote instance        = db2instp
  timeout(seconds)       = 120
  Primary log position(file, page, LSN) = S0000019.LOG, 0, 00000000055F0000
  Standby log position(file, page, LSN) = S0000019.LOG, 0, 00000000055F0000
  Log gap running average(bytes) = 0
```

Summary of Linux Recovery Test

With this round of recovery testing, we had the opportunity to enhance the availability of our infrastructure and consequently our applications. We added infrastructure redundancy by splitting our clustered servers across two z/VM LPARs. Furthermore, to reflect the environment of some of our Linux on System z customers, we expanded the internal server clusters to native Linux LPARs as well. Additionally, the two z/VM LPARs and native Linux LPARs were split across two physical CECs. We also had the opportunity to enhance the availability of our networking infrastructure, DASD and disk I/O, and software products.

Below is a list of the areas we tested during the expanded recovery test:

- Recovering from Network Failure – VSWITCH configuration
- Recovering from Network Failure – VSWITCH configuration
- Recovering from Network Failure – channel bonding configuration
- Recovering from SCSI path Failure – multipathing configuration
- Recovering from DASD Failure – GDPS/PPRC configuration
- Recovering from Hardware Crypto Failure
- Recovering from VM console failure
- Recovering from RACF Failure
- Recovering from z/VM TCP/IP Failure
- Recovering from Load Balancer Failure
- Recovering from WebSEAL Failure
- Recovering from LDAP Failure – Peer-to-peer replication configuration
- Recovering from Apache Failure – Linux Virtual Server configuration
- Recovering from WebSphere Failure
- Recovering from DB2 UDB Failure
- Recovering from DB2 z/OS Failure
- Recovering from z/VM Failure

While all of the scenarios were successful, we did experience some bumps on the road getting there. We discuss our recommendations in this chapter.

Our recommendations

Redundancy, redundancy, redundancy

We recommend that you duplicate everything to eliminate single points of failure. Here is a list of considerations:

Duplication:

- LPARs
- Operating systems
- Software servers
- Control Units
- I/O paths (FICON, FCP)
- Ports and cards (FICON, FCP, disk host adapters, OSA)
- FICON Directors
- Fiber channel switches
- Network routers

Mirroring:

- Disks (system, data, swap)
- Data

We were able to duplicate all of our software servers, as well as LPARs, operating systems, VSWITCH, various I/O paths, and hardware cryptographic cards. We also tested DASD HA using GDPS/PPRC Multiplatform Resiliency, which utilizes disk mirroring technologies.

Networking recommendations

Pri-router and OSA Layer3: Linux systems (native or z/VM guests) running a router or Network Address Translation (NAT) Server using OSA in Layer3 mode need to have pri-router set up. We learned the hard way that you can only have a single Linux system with pri-router per OSA card. Therefore for each router or NAT Server in your environment you need a separate OSA. The good news is that you can have as many non pri-router Linux systems sharing that same OSA as you want.

OSA Layer2 setup: On System z you can configure an OSA in Layer2 mode. This removes the limitation of only having one router or NAT Server per OSA. It also removes the need to set pri-router in the Ethernet definitions.

```
echo 1 > /sys/bus/ccwgroup/drivers/qeth/x.x.xxxx/layer2
```

when defining the Ethernet interface dynamically. Alternately, you can select Layer2 when defining the network interface during Linux install.

Layer2 mode also makes debugging problems with a network sniffer possible because each Linux guest will have a unique MAC address. This is unlike Linux guests using Layer 3 which all share the identical MAC.

Consistent MTU Size for Linux: The default Maximum Translation Unit (MTU) size for SUSE LINUX Enterprise Server 9, SUSE LINUX Enterprise Server 10, and RedHat Enterprise Linux 4 is 1492. Unless overridden in the network interface configuration file, Linux images for those distributions will have a default MTU size of 1492.

To eliminate networking issues, we recommend using a consistent MTU size in a complex networking environment involving multiple different networking technologies. We ended up using the system default of 1492.

Multipathing

We recommend you use multipathing whenever possible. Multipathing increases system stability and resilience. Multipathing tools can be used when SCSI disks are attached to achieve high availability and load balancing. We used LVM2 for multipathing with our FCP SCSI device - making one path fail and ensuring one of the others can be used to continue accessing the device without knowledge to the application.

DASD HA with GDPS/PPRC

We recommend having DASD HA for data protection and availability. We used GDPS/PPRC Multiplatform Resiliency and found this solution to be flawless in terms of recovery and data integrity. To implement this solution, you need the required hardware as well as purchase the service offering.

Using hardware cryptographic cards

System z Peripheral Component Interconnect (PCI) cryptographic cards can accelerate asymmetric cryptographic operations for Linux on System z. These PCI cryptographic cards (PCI Cryptographic Coprocessor, PCI Cryptographic Accelerator, PCI Extended Cryptographic Coprocessor, Crypto Express2) are commonly used to provide leading-edge performance of the complex Rivest-Shamir-Adelman (RSA) cryptographic operations used in the SSL protocol.

In addition to the performance benefits of using hardware cryptographic cards, z/VM virtualizes their usage for Linux guests so that high availability and automatic

recovery are all done under the covers, provided you have more than one cryptographic card of the same type. We highly recommend this configuration because of its performance, availability, ease of setup, and automatic recovery.

Chapter 14. Future Linux on System z projects

Following are some areas of future testing for the Linux on System z team.

Systems management tools

Testing of a set of systems management technologies and tools across multiple disciplines (availability management, data management, capacity management, change control, log management, user administration, problem determination, provisioning, and so on) in a multi-tier, integrated operating system environment (z/OS, Linux, and virtualization) to validate correctness, interoperability, and their contribution to operational simplification. Also, identify an optimum “quick start” set of systems management tools and techniques that work together and/or complement each other efficiently.

Includes testing the model of an IT shop as an in-house service provider catering to different Lines of Business (LOBs) within the company, with each LOB having its own pool of servers, and the admin responsibilities for those servers split appropriately between the IT shop and the LOBs.

Appendix A. Some of our parmlib members

This section describes how we have set up some of our parmlib members for z/OS. Table 15 summarizes our new and changed parmlib members for z/OS V1R8 and z/OS.e V1R8. Samples of some of our parmlib members are available on the Samples page of our Web site.

Table 15. Summary of our parmlib changes for z/OS V1R8 and z/OS.e V1R8

Member name	z/OS release	Change summary	Related to
IFAPRD*xx*	z/OS.e V1R8	No changes from z/OS.e V1R7. (See our December 2005 edition.)	z/OS.e, dynamic enablement

Appendix B. Some of our RMF reports

In this appendix we include some of our RMF™ reports.

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  V1R8              SYSTEM ID Z1              START 01/26/2007-10.00.00  INTERVAL 00.29.59
                                     RPT VERSION V1R8 RMF      END   01/26/2007-10.30.00  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
001/26 10.00.00 29.59  28.5   2.1 915.4   0.0   130   129   144   143   468   405     1     0   243    92 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  V1R8              SYSTEM ID JE0             START 01/26/2007-10.15.00  INTERVAL 00.14.59
                                     RPT VERSION V1R8 RMF      END   01/26/2007-11.00.00  CYCLE 0.100 SECONDS
0
 NUMBER OF INTERVALS 3
-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
001/26 10.15.00 14.59  27.6   1.6  1649   0.0     3     2    10    10   389   386     2     0    30    19 0.00   0.00
 01/26 10.30.00 14.59  30.8   1.4  1926   0.0     3     1    10    10   388   378     1     0    28    20 0.00   0.00
 01/26 10.45.00 15.00  31.3   1.4  2014   0.0     3     2    10    10   378   375     1     0    25    19 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  V1R8              SYSTEM ID JH0             START 01/26/2007-10.15.00  INTERVAL 00.14.59
                                     RPT VERSION V1R8 RMF      END   01/26/2007-11.00.00  CYCLE 0.100 SECONDS
0
 NUMBER OF INTERVALS 3
-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
001/26 10.15.00 14.59  19.0   3.0 114.6   0.0     3     2     0     0   383   382     1     0    24    16 0.00   0.00
 01/26 10.30.00 14.59  18.0   3.2 109.1   0.0     3     1     1     0   384   380     1     0    23    17 0.00   0.16
 01/26 10.45.00 15.00  16.3   3.1  97.8   0.0     3     1     1     1   381   380     1     0    23    17 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  V1R8              SYSTEM ID J80             START 01/26/2007-10.15.00  INTERVAL 00.14.59
                                     RPT VERSION V1R8 RMF      END   01/26/2007-11.00.00  CYCLE 0.100 SECONDS
0
 NUMBER OF INTERVALS 3
-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
001/26 10.15.00 14.59  74.6   4.1  8706   0.0   192   191    48    47   454   449     1     0    83    66 0.00   0.01
 01/26 10.30.00 14.59  80.0   4.0  8610   0.0   194   191    48    47   462   452     1     0    78    64 0.00   0.01
 01/26 10.45.00 15.00  79.2   3.7  9507   0.0   192   189    46    46   453   444     1     0    75    66 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  V1R8              SYSTEM ID J90             START 01/26/2007-10.15.00  INTERVAL 00.14.59
                                     RPT VERSION V1R8 RMF      END   01/26/2007-11.00.00  CYCLE 0.100 SECONDS
0
 NUMBER OF INTERVALS 3
-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
001/26 10.15.00 14.59  80.2   3.5 921.9   0.0   167   166    15    15   372   367     1     0    26    18 0.00   0.04
 01/26 10.30.00 14.59  84.7   3.4 947.7   0.0   167   165    15    14   370   366     1     0    28    18 0.00   0.01
 01/26 10.45.00 15.00  79.3   0.0   0.0   0.0   166   164    14    14   370   366     1     0    25    17
1                                        R M F   S U M M A R Y   R E P O R T
PAGE  001
             z/OS  V1R8              SYSTEM ID Z0              START 01/26/2007-10.15.00  INTERVAL 00.14.59
                                     RPT VERSION V1R8 RMF      END   01/26/2007-11.00.00  CYCLE 0.100 SECONDS
0
 NUMBER OF INTERVALS 3
-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
| 001/26 10.15.00 15.00  16.3   5.7 610.8   0.0     4     2    14    14   389   386     2     0    38    28 0.00   0.01
|  01/26 10.30.00 14.59  15.6   6.0 565.9   0.0     3     2    14    14   390   387     1     0    35    27 0.00   0.00
|  01/26 10.45.00 15.00  16.3   6.1 554.7   0.0     3     2    14    14   389   384     1     0    36    28 0.00   0.01
```

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 1894 CICS transactions/second.

```
|              HARDCOPY  RMF V1R8   Sysplex Summary - UTCPLXJ8       Line 1 of 80
|  Command ===>                                                  Scroll ===> CSR
| 0WLM Samples: 479     Systems: 10 Date: 01/26/07 Time: 10.16.00 Range: 120   Sec
| 0
|                        >>>>>>>>XXXXXXXXXXXXXXXXXX<<<<<<<<
|
|   Service Definition: WLMDEF02              Installed at: 10/25/06, 13.27.45
|        Active Policy: WLMPOL01              Activated at: 10/25/06, 13.29.47
|                  ------- 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
| 0BATCH     W           72                                0.692 0.719  8.100  8.731
|  BATCHHI   S  2    50  63                           0.80  0.000 0.000  0.000  0.000
|  DISCR     S  D        20                                0.692 0.719  8.100  8.731
|  WLMBTCHH  S  2    50  87                           0.57  0.000
|  CICS      W           78                                 1894 0.000  0.046  0.062
|  CICS      S  2       N/A  0.600 80%        100%  0.50   1532 0.000  0.019  0.024
| CICSCONV S  3       N/A  10.00 50%         69%  0.70  3.592 0.000  9.292  9.292
| CICSDEFA S  3       N/A  1.000 90%         94%  0.50  131.2 0.000  0.471  0.364
| CICSMISC S  3       N/A  1.000 90%        100%  0.50  226.9 0.000  0.001  0.001
| CICSRGN  S  2    60  78                           0.77  0.000
| ICSS     W           68                                28.38 0.000  0.013  0.014
| FAST     S  2    50 100                           0.50  22.70 0.000  0.004  0.004
| SLOW     S  4    50 100                           0.50  5.675 0.000  0.052  0.053
| VEL30    S  2    30  68                           0.44  0.000
| IMS      W          N/A                                90.47 0.000  0.270  0.627
| IMSTMHI  S  2       N/A  0.500 90%         95%  0.50  31.18 0.000  0.109  0.134
| IMSTMLOW S  4       N/A  1.000 90%        100%  0.50  1.442 0.000  0.054  0.059
| IMSTMMED S  3       N/A  0.700 90%         72%  4.00  57.85 0.000  0.360  0.907
| STC      W           42                                63.20 0.001  4.659  4.660
| DB2HIGH  S  2    50  60                           0.84  0.000 0.000  0.000  0.000
| DDF      S  5     5  96                           0.05  7.983 0.001  0.013  0.014
| IMS      S  2    50  79                           0.64  0.000
| IMSHIGH  S  2    60  81                           0.74  0.000
| LDAP     S           91                                0.000 0.000  0.000  0.000
|          3  5    10  91                           0.11  0.000 0.000  0.000  0.000
| MQSERIES S  2    40  75                           0.53  0.017 0.000  2.30H  2.30H
|
```

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 77.6% of our CICS transactions are completing in 0.5 seconds, and our CICS workload is processing 551.25 transactions per second.

```
| 1                                  W O R K L O A D   A C T I V I T Y
| PAGE   5
|          z/OS V1R8             SYSPLEX UTCPLXJ8          START 01/26/2007-10.30.00 INTERVAL 000.14.59   MODE = GOAL
|                              CONVERTED TO z/OS V1R8 RMF     END   01/26/2007-10.44.59
```

```
                                        POLICY ACTIVATION DATE/TIME 10/25/2006 13.29.47

    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                22
    MPL      0.00  EXECUTION             18
    ENDED  346638  QUEUED                 0
    END/S  385.16  R/S AFFIN               0
    #SWAPS      0  INELIGIBLE             0
    EXCTD  473510  CONVERSION             0
    AVG ENC  0.00  STD DEV              112
    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             CONV  I/O LOCK MISC PROD LTCH                                        LOCAL SYSPL REMOT
    CICS  BTE  77.6   3.3   0.0   0.3  0.1  95.8  0.4  0.0  0.0  0.1  0.0                                          88.2   8.0   0.0
    CICS  EXE  28.8  60.6   0.0   6.9  0.0   0.0 29.8  0.0  0.2  2.5  0.0                                           7.2  42.5   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  19.3  86.2   0.0   0.0  0.0   0.0  0.4  0.6 12.6  0.0  0.2                                           0.0   0.0   0.0
    IMS   BTE   0.9  98.9   0.0   0.0  0.0   0.0  0.0  1.1  0.0  0.0  0.0                                           0.0   0.0   0.0
    IMS   EXE   5.4  90.9   0.0   0.0  0.0   0.0  0.0  9.1  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  31.6  12.1   0.0   0.0  0.0   0.0 55.4 32.5  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
    JE0             100      N/A  0.5
    J80             100      N/A  0.5
    J90             100      N/A  0.5
    Z0              100      N/A  0.5
1                                       W O R K L O A D   A C T I V I T Y
                                                                                                          PAGE   6
          z/OS V1R8              SYSPLEX UTCPLXJ8           START 01/26/2007-10.30.00 INTERVAL 000.14.59   MODE = GOAL
                           CONVERTED TO z/OS V1R8 RMF       END   01/26/2007-10.44.59

                                        POLICY ACTIVATION DATE/TIME 10/25/2006 13.29.47

                                      ----------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          344K             344K        99.3        99.3  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
    <= 00.00.00.360          345K             741         99.5         0.2  >
    <= 00.00.00.420          345K             461          100         0.1  >
    <= 00.00.00.480          346K             324          100         0.1  >
    <= 00.00.00.540          346K             229          100         0.1  >
    <= 00.00.00.600          346K             151          100         0.0  >
    <= 00.00.00.660          346K             128          100         0.0  >
    <= 00.00.00.720          346K              86          100         0.0  >
    <= 00.00.00.780          346K              52          100         0.0  >
    <= 00.00.00.840          346K              36          100         0.0  >
    <= 00.00.00.900          346K              39          100         0.0  >
    <= 00.00.01.200          346K             102          100         0.0  >
    <= 00.00.02.400          347K              95          100         0.0  >
    >  00.00.02.400          347K             134          100         0.0  >
    ================================================================================================================ WORKLOAD

    REPORT BY: POLICY=WLMPOL01   WORKLOAD=CICS
                                       cics workload

    TRANSACTIONS   TRANS-TIME HHH.MM.SS.TTT  --DASD I/O--  ---SERVICE----  SERVICE TIMES  ---APPL %---  PAGE-IN RATES   ---STORAGE---
    AVG     41.00  ACTUAL               143  SSCHRT  2536  IOC     23626K  CPU   2079.9   CP    297.69  SINGLE     0.0  AVG  17179.02
    MPL     41.00  EXECUTION             62  RESP     1.7  CPU    341640K  SRB    607.7   AAPCP   3.00  BLOCK      0.0  TOT  704281.3
    ENDED  496115  QUEUED                 0  CONN     0.7  MSO      1444M  RCT      0.0   IIPCP   0.28  SHARED     0.0  CEN  704281.3
    END/S  551.25  R/S AFFIN              0  DISC     0.8  SRB     97488K  IIT     13.6                 HSP        0.0  EXP      0.00
    #SWAPS      0  INELIGIBLE             0  Q+PEND   0.3  TOT      1907M  HST      0.0   AAP     0.97  HSP MISS   0.0
    EXCTD  524444  CONVERSION             0  IOSQ     0.0  /SEC     2119K  AAP      8.7   IIP     1.47  EXP SNGL   0.0  SHR    227.98
    AVG ENC  0.00  STD DEV            1.130                                 IIP     13.3                 EXP BLK    0.0
    REM ENC  0.00                                          ABSRPTN    52K                               EXP SHR    0.0
    MS ENC   0.00                                          TRX SERV   52K
```

Appendix C. 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 16 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 16. Available year-end editions of our test report

Title	Order number	The year...	And these releases...	Softcopy collection kits
zSeries Platform Test Report	SA22-7997-04	2006	z/OS V1R8	SK3T-4269-18 SK3T-4271-18
zSeries Platform Test Report	SA22-7997-02	2005	z/OS V1R7	SK3T-4269-17 SK3T-4271-17
zSeries Platform Test Report	SA22-7997-00	2004	z/OS V1R6	SK3T-4269-15 SK3T-4271-15 SK3T-4269-14 SK3T-4271-14
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, *z/OS V1R8.0 System z Parallel Sysplex Recovery*, GA22-7286.

Appendix D. 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 17 lists some of the IBM Web sites that we reference in this edition or previous editions of our test report:

Table 17. 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/
z/OS 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/
z/OS.e Internet library	www.ibm.com/servers/eserver/zseries/zose/bkserv/

Other Web sites

Table 18 lists some other non-IBM Web sites that we reference in this edition or previous editions of our test report:

Table 18. 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 E. 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

V

W

Z

Readers' Comments — We'd Like to Hear from You

z/OS
System z Platform Test Report for z/OS and Linux Virtual Servers
Version 1 Release 8

Publication No. SA22-7997-05

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

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.

Comments:

Thank you for your support.

Submit your comments using one of these channels:

- Send your comments to the address on the reverse side of this form.

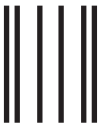
If you would like a response from IBM, please fill in the following information:

Name	Address
Company or Organization	
Phone No.	E-mail address

Readers' Comments — We'd Like to Hear from You
SA22-7997-05

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

SA22-7997-05