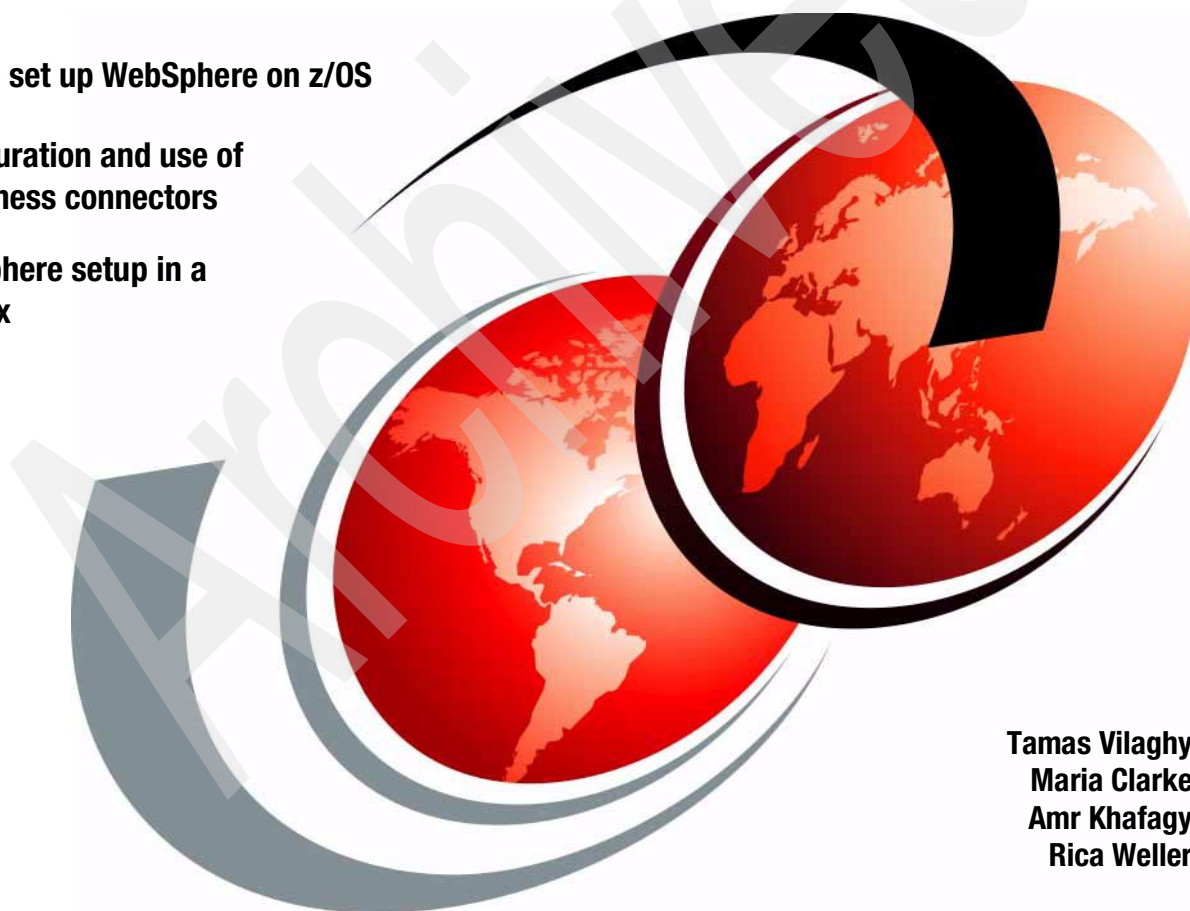


e-business Cookbook for z/OS Volume II: Infrastructure

How to set up WebSphere on z/OS

Configuration and use of
e-business connectors

WebSphere setup in a
sysplex



Tamas Vilaghy
Maria Clarke
Amr Khafagy
Rica Weller



International Technical Support Organization

**e-business Cookbook for z/OS Volume II:
Infrastructure**

July 2002

Archived

Take Note! Before using this information and the product it supports, be sure to read the general information in “Notices” on page xi.

Second Edition (July 2002)

This edition applies to Version 4 of Websphere Application Server for z/OS for use with the z/OS Operating System.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HYJ Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

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

© Copyright International Business Machines Corporation 2000, 2002

Note to U.S Government Users – Documentation related to restricted rights – Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	xi
Trademarks	xii
Preface	xiii
The team that wrote this redbook	xiii
Comments welcome	xv
Chapter 1. Environment description	1
1.1 z/OS software	2
1.1.1 Resource Recovery Services	2
1.1.2 System Logger	3
1.1.3 Workload Management (WLM)	3
1.1.4 RACF	4
1.1.5 TCP/IP	5
1.1.6 UNIX System Services	5
1.1.7 DB2	5
1.1.8 LDAP	6
1.2 Our environment	7
1.2.1 Initial configuration	7
1.2.2 Sysplex, the final configuration	8
Chapter 2. WebSphere Application Server	9
2.1 Introduction to WebSphere Application Server V4.0.1 for z/OS	10
2.1.1 IBM WebSphere Application Server product information	10
2.2 WebSphere configuration options	11
2.3 Planning the installation	13
2.4 Installation of the WebSphere Application Server	16
2.4.1 Installing the Software Development Kit (SDK)	16
2.4.2 Installing WebSphere Application Server for z/OS	17
2.5 Customization	18
2.5.1 “Before you start” checklist	18
2.5.2 WebSphere configuration	19
2.5.3 Installation Verification Program (IVP)	20
2.5.4 Starting WebSphere Application Server	21
2.5.5 Checking the environment	22
2.5.6 Enabling Web application runtime and IHS plug-in	25
2.6 Web Application runtime selection	36
2.7 System Management User Interface (SMUI)	41
2.7.1 Administration application	42

2.7.2 Operations application	44
2.8 System Management Scripting API (SMAPI)	45
2.9 WebSphere troubleshooting	47
2.9.1 WebSphere plug-in validation and basic debugging	47
2.9.2 WebSphere Application Server validation and basic debugging	48
2.9.3 Common configuration errors and the symptoms displayed	58
2.9.4 Turning on CTRACE	80
2.9.5 The WebSphere logstream	81
Chapter 3. WebSphere in a sysplex	83
3.1 Configuring WebSphere in a sysplex	84
3.1.1 Environment	84
3.1.2 Environment configuration checklist	84
3.1.3 Defining the new system to WebSphere	86
3.2 Migration to a new version in a sysplex	90
3.2.1 HFS structure	90
3.2.2 Migration procedure	92
3.3 Configuring Sysplex Distributor	94
3.3.1 Sysplex objectives	94
3.3.2 Sysplex Distributor implementation	98
3.3.3 Sysplex Distributor functionality	108
3.3.4 The big picture	110
3.4 Multiple WebSphere nodes in a sysplex	111
Chapter 4. Java-based access to DB2 for z/OS	117
4.1 JDBC and SQLJ—an introduction	119
4.1.1 JDBC	119
4.1.2 SQLJ	120
4.1.3 JDBC versus SQLJ	120
4.2 SQLJ/JDBC driver implementation	121
4.3 Software prerequisites	124
4.3.1 Prerequisites for JDBC	124
4.3.2 Prerequisites for SQLJ	125
4.4 Installation and configuration	125
4.4.1 Loading the JDBC and SQLJ libraries	126
4.4.2 Database and system administration	126
4.4.3 Setting the program control extended attribute (optional)	126
4.4.4 Setting the environment variables	127
4.4.5 Customizing the SQLJ/JDBC runtime properties file (optional)	130
4.4.6 Customizing the cursor properties file (optional)	134
4.4.7 Customizing the JDBC profile (optional)	135
4.4.8 Binding the DBRMs	135
4.4.9 Granting privileges	136

4.5	Preparing Java applications	137
4.5.1	JDBC application preparations	137
4.5.2	SQLJ application preparations	140
4.6	Executing Java applications	152
4.7	Special considerations for WebSphere for z/OS V4	155
4.7.1	HLQ for DB2 tables	156
4.7.2	Accessing data with WebSphere in previous versions of DB2	156
4.8	JDBC and SQLJ security issues	170
4.8.1	How authorization IDs are established	170
4.8.2	DB2 attachment types	171
4.9	SQLJ and JDBC performance considerations	171
Chapter 5.	WebSphere MQ	173
5.1	MQSeries for z/OS	174
5.1.1	Preparing your applications for the use of MQSeries	175
5.1.2	Interfacing with CICS, IMS, or batch	175
5.1.3	Planning to use MQSeries in a network	176
5.2	MQSeries classes	177
5.2.1	MQSeries classes for Java	177
5.2.2	MQSeries classes for Java Message Service (JMS)	178
5.2.3	Software prerequisites	178
5.2.4	Installation and configuration	179
5.2.5	Installation verification process	181
Chapter 6.	Java connectors for CICS	185
6.1	CICS Transaction Gateway	186
6.2	CTG APIs	186
6.2.1	EPI beans	187
6.3	CICS CCF connector	187
6.4	CICS Connector for CICS TS	188
6.5	JCA connectors	189
6.6	Enabling CICS JCA connector support	190
6.6.1	Preparing the CICS ECI resource adapter	190
6.6.2	Defining connection information	193
Chapter 7.	Java-based access to IMS	197
7.1	Introduction to IMS connector options	198
7.2	APPC-based access to IMS	198
7.2.1	Elements of the WebSphere IMSAPPC Connector	201
7.2.2	Preparations for setting up the IMS - APPC Procedural Application Adapter for WebSphere	202
7.2.3	Steps for setting up the WebSphere (local) side	204
7.2.4	Steps for setting up the IMS (partner) side	208
7.2.5	Using IMS Connector for Java based on APPC with WebSphere	213

7.2.6	Guideline for recovery of IMS transactions	216
7.2.7	When to choose APPC	216
7.3	OTMA Callable Interface	217
7.3.1	Software requirements for OTMA C/I installation	218
7.3.2	OTMA/CI initialization	219
7.3.3	OTMA/CI security	220
7.3.4	OTMA/CI restrictions	221
7.3.5	Compiling and link-editing requirements for OTMA/CI	221
7.3.6	Call functions implemented by OTMA/CI	221
7.3.7	Guidelines for using OTMA/CI with WebSphere Application Server on z/OS	222
7.3.8	When to choose OTMA/CI	224
7.4	MQSeries-IMS bridge	225
7.4.1	Introduction to the IMS bridge	225
7.4.2	Submitting IMS transactions from MQSeries	226
7.4.3	Submitting IMS transactions from a browser	226
7.4.4	Customizing the IMS bridge	227
7.4.5	Controlling the IMS bridge and IMS connections	229
7.4.6	Security considerations for using MQSeries with IMS	231
7.4.7	When to use the MQSeries-IMS bridge	233
7.5	IMS Connect	234
7.5.1	IMS Connect overview	234
7.5.2	Installing IMS Connect V7	236
7.5.3	Considerations for the IMS Connect configuration	237
7.5.4	Configuring IMS Connect	238
7.5.5	Customizing IMS Connect	243
7.5.6	IMS Connect Local Option support	246
7.5.7	Running the Installation Verification Program (IVP)	250
7.5.8	Confirming IMS Connect install with the sample Java client	251
7.5.9	Starting an IMS Connect trace	257
7.6	IMS Connector for Java based on IMS Connect	258
7.6.1	J2EE Connector support in IMS Connector for Java	259
7.6.2	Elements of the IMS Connector for Java	260
7.6.3	Prerequisites for using IMS Connector for Java	262
7.6.4	Preparing to use IMS Connector for Java	264
7.6.5	Preparing your WebSphere Application Server environment	269
7.6.6	References	282
Appendix A. z/OS UNIX System Services enablement		283
Installing z/OS UNIX		284
Hierarchical File System concepts		284
Methods of installation		284
Customizing z/OS UNIX		285

Use the z/OS UNIX Configuration Wizard.	286
Perform the security preparation.	286
Customize BPXPRMxx PARMLIB members.	288
Allocating other file systems	291
Defining BPXPRMxx PARMLIB members in IEASYSxx.	293
Customizing other PARMLIB members	293
Preparing PROCLIB members	296
Adding z/OS UNIX ISPF data sets	297
Customizing the shell	297
Prioritizing the kernel.	298
Creating HFS data sets for z/OS UNIX users	304
Checking for setup errors	305
z/OS UNIX security	305
Defining a superuser.	308
Security requirements for installation/applying maintenance	311
Defining RACF group identifiers and user identifiers	311
Security verification.	313
Setting up a default OMVS segment.	314
Setting up BPX.* FACILITY class profiles.	317
Setting up z/OS UNIX daemons.	319
Setting up security.	319
Setting up HFS control	321
Daemon start options	322
Evaluating virtual storage needs	322
Appendix B. TCP/IP enablement	327
Customize the TCP/IP basic configuration.	328
Customize TCP/IP configuration files	328
Customizing other TCP/IP files	333
Setting up the resolver address space	335
Customizing the TCP/IP started task procedure	338
Customizing SYS1.PARMLIB members	338
Security definitions for TCP/IP	343
Starting TCP/IP	346
Adding the new host to your DNS server	347
Testing your TCP/IP setup and connection.	347
Customizing IBM-supplied daemons	348
Customizing inetd	350
Customizing the syslog daemon	354
Customizing FTP.	355
Appendix C. DB2 quick setup	359
Overview	360

WebSphere Application Server use of DB2	360
DB2 pre-installation considerations	360
DB2 installation in a sysplex	361
Editing and running the customized JCL	366
Building the second, and subsequent, data sharing systems	371
Other dialog parameters	372
Appendix D. Sample files	377
simple.sh	378
simple.sh template	378
simple.sh for JDBC	379
simple.sh for SQLJ	380
db2jdbcsqlj.properties	380
db2jdbc.properties	380
db2sqlj.properties	381
BIND and INSERT jobs	381
Sample JDBC BIND job for a local DB2	381
Sample JDBC BIND job for a remote DB2	381
Sample SQLJ BIND job for a local DB2	382
Sample SQLJ BIND job for a remote DB2	383
Sample INSERT job	383
Output from the INSERT job	384
Java and SQLJ samples	384
newsample01.java	384
Output from newsample01 with local DB2	386
newsample02.sqlj	387
Output from newsample02 with local DB2	389
Appendix E. Command reference	391
Appendix F. WLM quick setup	399
Appendix G. RACF REXX sample	413
Appendix H. Additional material	419
Locating the Web material	419
Using the Web material	419
System requirements for downloading the Web material	420
How to use the Web material	420
Related publications	421
IBM Redbooks	421
Other resources	422
Referenced Web sites	423

How to get IBM Redbooks 424

 IBM Redbooks collections..... 424

Index 425

Archived

Archived

Notices

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

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

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

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

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

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

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

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

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


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

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which 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 the International Business Machines Corporation in the United States, other countries, or both:

Redbooks(logo)TM 

Balance[®]

C/MVSTM

CICS[®]

DB2[®]

DB2 Universal DatabaseTM

DeveloperWorksTM

DFSTM

DFSMS/MVS[®]

Distributed Relational Database

ArchitectureTM

DRDA[®]

IBM[®]

IBM.COMTM

IBMLinkTM

IMSTM

IMS/ESA[®]

iSeriesTM

Language Environment[®]

MQSeries[®]

MVSTM

MVS/ESATM

OS/2[®]

OS/390[®]

OS/400[®]

PAL[®]

Parallel Sysplex[®]

PerformTM

RACF[®]

RETAIN[®]

RMFTM

S/390[®]

SecureWay[®]

SPTM

SP1[®]

SP2[®]

SupportPacTM

System/390[®]

TME[®]

VisualAge[®]

VTAM[®]

WebSphere[®]

z/OSTM

zSeriesTM

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

Lotus[®]

Lotus Notes[®]

Notes[®]

Word Pro[®]

The following terms are trademarks of other companies:

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

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

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

C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

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

Preface

This IBM Redbook is the second volume of a series entitled the e-business Cookbook for z/OS. It is intended for System Programmers involved in setting up and configuring an environment to run Java-based e-business applications on z/OS and OS/390.

We focus on the most popular and viable application topologies. Software products covered are z/OS UNIX System Services, TCP/IP, IBM HTTP Server for z/OS, IBM WebSphere Application Server for z/OS V4, and the Java connectors for DB2, CICS, MQSeries and IMS. We also detail how to set up and run WebSphere in a Parallel Sysplex environment.

The information in this book will help you configure an environment on z/OS that will be ready to implement end-to-end e-business solutions using Java, WebSphere and traditional backend systems. We also provide information on security aspects and performance optimization regarding many of the products.

The other volumes in this series are *e-business Cookbook for z/OS Volume I: Technology Introduction*, SG24-5664, and *e-business Cookbook for z/OS Volume III: Java Development*, SG24-5980.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Tamas Vilaghy is a project leader at the International Technical Support Organization, Poughkeepsie Center. He leads redbook projects dealing with e-business on zSeries servers. Before joining the ITSO recently, he worked in System Sales Unit and Global Services departments of IBM Hungary. Tamas spent two years in Poughkeepsie from 1998 to 2000 dealing with zSeries marketing and competitive analysis. From 1991 to 1998 he held technical, marketing and sales positions dealing with zSeries.



Maria Clarke is an e-business Specialist in IBM Global Services Australia. She has 16 years of experience in supporting OS/390, primarily as a Systems Programmer. This includes four years of UNIX System Services and OS/390 e-business support. More recently she worked in e-business enablement services supporting WebSphere across multiple platforms. She holds a Bachelor of Science degree in Computing/Mathematics from Monash University. Maria is an IBM Certified Systems Expert in “Administration for IBM WebSphere Application Server”.



Amr Khafagy is a Senior Technical Services Professional in IBM Canada. He has 16 years of experience in MVS, OS/390 systems programming and UNIX System Services. Amr participated in the previous version of the *e-business Cookbook Volume II*. He holds a Bachelor's degree in Engineering.



Rica Weller is a S/390 IT Specialist with the WebSphere and MQWorkflow Competence Team in the Boeblingen lab in Germany. She holds a Master's degree in Management Systems from Massey University in New Zealand and a postgraduate Diploma degree in Business Administration from the University of Technology, Dresden, Germany. Before joining IBM in 1999, she worked for several IT companies developing logistics models. With IBM, she focuses on technical support for S/390 and WebSphere/390.



Thanks to the following people for their contributions to this project:

- ▶ Ivan Joslin, Dave Cohen, Mary Ellen Kerr, David Booz, Satish K Mamindla, Louis A. Amodeo
IBM WebSphere for z/OS and OS/390 Development Lab, Poughkeepsie
- ▶ Alex Louwe Kooijmans, WebSphere and Java specialist, the project leader for the first edition of the Cookbook
- ▶ Mitch Johnson
Software Group, WebSphere Services

- ▶ Rich Conway, ITSO, who installed and maintained our system, and wrote the DB2 quick setup appendix
- ▶ Holger Wunderlich, ITSO project leader
- ▶ Phil Wakelin, project leader
International Technical Support Organization
- ▶ Don Bagwell, who provided his whitepaper about the troubleshooting tips regarding WebSphere
Washington Systems Center
- ▶ Hilon Potter
IBM eTP Center, Poughkeepsie

Comments welcome

Your comments are important to us!

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

- ▶ Use the online **Contact us** review redbook form found at:
ibm.com/redbooks
- ▶ Send your comments in an Internet note to:
redbook@us.ibm.com
- ▶ Mail your comments to the address on page ii.

Environment description

WebSphere has some specific environmental requirements. In this chapter we provide brief explanations of the z/OS components that WebSphere relies on.

We do not explain how to configure z/OS products because that is beyond the scope of this redbook. Instead, we give a brief description of products used by WebSphere and how they are used. We describe the environment that we used for configuring WebSphere. Throughout this book, a reference to WebSphere relates to “WebSphere Application Server for z/OS.”

1.1 z/OS software

When you configure the environment prior to installing and configuring WebSphere, you need to carefully consider some of the settings you make. We have documented these in the 2.3, “Planning the installation” on page 13 and recommend that you read this before starting your installation. A poor choice early on can be difficult to correct later.

WebSphere uses many of the features of z/OS. We used z/OS 1.2 and WebSphere 4.0.1. Most references to z/OS also apply to OS/390 R10. Some z/OS products may already be configured by others before you start installing WebSphere, other products you will need to configure yourself. The following is a brief description of z/OS products that WebSphere relies on.

1.1.1 Resource Recovery Services

Resource Recovery Services (RRS) consists of the protocols and program interfaces that allow an application program to make consistent changes to multiple protected resources.

RRS can coordinate changes to one or more protected resources, which can be accessed through different resource managers and reside on different systems. z/OS ensures that all changes are made or no changes are made, which is called a rollback.

The Object Transaction Service (OTS), which is a Communication Manager related to CORBA manages transactions and provides the interface to z/OS RRS, which coordinates resource recovery across several Resource Managers. On the Server Region side, this is a very thin layer that delegates requests to the Control Region. OTS requires the existence of RRS, which requires the definitions of logstreams in a Couple data set.

The IMS, CICS and DB2 Resource Managers have also implemented an interface to RRS so that, in the case of a WebSphere client application, resource coordination between the various Resource Managers can be driven by RRS. When the WebSphere client application completes its work and issues a commit, the Control Region notifies RRS of this event and the two-phase commit process begins. The client is notified about the outcome and may then continue with the next transactional Unit of Work (UOW).

The activation of RRS implies the use of the RRS Attachment Facility (RRSAF) when using DB2.

1.1.2 System Logger

System Logger is a z/OS component that allows an application to log data from a sysplex. You can log data from one system in a sysplex or from multiple systems across the sysplex. A System Logger application writes log data into a log stream (in the Coupling Facility or DASD). A Coupling Facility log stream can contain data from multiple systems, allowing a system logger application to merge data from systems across the sysplex. In a DASD-only log stream, interim storage for log data is contained in local storage buffers on the system.

All compatible Resource Managers in the sysplex have access to the log stream structure in the Coupling Facility. The System Logger implementation provides for fault tolerant management of log data. There is no single point of failure if you have two CFs installed. While the log data is passed quickly and efficiently to the Coupling Facility, it is also logged on the generating system in a staging file or data space. Periodically, when the log threshold is met, the logs are “off-loaded” to VSAM linear data sets. This process ensures integrity of the log data while providing a highly efficient recovery mechanism for sysplex-wide resources.

If you plan to configure WebSphere to run in a sysplex, you need to configure the system logger to use the Coupling Facility log stream.

1.1.3 Workload Management (WLM)

With workload management, you define performance goals and assign a business importance to each goal. You define the goals for work in business terms, and the system decides how much resource, such as CPU and storage, should be given to it to meet the goal. Workload Manager will constantly monitor the system and adapt processing to meet the goals you set.

WebSphere requires that WLM is running in goal mode and Application Environments (AE) are defined. To balance work on the system, WebSphere uses the services of the WorkLoad Manager (WLM) provided by z/OS and z/OS. Three distinct WLM services are employed by WebSphere:

1. Routing

The WLM routing service is used to direct clients to servers on a specific z/OS server, based on a measurement of current system utilization known as the performance index.

2. Queuing and address space management

The WLM queuing service is used to dispatch work requests from the Websphere Application Server for z/OS Control Region to one or more Websphere Application Server for z/OS Server Regions. It is possible for a Work Manager to register with WLM as a “queueing manager” (e.g.,

WebSphere Application Server for z/OS Control Region is a Queue Manager). This tells WLM that this server would like to use WLM-managed queues to direct work to other servers, which allows WLM to manage server spaces to achieve the specified performance goals established for the work.

3. Prioritizing work to meet performance goals

WebSphere delegates the responsibility for starting and stopping Server Regions to the WLM Address Space management service. This allows WLM to manage application server instances in order to achieve the performance goals specified by the business.

WebSphere requires that WLM is running in goal mode. Refer to Appendix F, “WLM quick setup” on page 399 for details on how to configure WLM.

1.1.4 RACF

OS/390 and z/OS systems are famous for robustness and security. The z/OS operating system security service interface is known as the System Authorization Facility (SAF). All security requests go through this facility. Conditionally, the call may be directed to Resource Access Control Facility (RACF), or ISV-/user-supplied security manager, or both.

This is the mechanism by which security managers provided by other vendors are implemented on z/OS. The SAF interface is published and is the security focal point for all products providing resource access control. The heart of security in WebSphere is an external security server, such as RACF. As a consequence, many authorization actions have to be defined.

Roughly the following definitions have to be made.

All entities or principals, internally or externally have to be authenticated in an implicit or explicit way. This statement is *not only* valid for individuals but also for the Control and Server Regions running under WebSphere. The identities of the system regions are defined during the setup of the system. When new Control and Server Regions are added for J2EE applications, a number of IDs have to be decided upon and defined in RACF.

Using the user IDs that have been defined many authorizations in several RACF classes have to be granted:

- ▶ Association between started task procedures and user IDs via the STARTED class
- ▶ Access to servers via the CBIND class
- ▶ Access to DB2 via the DSNR class
- ▶ Access to logstream via the LOGSTREAM class

- ▶ Access for Server Regions to Control Regions via the SERVER class
- ▶ In case WebSphere J2C connectors are used:
 - class for access to CICS
 - class for access to IMS

If you plan to run WebSphere in a sysplex you should ensure that the RACF database is shared between the systems in the sysplex, because these permissions must be consistent across the sysplex.

1.1.5 TCP/IP

Communication Server for z/OS and OS/390 contains TCP/IP.

IIOP, FTP and other TCP/IP-based protocols are used by WebSphere. Refer to Appendix B, “TCP/IP enablement” on page 327 for detailed information on TCP/IP.

1.1.6 UNIX System Services

WebSphere requires that z/OS UNIX System Services be enabled and configured. Typically, this will already have been done, but you can refer to Appendix A, “z/OS UNIX System Services enablement” on page 283 for assistance. WebSphere requires a Hierarchical File System (HFS) with read/write access. In a sysplex configuration all the systems running WebSphere need shared access to the WebSphere configuration HFS. In a sysplex configuration the recommendation is to use shared HFS. Until OS/390 V2R8 this could only be achieved by using Network File System (NFS). For OS/390 V2R9 and later and z/OS, a shared HFS function is provided. Now you can use either NFS or the shared HFS function.

1.1.7 DB2

WebSphere V4 requires the use of DB2 UDB for OS/390 and z/OS V7.1. WebSphere configuration and session control information is held in DB2 databases. The control information of WebSphere consists of about 60 tables in two databases: BBOMDB01, BBOJDB01.

This requirement is reflective of the close integration required between application servers and datastores when providing an Enterprise Java Beans (EJBs) execution environment (also known as an EJB container).

DB2 is also the default database for the LDAP server.

If you run WebSphere in a sysplex and share the workloads, then configure DB2 for z/OS and OS/390 in data sharing mode, which requires the Coupling Facility (CF). Refer to Appendix C, “DB2 quick setup” on page 359 for more information on configuring DB2.

1.1.8 LDAP

The z/OS Lightweight Directory Access Protocol (LDAP) server, part of the SecureWay Security Server for z/OS, is based on a client/server model that provides client access to an LDAP server. One or more LDAP servers contain the data making up the LDAP directory tree. An LDAP client application connects to an LDAP server using LDAP APIs and asks it a question. The server responds with the answer, or with a pointer to where the application can get more information (typically, another LDAP server).

An LDAP directory provides an easy way to maintain directory information in a central location for storage, update, retrieval, and exchange. A directory is like a database, but tends to contain more descriptive, attribute-based information. LDAP is a directory service protocol that runs over TCP/IP. Refer to *SecureWay Security Server LDAP Server Administration and Use*, SC24-5923 for more information.

LDAP provides the directory services for the Java Naming and Directory Interface (JNDI) and CORBA (referred to as Managed Object Framework, or MOFW) naming and interface repository services. The contents of the directory are stored in a DB2 table.

LDAP is used by both the “system servers”, such as the BBOSM Systems Management Server, and the J2EE servers, but in different ways.

- ▶ The Systems Management Server (SMS) and Interface Repository (IR) regions include DLL code to directly access naming information in the LDAP database.
- ▶ J2EE servers access the LDAP database through the LDAP server with the use of the client/server Java Naming Directory Interface (JNDI). The name service is mapped to the X.500 data model in the LDAP server on z/OS. Note that the LDAP server is built on the data sharing implementation of DB2. This makes the Object Management Group (OMG) Name Service implementation on z/OS fully transactional and recoverable. The Name Server also provides the residence for the lifecycle-related objects, such as Factory Finders and location objects.

1.2 Our environment

We ran z/OS V1R2.0. We used two different TCP/IP addresses to access USS and TSO through Telnet.

1.2.1 Initial configuration

In Chapter 2, “WebSphere Application Server” on page 9 we install WebSphere into the first system of a sysplex. The procedure is the same as for a monoplex. We already had DB2 data sharing, RRS, WLM and LDAP configured to use the Coupling Facility, because in Chapter 4, “Java-based access to DB2 for z/OS” on page 117 we plan to move WebSphere to a sysplex. Our initial configuration looked like Figure 1-1.

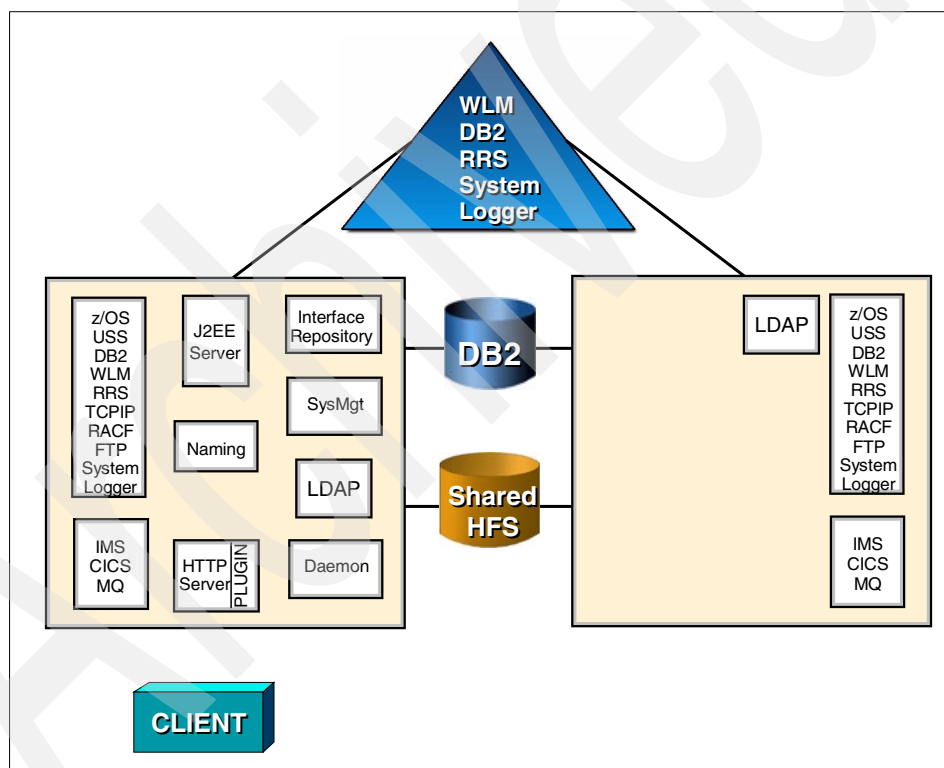


Figure 1-1 WebSphere environment

1.2.2 Sysplex, the final configuration

In Chapter 3, “WebSphere in a sysplex” on page 83 we install WebSphere into the second system in a sysplex. The procedure is the same as for installation into subsequent systems in the sysplex. In the sysplex we decided to run the Sysplex Distributor on a different IP stack, which we called TCPIPF. Sysplex Distributor acts as a workload balancing tool in front of the WebSphere servers. Our final configuration looked like this:

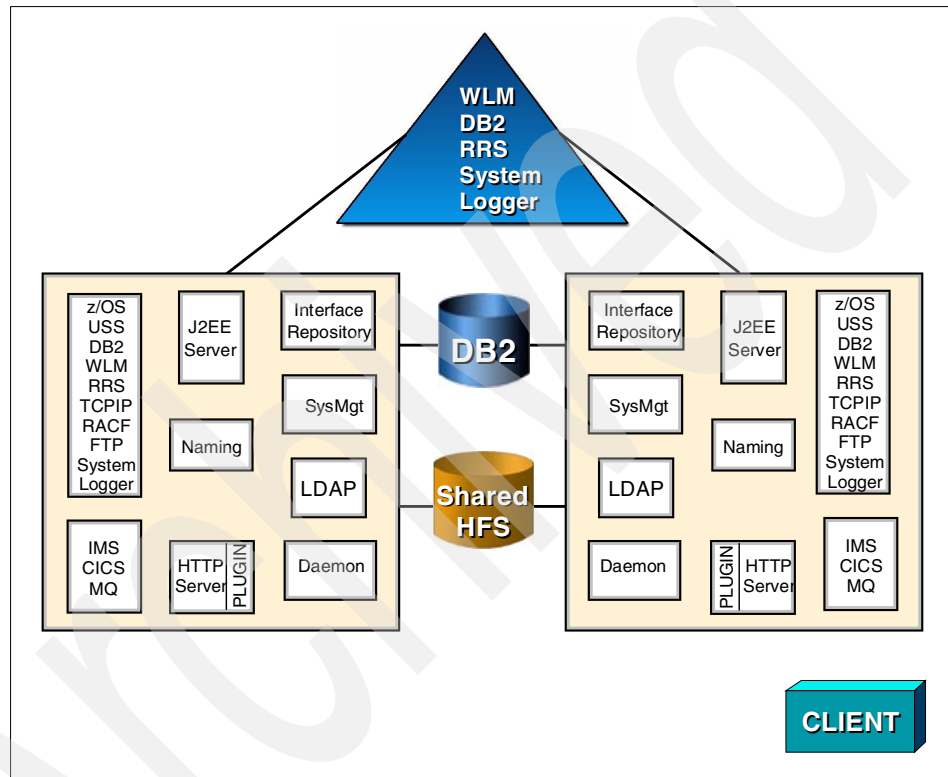


Figure 1-2 WebSphere sysplex environment

WebSphere Application Server

This chapter is based on WebSphere Application Server Version 4.0.1. The IBM HTTP Server for z/OS is also referred to as the *HTTP server*. The IBM WebSphere Application Server for z/OS and OS/390 is referred to as *WebSphere*.

In this chapter we discuss:

- ▶ Introduction to WebSphere Application ServerV4.0.1 for z/OS and OS/390
- ▶ WebSphere configuration options
- ▶ Planning the installation
- ▶ Customization
- ▶ Web Application runtime selection
- ▶ System Management User Interface (SMUI)
- ▶ System Management Scripting API (SMAPI)
- ▶ WebSphere troubleshooting

2.1 Introduction to WebSphere Application Server V4.0.1 for z/OS

WebSphere Application Server is an e-business application deployment environment. It is built on open, standards-based technology, such as CORBA, HTML, HTTP, IIOP, and J2EE-compliant Java technology standards for servlets, Java Server Pages technology (JSP), and Enterprise Java Beans (EJB), and supports all Java APIs needed for J2EE compliance.

You can build and run mission-critical, object-oriented applications across an enterprise and beyond. You can conduct e-business over the Web because the Application Server supports multiple client platforms.

You can even integrate new functions with existing z/OS and OS/390 applications and data. Applications and their components can be deployed globally on any WebSphere-supported server. It enables new e-business applications and transactions to be scaled up for deployment on the server of choice, regardless of the development platform.

WebSphere continues to support connector access to CICS, IMS, and DB2.

If you don't need full J2EE support, you can configure WebSphere Application Server V4.0.1 very simply without requiring the functions of LDAP, workload management goal mode, or DB2 V7. (DB2 V5 is required for some items such as session state.) In this case you can run your non-J2EE-compliant applications in the WebSphere plug-in that is loaded into the HTTP server. This operation mode is called the Alternate configuration option. For more information refer to:

http://www.ibm.com/software/webservers/appserv/zos_os390/about.html

2.1.1 IBM WebSphere Application Server product information

Prior to starting the basic setup of the application server, you need to review the following documentation.

- ▶ For WebSphere installation information, see the following documents:
 - *WebSphere Applications Server V4.0.1 for z/OS and OS/390 Program Directory*, which is shipped with the WebSphere product.
 - *WebSphere Application Server V4.01 for z/OS and OS/390 Installation and Customization*, GA22-7834.
 - If you are migrating from previous versions of IBM WebSphere Application Server, refer to *WebSphere Application Server V4.01 for z/OS and OS/390 Migration*, GA22-7860.

- For the most current version of related product documentation, go to the application server Web site library page at:

http://www.ibm.com/software/webservers/appserv/zos_os390/library.html

- ▶ Check the updates on the Web site.

For the most current application server documentation and updates, go to the Web site at:

<http://www.ibm.com/software/webservers/appserv/library.html>

- ▶ Check the related documentation:

- For additional related documentation that is frequently used, see the Bibliography in “Related publications” on page 421.
- For the latest application server product offerings, information and news, go to the Web site at:

http://www.ibm.com/software/webservers/appserv/zos_os390/

- ▶ The WebSphere Troubleshooter for OS/390 is available on the Web and provides the most current hints and tips (debugging, tuning, and browser) for the application server and the HTTP server. To access the Troubleshooter, go to the Web site at:

<http://www.ibm.com/software/webservers/appserv/troubleshooter.html>

Java documentation

For documentation on Java for OS/390, go to the following Web site:

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

To obtain any documentation regarding Java, go to the Web site at:

<http://www.javasoft.com>

IBM HTTP Server (IHS)

For the most current HTTP server documentation and information updates, go to the Web site at:

<http://www.ibm.com/software/webservers/httpservers/doc53.html>

For this redbook, we used the IBM HTTP Server for z/OS Version 5.3. Refer to *HTTP Server Planning Installing and Using*, SC34-4826 for more information.

2.2 WebSphere configuration options

We describe four possible configuration options to drive EJBs using the HTTP protocol. We show three in Figure 2-1 on page 12.

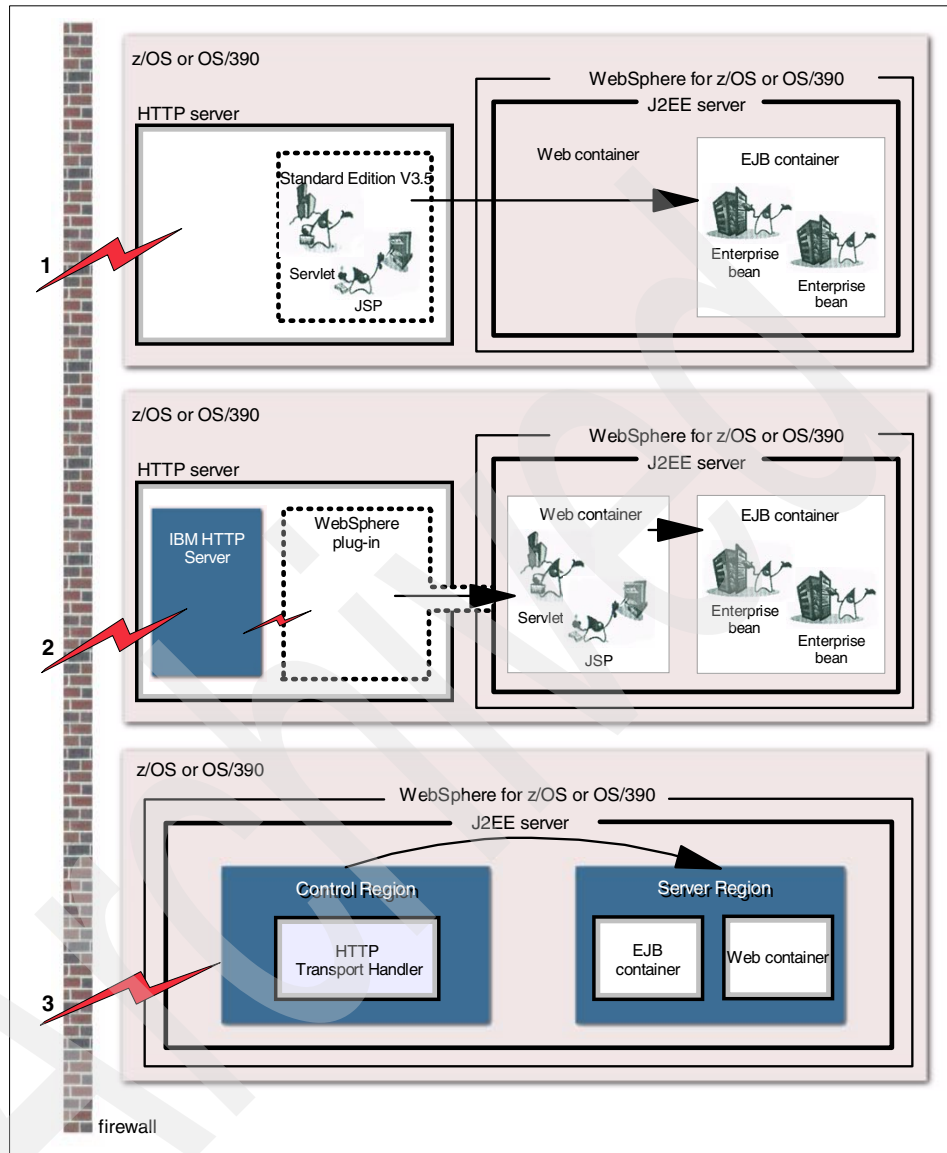


Figure 2-1 WebSphere configuration options

In the first configuration, the IBM HTTP Server handles the inbound request, and the WebSphere Application Server Standard Edition acts as runtime environment for Web Applications. Within this runtime, servlets can make use of the RMI/IOP protocol to access enterprise beans in the WebSphere for z/OS J2EE server.

Scenario 1 in Figure 2-1 on page 12 shows this interaction between servlet and bean within the same z/OS or OS/390 image. You can vary this scenario to run the EJB-triggering Web Application anywhere (for example, on IHS Linux in a DMZ).

In the second configuration, the IBM HTTP Server handles the inbound request, the WebSphere Application Server for z/OS plug-in routes the requests, and the WebSphere for z/OS J2EE server is the execution environment. In this configuration, servlets run in the J2EE server's Web container, and use the IIOP protocol to access enterprise beans in the J2EE server's EJB container. Figure 2-1 on page 12 shows this connection between servlet and bean within the same J2EE server, but this connection can occur between different J2EE servers on the same or different z/OS or OS/390 images. The plug-in maintains a table that describes all successfully deployed Web Application in all the Web containers. You can also run Web Applications within the plug-in.

In the third configuration, the HTTP Transport Handler component that exists in the control region of a J2EE Server, is enabled, so the control region listens to HTTP requests directly. Similar to EJB requests, the control region marshals the input request and routes it to the Web container in the appropriate server region for processing.

If APAR PQ57888 (PTF AQ64031) is installed, the HTTP Transport Handler supports HTTP session affinity within a server instance. This means that even if an HTTP session is maintained in memory, the control region is able to route back to the server region where the previous interaction with the same client occurred and session data is maintained. This increases scalability, as multiple server regions can exist even if HTTP session data is maintained in memory.

There is a fourth configuration, which is called "alternate configuration option". In this case the HTTP Server runs the V3.5 plug-in, where Web applications can be run. These Web applications can use the RMI/IIOP protocol to access EJBs in the WebSphere for z/OS V4 J2EE server.

2.3 Planning the installation

In our project we installed WebSphere Application Server into all the systems in a sysplex. We assumed that basic sysplex functionality such as WLM, shared DB2 and so forth were available. Details of the environment we started with are described in Chapter 1, "Environment description" on page 1.

Detailed planning information can be found in Chapter 2 of *WebSphere Application Server V4.01 for z/OS and OS/390 Installation and Customization*, GA22-7834. In addition, you should consider all of the following issues before starting to install and configure WebSphere Application Server in a sysplex.

- ▶ View error logs

If you want to get WebSphere errors from all the systems in the same error log, you should set up the system logger to use the coupling facility.

- ▶ WebSphere Application Server product HFS

When we refer to the WebSphere product HFS, we are talking about the HFS supplied with WebSphere and populated by SMP/E. It contains executables and other files distributed with the WebSphere product and is replaced when you upgrade WebSphere. The WebSphere product HFS can be shared between all the systems in the sysplex. If you do this, you need to consider how you plan to migrate to new versions of WebSphere. 3.2, “Migration to a new version in a sysplex” on page 90 provides a sample migration strategy.

- ▶ WebSphere Application Server configuration HFS

The WebSphere configuration HFS is created as part of the installation process. It is empty until it is populated with local configuration information. This directory (HFS mountpoint) must be shared between all the systems in the sysplex running WebSphere Application Server. To share the configuration directory, you can use either NFS or the shared HFS function (OS/390 2.9 and above).

- ▶ ARM may cause problems during the installation process and should be disabled for WebSphere servers during this time.

- ▶ Server replication

You need to consider on which systems your applications will be running in the sysplex. On every system? Only on some? Do you plan to do non-disruptive upgrades? We chose to replicate every server on every system in the sysplex. This would give us maximum flexibility and availability, and allow us to upgrade non-disruptively.

- ▶ Security

Shared RACF needs to be configured in the sysplex. *WebSphere Application Server V4.01 for z/OS and OS/390 Installation and Customization*, GA22-7834 provides details on what needs to be set up for WebSphere in RACF. Also, the ISPF configuration dialog provides a sample job to create all the profiles required initially.

- ▶ There are a few configuration files, not directly related to WebSphere Application Server, that you need to configure—like the IHS config files. We recommend that you consolidate these to one directory. This directory is not

required to be shared by all systems in the sysplex, but we recommend that you put these files in the WebSphere Application Server configuration HFS.

- If, in your installation, these configuration files are not only used by WebSphere Application Server, you might need to consider a different solution. This HFS will not exist until the configuration process is finished but we only need to reference the file names at this time.

```
e.g.: /WebSphere390/httpd.conf  
      /WebSphere390/httpd.conf  
      /WebSphere390/sc63_db2sqljjdbc.properties  
      /WebSphere390/sc64_db2sqljjdbc.properties
```

- DB2 data sharing

WebSphere uses shared DB2 tables to hold its configuration. Although applications can run anywhere, their data must also be accessible. Refer to Appendix C, “DB2 quick setup” on page 359.

For DB2, there are a number of recommendations in Chapter 2 of *WebSphere Application Server V4.01 for z/OS and OS/390 Installation and Customization*, GA22-7834. We found the following particularly important:

- Create a BP32K buffer pool in the coupling facility for every system in the sysplex. There must be at least 100 buffers for every system.
- Do not make the DB2 location name the same as one of the DB2 subsystem IDs. This is confusing and makes the configuration of WebSphere on subsequent systems more difficult.
- The db2sqljjdbc.properties file is unique for every system in the sysplex. If you plan to keep all your configuration files together in a shared HFS, then you need to copy this file and rename it to a system-specific name, for example /WebSphere390/sc63_db2sqljjdbc.properties. Make a note of the location—you will need to specify it in the DB2SQLJPROPERTIES environment variable when you define a J2EE server instance.

- TCP/IP DNS names

The Daemon IP Name is used for the DNS resolution of the Daemon Server. Its notation is like any DNS naming entry, but has to be chosen carefully. You can choose any name you want, but, once chosen, it is difficult to change.

Note: We strongly recommend that you choose a generic or flexible value for Daemon IP Name, as this makes it easier if you intend to expand your WebSphere applications beyond the monoplex to a full sysplex configuration.

All communication to WebSphere on *any* system in the sysplex is through a generic Daemon IP Name. This value has to be the same for every system in the sysplex and is the same as the Resolve IP Name.

You need to add a new DNS entry for the Daemon IP Name that reflects the WebSphere sysplex environment, i.e., `wasplex2.itso.ibm.com`. The value that the Daemon IP Name resolves to will depend on your DNS configuration. For a DNS lookup it will resolve to the IP address of the first system on which you install WebSphere. If you plan to use Sysplex Distributor, it will resolve to the DVIPA address. Refer to 3.3, “Configuring Sysplex Distributor” on page 94 for more detail.

2.4 Installation of the WebSphere Application Server

Prior to customizing WebSphere Application Server for z/OS, the Java Software Development Kit (SDK) needs to be installed.

2.4.1 Installing the Software Development Kit (SDK)

Before customizing WebSphere you need to install the Java SDK, known as the IBM Developer Kit for OS/390 Java 2 Technology Edition (5655-D35). It is available as an element of WebSphere for z/OS, but is also available separately. The SDK level supported by this product is 1.3.1. Make sure you get APAR PQ57514 to get the latest JAVA SDK level.

For information on installing the SDK, refer to the instructions provided with the product. For more information, refer to *Java Programming Guide for OS/390*, SG24-5619.

You can download the SDK from the Java for OS/390 Web site at:

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

The Java 1.3.0 SDK (FMID HJVA130), included as part of WebSphere Application Server Version 4.0.1, is identical to the version available separately as IBM Developer Kit for OS/390 Java 2 Technology Edition Version 1.1.

Whether installed as a standalone product or as part of WebSphere, the SDKs are normally installed under `/usr/lpp/java`.

When you download the SDK from the Internet and install it without SMP/E directly into the HFS, it is possible that subsequent installation or maintenance of Java may have unintended results.

In order to guarantee that the Java for OS/390 that is installed as part of WebSphere is correctly positioned on your system, the following is a possible solution:

- ▶ Allocate a separate HFS data set to contain the Java directories that are created as part of the WebSphere 4.0.1 installation.
- ▶ During the installation of WebSphere, mount this HFS data set at a service mount point such as /service/usr/lpp/java.

Then unpack the SDK to the installed system at /service/usr/lpp/java.

- ▶ This can be done via an SMP/E **apply** of the Java FMID or by using the **pax** command as described in the Java documentation. The documentation is available from the Web site you downloaded the SDK from.

Attention: It is important to ensure that you don't have a prior level of Java mounted at this mount point, since the maintenance job will remove the whole directory structure.

- ▶ When the installation is complete, mount a copy of this HFS to your target system at mount point /usr/lpp/java. This will make previously installed versions of Java at this mount point unavailable. An alternative solution is described in 3.2.1, "HFS structure" on page 90 for a non-disruptive migration strategy when running in a sysplex environment.

2.4.2 Installing WebSphere Application Server for z/OS

Refer to the *WebSphere Application Server V4.0.1 for z/OS and OS/390: Program Directory*, downloadable from:

http://www-3.ibm.com/software/webservers/appserv/zos_os390/library.html

for a description of the SMP/E installation process.

WebSphere Application Server Version 4.0.1 can be obtained as an SMP/E-installable product for use with OS/390 Release 8 or higher and z/OS 1.1 or higher.

The WebSphere Application Server 4.0.1 for z/OS must be installed into an SMP/E environment separate from that on z/OS. This includes SMP/E target and distribution zones, as well as the HFS data set. See Section 6.0, "Installation Instructions," in *IBM WebSphere Application Server Version V4.0.1 Program Directory* for more information on this topic.

For WebSphere Application Server 4.0.1, the LTS data set must be in PDSE format. This is also documented in the *Program Directory*: Section 5.2.3, DASD Storage Requirements on page 19. If not done, you might see the following error messages:

```
GIM23901E  ** LINK-EDIT PROCESSING FOR SYSMOD xxxxxxxx FAILED FOR MODULE  
yyyyyyyy IN LMOD zzzzzzzz IN THE SMPLTS LIBRARY.  
IEW2606S 4B39 MODULE INCORPORATES PROGRAM MANAGEMENT 3 FEATURES AND CANNOT BE  
SAVED IN LOAD MODULE FORMAT.
```

Tip: Even if you usually don't look in the *Program Directory* for the DASD space requirements of a program product, the WebSphere Application Server Version 4.0.1 is a very large product, so you need to make sure that there is sufficient space in all target and temporary data sets for receive and apply processing.

2.5 Customization

We now describe how to customize WebSphere.

2.5.1 “Before you start” checklist

This is a concise list of what you need before you start configuring WebSphere. Refer to *WebSphere Application Server V4.01 for z/OS and OS/390 Installation and Customization*, GA22-7834 for more detail.

- ▶ Verify that the basic environment is configured and available.
 - WLM is running in goal mode.
 - Shared HFS or NFS is available (for sysplex environments).
 - DB2 is in data sharing mode (for sysplex environments). Refer to Appendix C, “DB2 quick setup” on page 359.
 - Configure a shared RACF database (for sysplex environments). WebSphere assumes that a user ID represents the same user identity on all systems in the sysplex.
 - RRS can be configured now or the WebSphere customization dialogs provide sample code to be run later.
 - System Logger is configured.
 - TCP/IP communication subsystem. Refer to Appendix B, “TCP/IP enablement” on page 327.
 - FTP is available.
 - HTTP server configured for basic functions.
- ▶ You need a USERID which is enabled to switch into the *superuser* mode.
- ▶ DB2 sysadmin authority is required. You can run the DB2 configuration jobs yourself or you can get a DB2 administrator to run them. The DB2 admin

USERID used must have a valid RACF OMVS segment. This USERID also needs the ability to run in *superuser* mode.

- ▶ You need to have the authority to perform the following tasks:
 - Creating the security rules, for which you need access to a RACF administrator (or the SPECIAL ATTRIBUTE assigned to your userid)
 - Updating PARMLIB, PROCLIB
 - Issuing console commands
 - Defining a log stream
 - Defining your CTRACE data set
 - Defining your WebSphere config HFS
 - Mounting HFSS, *superuser* authority required
- ▶ WLM policies need to be added for the WebSphere servers. You need access to the WLM administrative application.
- ▶ You will be adding data sets to LPA and LNKST. This can be done dynamically but eventually you will need to IPL. The load libraries are all PDSEs. They cannot be part of the LPA concatenation so the SETPROG command needs to be used to add them to LPA dynamically. PDSE LNKST data sets that will be processed during IPL must be cataloged in the Master Catalog.
- ▶ Recent maintenance to z/OS, TCP/IP, USS and so forth is recommended.

2.5.2 WebSphere configuration

Customization of WebSphere involves several tasks:

- ▶ Tailoring the z/OS environment
- ▶ Creating customized LDAP and WebSphere Application Servers
- ▶ Defining the initial runtime configuration for WebSphere
- ▶ Installing the System Management User Interface (SMUI) on your workstation

These tasks are explained in Chapter 3 of *WebSphere Application Server V4.01 for z/OS and OS/390 Installation and Customization*, GA22-7834.

The ISPF configuration dialog does a great job of leading you through the installation process. Here is some extra information to assist you when you use it:

- ▶ Make sure that you have all the information and authorities you need before you start.
- ▶ Save the installation information you enter into the dialog frequently.
- ▶ You can change the mount point to the WebSphere configuration directory to a name that reflects the standards at your site. We changed ours to

/WebSphere390. It is simpler than the default but close enough to the suggested default to be easily related back to the documentation.

- ▶ BBOERRLG - When you define your logstream, the LS_SIZE parameter needs to be coded if your dataclass does not specify the space information required.
- ▶ BBOLD2DB - This job needs to be run by a user ID with both DB2 sysadm and uid=0.
- ▶ If you want to keep the IVP scripts, write them to a permanent directory, not /tmp as suggested.
- ▶ When you select “Use OPERMCDS to control commands” as part of your security customization, check the generated RACF commands carefully. By default the generated rules have UACC(NONE) and may override existing profiles.
- ▶ When you get the following error message, daemon unable to bind to port 900 after starting the daemon, there is a chance that in a multiple IP stack environment the daemon does not know how to bind to the TCP/IP stack. You will need to code the environment variable SRVIPADDR to identify the TCP/IP stack you want the daemon to bind to. It is required for the Systems Management server and the Naming server. Update:
`config_dir/controlinfo/envfile/plex_name/SYSMGT01/current.env` with
`SRVIPADDR=your.ip.addr`.

In our sysplex, SANDBOX, with a WebSphere config HFS mounted at WebSphere390, we updated:

`/WebSphere390/controlinfo/envfile/SANDBOX/SYSMGT01/current.env`

Note: current.env is overwritten when you activate a conversation. From the Administration application you need to add SRVIPADDR=your.ip.addr as an environment variable to permanently define this value.

2.5.3 Installation Verification Program (IVP)

Three IVPs are supplied. You will need to set up two application server regions and an HTTP server. *WebSphere Application Server V4.01 for z/OS and OS/390 Installation and Customization*, GA22-7834 explains how this is done.

When you create the J2EE server instance, remember to specify the `db2sqljjdbc.properties` file you defined for this system.

Your final configuration should look like Figure 2-2 on page 21.

- After the daemon has initialized, it starts the other servers: BBOSMS, BBONM and BBOIR.
- 2. Application Servers, e.g.: /S BBOASR2.BBOASR2A
 - This starts the J2EE application server BBOASR2 procedure.
- 3. Web server (HTTPD), e.g.: /S BBWEBSRV

The following messages indicate a successful start. As each message appears, you can start the next component. We found that complete initialization could take longer and that applications would not run until all the TCP/IP connections were made. To check these we used the **netstat** command.

Table 2-1 Messages indicating a successful start

Subsystem job name	Job log Output	Look for this message(s)
BBODMN	JESMSG LG	BBOU0016I INITIALIZATION COMPLETE FOR DAEMON DAEMON01.
BBOSMS	JESMSG LG	BBOU0020I INITIALIZATION COMPLETE FOR CB SERIES CONTROL REGION SYSGMT01.
BBONM	JESMSG LG	BBOU0020I INITIALIZATION COMPLETE FOR CB SERIES CONTROL REGION NAMING01.
BBOIR	JESMSG LG	BBOU0020I INITIALIZATION COMPLETE FOR CB SERIES CONTROL REGION INTFRP01.
BBOASR2	JESMSG LG	BBOU0020I INITIALIZATION COMPLETE FOR CB SERIES CONTROL REGION BBOASR2A.
BBOASR2S	JESMSG LG	+Server "BBOASR2A" open for business. Note: This message has been moved into the warning messages category by PTF UQ61610 and will show up only if TRACEBUFLOC=SYSPRINT is specified.
WEBSRV	SYSPRINT	IMW0235I Server is ready.
	SYSOUT	IBM WebSphere Application Server for OS390 native plug-in initialization went OK.
BBOLDAP	SLAPDOUT	GLD0122I Slapd is ready for requests.

2.5.5 Checking the environment

Before you start up your WebSphere Application Server, you need to verify that all required subsystems are up and running. The key subsystems in the operation of WebSphere Application Server V4.0.1 for z/OS and OS/390 are:

► DB2

Make sure that DB2 is up and running.

From SDSF, issue **pre db2**, then **da**. You should see something similar to Example 2-1

Example 2-1 DB2 status in SDSF

SDSF DA SC63	SC63	PAG	0	SIO	0	CPU	3/	3	LINE	1-5 (5)	
COMMAND INPUT	====>									SCROLL	====> CSR
NP	JOBNAME	StepName	ProcStep	JobID	Owner	C	Pos	DP	Real	Paging	SIO
	DB2GMSTR	DB2GMSTR	IEFPROC	STC20786	STC	NS	FE	568	0.00	0.00	
	DB2GIRLM	DB2GIRLM		STC20787	STC	NS	FE	96	0.00	0.00	
	DB2GDBM1	DB2GDBM1	IEFPROC	STC20788	STC	NS	FE	7895	0.00	0.00	
	DB2GDIST	DB2GDIST	IEFPROC	STC20789	STC	NS	FE	269	0.00	0.00	
	DB2GSPAS	DB2GSPAS	IEFPROC	STC20790	STC	NS	FE	107	0.00	0.00	

Note that the DB2 Stored Procedure Address Space (SPAS) and the Distributed Data Facility (DIST) address spaces are not required.

► Resource Recovery Services (RRS)

Make sure that RRS is running.

► WorkLoad Manager (WLM)

To check whether your WLM application environment is active (so that your servers can be started at all), issue the command in SDSF:

```
/D WLM,APPLENV=*
```

You should see something similar to Example 2-2.

Example 2-2 Output of D WLM command

RESPONSE=SC63
IWM029I 12.00.07 WLM DISPLAY 314
APPLICATION ENVIRONMENT NAME STATE STATE DATA
BBJ2MQ AVAILABLE
BB0LTR AVAILABLE
BBPAAW AVAILABLE
BBPROD AVAILABLE
BBPWEB AVAILABLE
BBTEST AVAILABLE
BBTWEB AVAILABLE
CBINTFRP AVAILABLE
CBNAMING AVAILABLE
CBSYSGMT AVAILABLE
DB2XML AVAILABLE
WLMENV STOPPED
WLMENV1 STOPPED
WLMJAVA AVAILABLE

► Lightweight Directory Access Protocol (LDAP)

Check whether LDAP came up correctly in the SDSF log. Check for the messages shown in Example 2-3.

Example 2-3 LDAP startup messages

GLD0022I z/OS Version 1 Release 2 Security Server LDAP Server
Starting slapd.

12.26.32 STC01453 BPXM023I (CBLDAP)
GLD02062I The LDAP program will operate in single-server mode.

12.26.38 STC01453 BPXM023I (CBLDAP)
GLD0122I Slapd is ready for requests.

You can also use the **ldapsearch** command from a UNIX command line interface.

► WebSphere Application Server infrastructure

Now you can start WebSphere Application Server itself. The various address spaces that you defined during the installation and customization need to be available. Check if all WebSphere address spaces are up and running.

From SDSF, issue **pre bbo***, then **da**.

You should see something similar to Example 2-4.

Example 2-4 SDSF output for displaying the WebSphere address spaces

SDSF	DA	IBM2	SC63	PAG	0	SIO	935	CPU	23/	17	LINE	1-7	(7)
COMMAND	INPUT	====>										SCROLL	====> PAGE
NP	JOBNAME	StepName	ProcStep	JobID	Owner	C	Pos	DP	Real	Paging		SIO	
	BBODMN	DAEMON01		STC08957	CBDMNCR1	NS	FE	269	0.00	0.00			
	BBOLDAP	BBOLDAP	BBOLDAP	STC08955	CBLDAP	LO	FF	802	0.00	0.00			
	BBOSMS	SYSMGTO1	BBOSMS	STC08958	CBSYMCR1	NS	FE	4034	0.00	0.00			
	BBONM	NAMING01	BBONM	STC08959	CBNAMCR1	NS	FE	347	0.00	0.00			
	BBOIR	INTFRP01	BBOIR	STC08960	CBINTCR1	NS	FE	113	0.00	0.00			
	BBONMS	BBONMS	BBONMS	STC09084	CBNAMSR1	IN	C1	821	0.00	0.00			
	BBOSMSS	BBOSMSS	BBOSMSS	STC09078	CBSYMSR1	IN	C1	18T	0.00	0.00			

Also see 2.5.4, “Starting WebSphere Application Server” on page 21.

► TCP/IP

TCP/IP must be up and running. The MVS D TCPIP command gives you an overview of the TCP/IP stacks. In addition, from OMVS issue the **netstat** command and make sure your entries are similar to the ones shown in Example 2-5 on page 25 for the DAEMON01, INTFRP01, NAMING01 and SYSMGTO1 servers. The DAEMON and the SYSMGTO1 Control Region

should be listening on the ports that you specified during customization (by default port 5555 for the DAEMON, port 900 for the SYSMGT01 Control Region). The other ports are normally assigned dynamically. It is also very important that LDAP is connected correctly to TCP/IP.

Example 2-5 Output from the netstat command

# netstat						
MVS TCP/IP	onetstat	CS	V1R2	TCPIP Name: TCPIP		13:26:11
User Id	Conn	Local	Socket	Foreign	Socket	State
-----	----	-----	-----	-----	-----	-----
BBOLDAP	000003FE	0.0.0.0..1389		0.0.0.0..0		Listen
DAEMON01	0000042D	0.0.0.0..5555		0.0.0.0..0		Listen
INTFRP01	00000433	0.0.0.0..1030		0.0.0.0..0		Listen
INTFRP01	00000434	0.0.0.0..1031		0.0.0.0..0		Listen
NAMING01	00000431	0.0.0.0..1028		0.0.0.0..0		Listen
NAMING01	00000432	0.0.0.0..1029		0.0.0.0..0		Listen
SYSMGT01	00000430	0.0.0.0..1027		0.0.0.0..0		Listen
SYSMGT01	0000042F	0.0.0.0..900		0.0.0.0..0		Listen

Besides the “mandatory” components, you may have other subsystems involved, depending on the application, such as HTTP, MQSeries, CICS, or IMS.

2.5.6 Enabling Web application runtime and IHS plug-in

Configuring the WebSphere service plug-in for the IBM HTTP Server

The WebSphere plug-in is used to look up Web Applications in the WebSphere Application Server runtime. The plug-in itself runs within the IHS. It is, in fact, a little Application Server itself where you can run Web Applications.

An IBM WebSphere Application server node can only be bound to one IBM HTTP server (plug-in).

httpd.conf configuration

WebSphere provides HTTP server GWAPI plug-in routines to allow the WebSphere plug-in to be instantiated, and subsequently destroyed within an HTTP server address space at runtime.

In order to load the WebSphere service plug-in into an IHS address space, the entry point to the WebSphere plug-in initialization routine must be configured so that it is invoked as part of the HTTP server. The HFS root where WebSphere is installed must be specified; by default this is /usr/lpp/WebSphere.

The WebSphere plug-in is configured in an HFS configuration file, typically the was.conf. If you don't specify the location of a was.conf file on the ServerInit parameter in httpd.conf, then the plug-in uses the default located in /usr/lpp/WebSphere/WebServerplug-in/properties.

Attention: If you want to modify was.conf, copy the default to your own directory. The supplied default is overwritten when maintenance is applied to WebSphere.

Similarly, the entry point to the WebSphere plug-in's exit routine must be configured such that it is invoked as part of the HTTP server's termination process. The exit routine does not require any parameter.

For WebSphere 4.0.1, the application server initialization routine exists as entry point init_exit within the was400plugin.so DLL, and the exit routine exists as entry point term_exit, also within the was400plugin.so DLL. These routines are also referred to as service routines.

Example

The following example illustrates the directives that must be added to the httpd.conf configuration file, that the IHS will load with the WebSphere plug-in service routine.

In these examples for the WebSphere plug-in, WebSphere is installed in /usr/lpp/WebSphere. Note that the location and name of the adapter exits have changed from previous versions.

- The ServerInit statement identifies the initialization module for the WebSphere for z/OS plug-in:

```
ServerInit  
/usr/lpp/WebSphere/WebServerplugin/bin/was400plugin.so:init_exit  
/usr/lpp/WebSphere,/WebSphere390/was.conf
```

Attention: The ServerInit directive must be on one line in the HTTPD.CONF file. It is shown here split on three lines for printing purposes only.

- There is one space between init_exit and /usr/lpp/WebSphere.
- There is a comma and no spaces between /usr/lpp/WebSphere and /WebSphere390/was.conf.
- We chose to store our customized was.conf file in the WebSphere configuration HFS that we created for the config files generated by WebSphere. This is convenient because the HFS is already shared in the sysplex. You can store your was.conf file elsewhere if that is more appropriate in your environment.

- The `ServerTerm` statement identifies the termination module for the WebSphere for z/OS plug-in:

```
ServerTerm
/usr/lpp/WebSphere/WebServerplugin/bin/was400plugin.so:term_exit
```

- The `Service` directive for WebSphere 4.0.1 is required to identify the context root of your application. This assists the plug-in to determine where the applications should run (in the Web container or in the plug-in). In this case, the context root is `/webapp/examples`:

```
Service /webapp/examples/*
/usr/lpp/WebSphere/AppServer/bin/was400plugin.so:service_exit
```

Note: The `Service` directive must be on one line in the `HTTPD.CONF` file. It is shown here split on two lines for printing purposes only.

httpd.envvars

`httpd.envvars` defines the environment of the WebSphere for z/OS runtime. In the `httpd.envvars` file, define the following variables:

JAVA_HOME	Set the value to the exact location where the SDK V1.3 is installed on your system. Example: <code>JAVA_HOME=/usr/lpp/java/IBM/J1.3</code>
LIBPATH	Add the reference to the exact location where the WebSphere for z/OS libraries are installed on your system. Example: <code>/usr/lpp/WebSphere/lib</code>
NLSPATH	Append the WebSphere plug-in's message catalog directory, <code>applicationserver_root/AppServer/Msg/%L/%N</code> , to the existing <code>NLSPATH</code> statement. Example: <code>/usr/lib/nls/msg/%L/%N:/usr/lpp/internet/%L/%N:</code> <code>/usr/lpp/WebSphere/AppServer/Msg/%L/%N</code>
RESOLVE_IPNAME	<fully qualified IP host name of server on which WebSphere Application Server 4.0.1 SMS exists>. This will be the same as the <code>DAEMON_IPNAME</code> .
RESOLVE_PORT	900 (or port on which WebSphere Application Server 4.0.1 SMS server is listening, if not default)

Note: If your HTTP Server (and therefore the plug-in as well) is on the same system as your WebSphere Application Server 4.0.1 runtime, and you configured the SMS server to use the default port value of 900, you don't need these two values. But coding them is relatively easy, and it prevents confusion. So go ahead and code these even though strictly speaking they're not always necessary.

Configure the plug-in

Create directories

Create work and log directories for the plug-in, with write permission for the effective USERID or UID:

```
/var/WebSrv/was_logs  
/var/WebSrv/work
```

The WebSphere plug-in configuration file

The plug-in typically uses was.conf to determine if your request will be run in the plug-in or in a Web container. If you don't specify the location of a was.conf on the ServerInit parameter in httpd.conf, the plug-in uses the default located in `usr/lpp/WebSphere/WebServerplug-in/properties`. You should copy this file to your configuration directory so you can customize it.

Tip: If you migrate a WebSphere Application Server Standard Edition configuration file and leave the `DeployedWebApp` statements enabled, you have activated the so-called “configuration option”.

- ▶ If you are using an existing was.conf file that you previously used for WebSphere Application Server Standard Edition Version 3.5 or 3.02, make sure that you edit it, as follows:
 - Use the value 4.00 for the `appserver.version` statement.
 - Delete the `deployedwebapp` and `webapp` definitions for any Web applications that you want to run in the Web container, rather than in the WebSphere for z/OS plug-in itself.
- ▶ To create a new was.conf file, copy the default file shipped in `/usr/lpp/WebSphere/WebServerplug-in/properties/was.conf` to the `/WebSphere390` directory (or to any directory you choose—just make sure that you reference it from httpd.conf).

In both cases, you will then need to edit was.conf to update the `appserver.workingdirectory` and `appserver.logdirectory` statements to specify the work and log directories that you created in the previous step.

Configure the Web container

Perform the following steps to configure the J2EE server's Web container:

- ▶ Create a `webcontainer.conf` file by copying the default file shipped in the `/usr/lpp/WebSphere/bin` directory to the control file directory for the J2EE server:

```
cp /usr/lpp/WebSphere/bin/webcontainer.conf  
/WebSphere390/controlinfo/envfile/sysplex/BB0ASR2A/webcontainer.conf
```

- ▶ Edit the `webcontainer.conf` file and update the virtual host¹ and context roots definitions.

For a quick configuration, you can use the following definitions:

```
host.default_host.alias=host_name_of_HTTP_server ...  
host.default_host.contextroots=/
```

With this context roots definition, the single forward slash provides a universal catch-all for applications being bound to the default virtual host. Every application, regardless of the context root definition in the application.xml file, will run under the context and security model of this virtual host.

- ▶ Edit an existing or create a new Java virtual machine (JVM) properties file (`jvm.properties`) in the configuration directory for the J2EE server in which your Web applications will be installed. This properties file must specify the location of the Web container configuration file, usually named `webcontainer.conf`, using the following property:

```
com.ibm.ws390.wc.config.filename=/path/file
```

- ▶ Create a file called `trace.dat` in the control file directory. Insert the following statement into the file:

```
*=all=enabled
```

- ▶ Make sure that `webcontainer.conf`, `jvm.properties` and `trace.dat` have the same ownership and permission bits set as `current.env`.

Customizing the Web container in a J2EE server

The default `webcontainer.conf` is the runtime environment for Web Applications, particularly your initial environments. If you wish you can update the `webcontainer.conf` file to:

- ▶ Configure one or more virtual hosts within a Web container.
- ▶ Specify whether or not you want to collect session data.

¹ The virtual host model is the ability of the WebSphere Application Server to base the Web Applications configuration and security context on the DNS host name that is used within the request to invoke the application. Even if the (resolved) IP packets don't carry the host name, it is inside the HTTP payload. Mandatory for this support is that your Web Clients support HTTP 1.1 or higher (like most browsers do).

Refer to *WebSphere Application Server V4.01 for z/OS and OS/390 Assembling Java 2 Platform, Enterprise Edition (J2EE) Applications*, SA22-7836 Chapter 8 for details.

After editing the `webcontainer.conf` file, you must refresh the J2EE server to activate the changes you made.

The HTTP Transport Handler

WebSphere Application Server V4.0.1 for z/OS provides initial support for listening to HTTP requests. This support is called HTTP Transport Handler (Figure 2-1 on page 12). The HTTP Transport Handler exists in the control region of a J2EE Server. The control region routes the request to the Web container in the appropriate server region for processing. The HTTP Transport Handler is currently not prepared to talk directly to browsers. This means that it needs some kind of front end (e.g., Edge Server or HTTP Server) to handle all nuances of the browsers.

Configuring the HTTP Transport Handler

In order to activate the HTTP Transport Handler in a control region, you have to define the following environment variables in the J2EE environment file:

BBOC_HTTP_IDENTITY=USER_ID

Specifies a valid SAF user ID that will be used as the current security principal for this HTTP request. The user ID will be treated as an authenticated user by the Web container. If this variable is not specified, the request is executed under the identity of the server's control region.

Example:

```
BBOC_HTTP_IDENTITY=WEBSEC1
```

BBOC_HTTP_INPUT_TIMEOUT=n

Specifies the time in seconds that the J2EE server allows for the complete HTTP request to be received before cancelling the connection. The default value is 10 seconds.

Example:

```
BBOC_HTTP_INPUT_TIMEOUT=10
```

BBOC_HTTP_LISTEN_IP_ADDRESS=IP_ADDRESS

Specifies the IP address, in dotted decimal format, that WebSphere for z/OS J2EE servers use to listen for HTTP client connection requests. This IP address is used by the server to bind to TCP/IP. Normally, the server listens on all IP addresses configured to the local TCP/IP stack. However, if you want

to fence the work or allow multiple heterogeneous servers to listen on the same port, you can use `BBOC_HTTP_LISTEN_IP_ADDRESS`. The specified IP address becomes the only IP address over which this control region receives inbound HTTP requests.

Example:

```
BBOC_HTTP_LISTEN_IP_ADDRESS=9.117.43.16
```

BBOC_HTTP_OUTPUT_TIMEOUT=*n*

Specifies the time, in seconds, that the J2EE server waits from the time the complete HTTP request is received until output is available to be sent to the client. The default value is 120 seconds.

Example:

```
BBOC_HTTP_OUTPUT_TIMEOUT=120
```

BBOC_HTTP_PERSISTENT_SESSION_TIMEOUT =*n*

Specifies the time, in seconds, that the J2EE server waits between requests issued over a persistent connection from an HTTP client. After the server sends a response, it uses the persistent time-out to determine how long it should wait for a subsequent request before cancelling the persistent connection. The default value is 30 seconds.

Example:

```
BBOC_PERSISTENT_SESSION_TIMEOUT=10
```

BBOC_HTTP_PORT=*n*

Specifies the port at which the J2EE server listens for HTTP requests. Any requests received over the HTTP port are directed to the Web container for processing. If this variable is not specified, the J2EE server will not listen for HTTP requests directly. The use of this HTTP port does not preclude the use of the WebSphere for z/OS plug-in with this J2EE server instance. The Web container is capable of simultaneously processing requests received directly through the HTTP port as well as from the WebSphere for z/OS plug-in.

Note: Currently, HTTP requests received over this HTTP port cannot be authenticated using the mechanisms described in the J2EE Specification.

Example:

```
BBOC_HTTP_PORT=8080
```

BBOC_HTTP_SESSION_GC=*n*

Specifies an integer value indicating the maximum number of HTTP requests that will be processed over a single connection from an HTTP client. When the maximum number of requests have been processed, the client connection is closed. Set this value to 0 or 1 to turn off persistent connection processing. The default value is 50.

Example:

```
BBOC_HTTP_SESSION_GC=50
```

BBOC_HTTP_TRANSACTION_CLASS=*TRANSACTION_CLASS*

Specifies a valid WLM transaction class, which is used in the creation of the WLM enclave for all HTTP requests. If a valid WLM transaction class is not specified, no transaction class is set for the enclave. This would override the default WLM definition of the Application Server for the incoming HTTP handler requests.

Example:

```
BBOC_HTTP_TRANSACTION_CLASS=TCLASSA
```

Functions not supported by the HTTP Transport Handler

The HTTP Transport Handler currently does not support the following:

- ▶ The authentication policy specified in your Web application's deployment descriptor

As an alternative, your security administrator can define a surrogate user ID under which all requests received will execute.
- ▶ HTTPS

This limitation means that your installation cannot use SSL connections for inbound servlet requests.

These functions will be made available through the service stream. For more information, see *WebSphere Application Server V4.01 for z/OS and OS/390 Assembling Java 2 Platform, Enterprise Edition (J2EE) Applications*, SA22-7836.

Configuring a new J2EE server

This section describes how to configure a new J2EE Server. Refer to Chapter 8 of *WebSphere Application Server V4.01 for z/OS and OS/390 Assembling Java 2 Platform, Enterprise Edition (J2EE) Applications*, SA22-7836 for details.

1. Decide on naming conventions for J2EE application components, J2EE server elements, and z/OS or OS/390 subsystems, such as DB2. For recommendations for naming conventions, see the appendix in *WebSphere*

Application Server V4.01 for z/OS and OS/390: Operations and Administration, SA22-7835. We chose J2PROD for our server name and J2PROD1 for the instance name.

2. Define the WLM application environment, service class, and classification rules for the new J2EE server and the applications it will host. This is done by using the IWMARIN0 dialog, described in *z/OS MVS Planning: Workload Management*, SA22-7602.

Table 2-2 WLM definitions

WLM field	Value
Runtime server	Production
Application environment name	J2PROD
Start parameter	IWMSSNM=&IWMSSNM
Limit on starting server address space for a subsystem instance	No limit

3. Set up the database resources or connectors for data access.
 - Define DB2 subsystems.
 - Define DB2 databases.
 - Find out what DB2 subsystem name you need to specify when you define the J2EE server.
 - Create any database tables that your J2EE application components need to use.

4. Define security profiles and permissions.

For recommendations and instructions for setting up security for J2EE servers and J2EE application clients, see *WebSphere Application Server V4.01 for z/OS and OS/390 Installation and Customization*, GA22-7834.

If you want to install a J2EE application that requires role-based security, you need to define profiles in the EJBROLE class, and then allow users or groups to have read access to those profiles. See *WebSphere Application Server V4.01 for z/OS and OS/390 Installation and Customization*, GA22-7834.

Refer to Appendix G, “RACF REXX sample” on page 413 for a sample REXX program that defines all the security you need for a new application. It runs using TSO Batch.

5. Create JCL procedures for the control and server regions.
 - In your working PROCLIB data set, create a new member named J2PROD (the generic server name). Copy the BBOASR2 sample member from BBO.SBBOJCL into this new member, and make appropriate updates.

- Also in your PROCLIB, create a new member named J2PROD1 (the JCL procedure name you will later specify to WLM). Copy the BBOASR2S sample member from BBO.SBBOJCL into this new member, and make appropriate updates.
6. Enable J2EE server support for Web applications.
- If your application contains servlets or JSPs, complete these tasks:
- Set up the IBM HTTP Server and the WebSphere plug-in. Refer to “Configuring the WebSphere service plug-in for the IBM HTTP Server” on page 25.
 - Configure the Web container. Refer to “Configure the Web container” on page 29.
7. Specify the properties for the JVM.
- If you decide to change any of the default JVM settings, then you will need to create your own jvm.properties file and place it in the same HFS directory that contains the current.env file for your J2EE server.
8. Define the server configuration.
- Use the WebSphere Application Server Administration application to define a J2EE server, server instance (see Figure 2-3 on page 35), and data source, and install the Enterprise Archive (EAR) file. You need to complete the following tasks:
- Start the Administration application.
 - Add a new conversation (configuration dialog).
 - Add the server.
 - Add the server instance.
Specify a unique db2sqljjdbc.properties file.
 - Add the J2EE resource (Figure 2-4 on page 36).
 - Add the J2EE resource instance.

Conversations

- add new con factory
- Sysplexes
 - SANDBOX
 - J2EEServers
 - BBOASR2
 - Server Instances
 - BBOASR2A
 - BBOASR2B
 - J2EEApplications
 - JMSTest
 - RemoteWebContainer
 - WebSphereSampleEARFile
 - PolicyIVP
 - J2EEModules
 - policysession_deploy.jar
 - PolicyIVP_WebApp.jar
 - polycmp_deploy.jar
 - policybmp_deploy.jar
 - BBOLTR
 - BBPAAW
 - BBPROD
 - BBTEST
 - Servers
 - Systems
 - J2EE Resources
 - Logical Resource Mappings

BBOASR2A

Server instance name:
BBOASR2A

Server instance description:

System name:
SC63

Server name:
BBOASR2

Logstream name:

IIOP Firewall port:
0

SSL Firewall port:
0

Environment variable list:

	Level	Name	Value
1	SI	TRACEBUFFLOC	SYSPRINT
2	SI	DB2SQLJPROPERTIES	/WebSphere390/sc63
3	SRV	LIBPATH	/pp/db2/db2710/lib/
4	SRV	CLASSPATH	/pp/db2/db2710/c/
5	SRV	JVM_LOGFILE	/tmp/jvm.log
6	SRV	TRACEALL	1
7	SPX	BBOLANO	ENUS
8	SPX	CBCONFIG	/WebSphere390
9	SRV	DAEMON_PORT	EEEE

Figure 2-3 Define the server instance

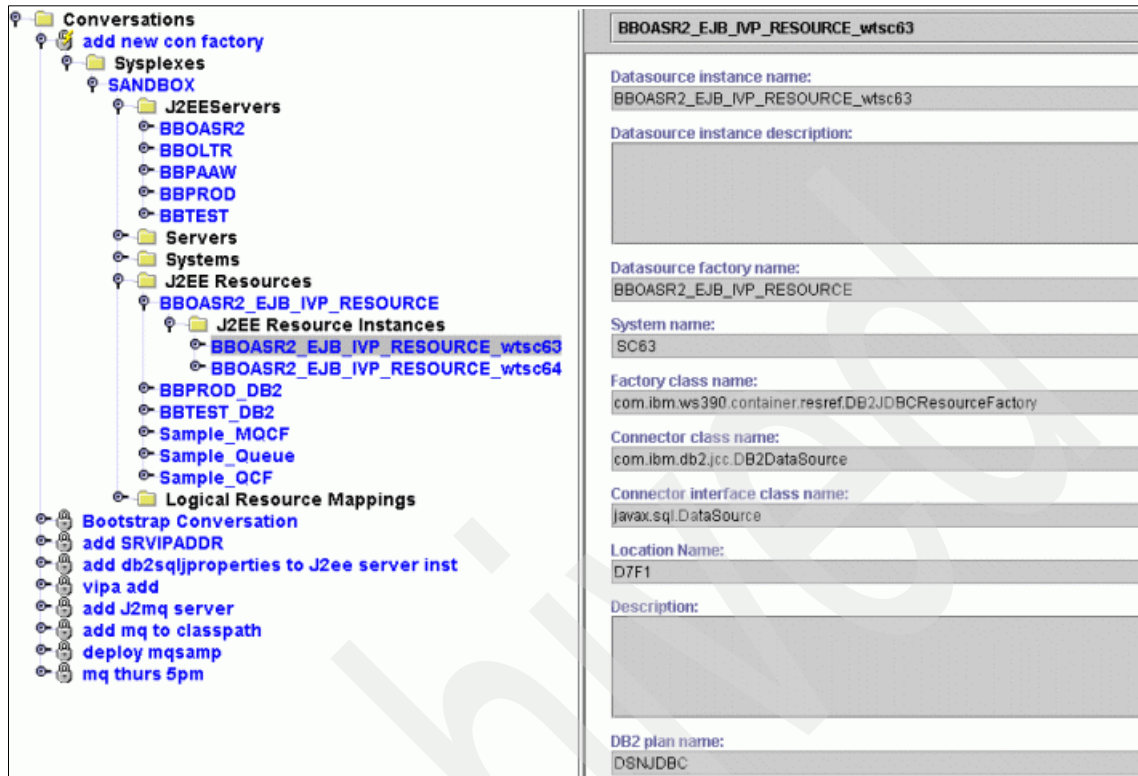


Figure 2-4 Define the J2EE resource

- Install the J2EE application.
- Validate the new conversation.
- Commit the conversation.
- Mark manual tasks as completed.
- Activate the new conversation.

2.6 Web Application runtime selection

Web Applications are non-EJB Web applications, composed from servlets, Java Server Pages (JSP) and static content (like HTML, GIF, JPEG). Web Applications are used to drive the EJB application. Requests for Web Applications are either run in the WebSphere plug-in (configuration option) or are passed to a J2EE server to be run in a Web container as shown in Figure 2-5 on page 37. How a request is processed is decided between the HTTP server, the WebSphere plug-in and the J2EE server.

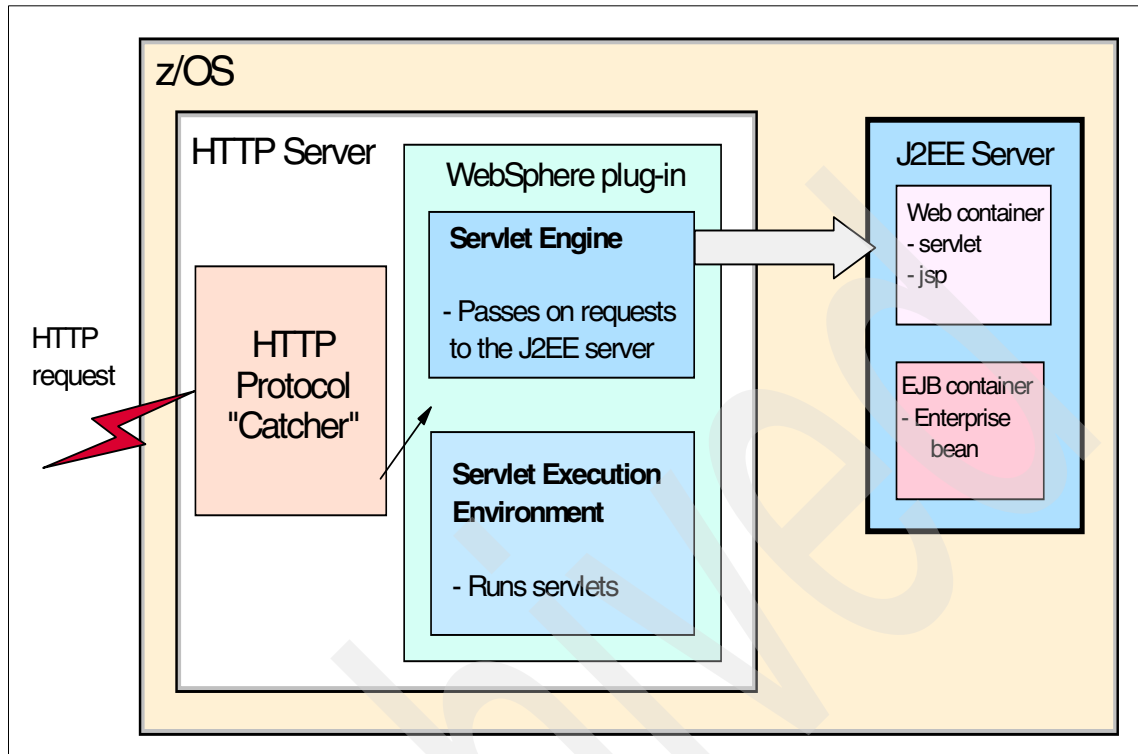


Figure 2-5 Plug-in functions

The IHS WebSphere plug-in interprets any requests passed to it by the HTTP server and either runs them within the plug-in itself or passes them on to a J2EE server Web container. The information about the application that is needed to process a request is held in the following files:

httpd.conf	Contains a service statement that starts the WebSphere plug-in if it is required.
httpd.envvars	Contains environment variables that allow the WebSphere for z/OS plug-in to communicate with the WebSphere for z/OS runtime.
was.conf	If the request is processed by the WebSphere plug-in, then this file contains Web application configuration and deployment information that the WebSphere plug-in uses for Web application processing. This file is not required if all your requests are intended to run in the Web container.
webcontainer.conf	If the request is executed inside the Web container, then this file contains definitions that enable the J2EE server's

Web container to isolate Web applications. It uses a *virtual host* to logically separate one or more Web applications from others installed in the same container. The *context root* is used to bind a request to a specific virtual host.

jvm.properties

Is associated with a specific J2EE server instance and contains a pointer to the webcontainer.conf file.

***.xml**

These files define the execution environment to the J2EE server. They are created from your deployment of the application.

Tracing a request

We will now investigate the processing of a specific request. The URL

```
http://wasplex2.itso.ibm.com/webapp/examples/simpleJSP
```

consists of these parts:

```
hostname:      wasplex2.itso.ibm.com
context root:  /webapp/examples/*
request(servlet): simpleJSP
default port:  80
```

Before the WebClient is able to send this request to the WebSphere Application Server, it needs to determine the server's IP address. To resolve, it sends a DNS lookup request for wasplex2.itso.ibm.com to the DNS server, which returns a —hopefully valid—IP address.

The HTTP server at wasplex2.itso.ibm.com listening on port 80 handles the inbound request. It then decides what to do with the request. It finds a matching Service statement in its httpd.conf configuration file.

```
Service /webapp/examples/*
/usr/lpp/WebSphere/AppServer/bin/was400plugin.so:service_exit
```

This Service statement contains the context root /webapp/examples/*, which matches part of the URL for the inbound request. The request is passed to the WebSphere plug-in whose location is specified by the Service statement.

The WebSphere for z/OS plug-in needs to determine the appropriate execution environment for the request: the Web container or the WebSphere plug-in.

The `ServerInit` statement defines how the plug-in is going to be initialized at IHS startup time. This is also the place where you tell the plug-in where it finds its configuration file (`was.conf`). The WebSphere plug-in scans the `was.conf` file at IHS startup time for application information and loads it into memory. If the WebSphere plug-in finds matching `DeployedWebAppl` statements, the WebSphere plug-in runs the servlet locally; that is, the Web application runs within the WebSphere plug-in.

The `httpd.envvars` file contains environment variables that allow the WebSphere plug-in to communicate with the WebSphere run-time.

If no matching information is found, the WebSphere plug-in assumes that the appropriate execution environment is the Web container. The WebSphere plug-in then uses the String Matcher Table (described in “The String Matcher Table” on page 40) to find the Web containers for the requested context root.

Example 2-6 extract of was.conf

```
deployedwebapp.examples.rooturi=/webapp/testapp
webapp.examples.servlet.simpleJSP.servletmapping=/myservlet
```

At this point, the WebSphere plug-in knows the inbound request is valid for this Web container, because the domain name and application context in the request match a defined virtual host and its associated context roots, respectively. The WebSphere plug-in passes the request to the Web container in the J2EE server.

The Web container uses the context root on the inbound request to find the servlet to run. It scans the XML files of applications (Example 2-7) installed in the J2EE server to find a matching context root value. It finds `/WebSphere390/apps/BBOASR2/WebSphereSampleEARFile/META-INF/application.xml`, which contains the descriptive meta information that ties together all of the WAR or EJB JAR files packaged in the EAR file.

Example 2-7 application.xml

```
<module>
  <web>
    <web-uri>examples.war</web-uri>
    <context-root>/webapp/examples</context-root>
  </web>
</module>
```

This context root matches the root specified on the inbound request. The Web container now has access to the `simpleJSP` servlet's code, and can dispatch the servlet for execution in the J2EE server Web container.

The String Matcher Table

A request received by the plug-in gets mapped to the WebSphere Application Server 4.0.1 runtime only when the plug-in matches the received URL to a known Web Application in the Web container. This matching is done in the WebSphere Application Server 4.0.1 plug-in's code and uses something called the String Matcher Table. This table is kept in memory, and is constructed using information the plug-in sees in the local was.conf file, as well as information it receives from the Web container about Web applications deployed there.

Here's an example of the table, taken from the ncf.log of the plug-in when appserver.loglevel=WARNING is set in the was.conf file.

String Matcher Table:

```
=====
/wg31.washington.ibm.com:8080/SimpleJSP/* --> (remote Web container JNDI home)
/localhost/webapp/examples/* --> LocalHostDispatch
/wg31.washington.ibm.com:8080/PolicyIVP/* --> (remote Web container JNDI home)
/localhost/ConfigViewer/* --> LocalHostDispatch
```

This table contains the same information you find in the Application Dispatching Information panel (see “Check plug-in Application dispatching information” on page 50).

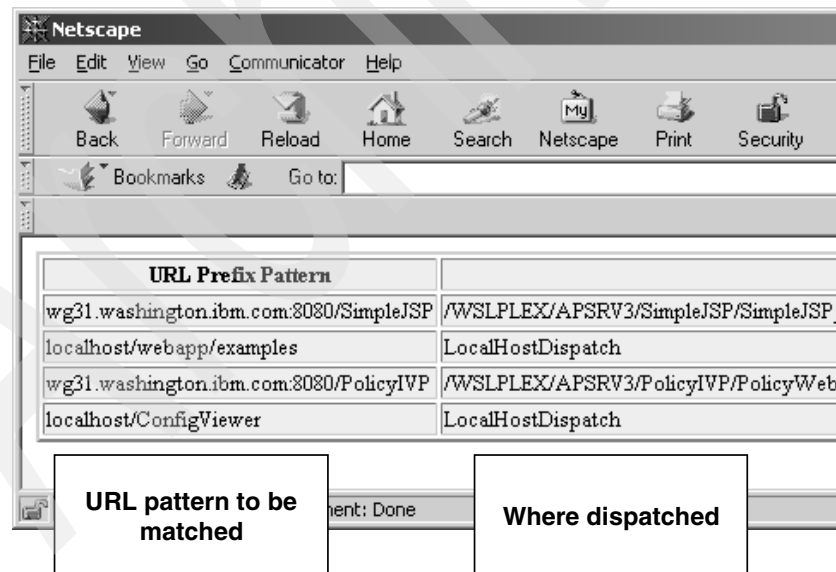


Figure 2-6 Application dispatching information

This is important because you don't need to crawl through the ncf trace to validate what Web applications the plug-in knows about—you can use the *application dispatching* information as well. You do have to look in the ncf trace to see that a request was properly mapped. That is discussed next.

For the sake of simplicity the process is often described as “The plug-in will look for a local definition in was.conf, and if not there then send it over to the Web container.” The truth is that the plug-in won't send the request unless it matches a known list of Web applications in the Web container. This list is the String Matcher Table. With that understanding, you may now proceed to validate that your request is getting mapped to the WebSphere Application Server 4.0.1 runtime's Web container. See “Check plug-in Application dispatching information” on page 50 for more information on the Web dialog.

2.7 System Management User Interface (SMUI)

The WebSphere for z/OS Administration and Operations applications are running on your Windows workstation and communicate with the System Management server. They are often called the “system management user interface” or “system management enhanced user interface” (SMEUI).

Installing SMEUI

The install file of the SMEUI (bboninst.exe) comes with WebSphere for z/OS. You can ftp the code from the /usr/lpp/WebSphere/bin/ directory. For instructions on how to install the SMEUI, see the section “Installing the Administration and Operations applications” in Chapter 3 of *WebSphere Application Server V4.01 for z/OS and OS/390 Installation and Customization*, GA22-7834.

Installation considerations

1. SMEUI uses TCP/IP to communicate between your workstation and the host. If your workstation does not use a domain name server, update your PC Hosts file with the host name of the server you are connecting to. For example:

```
201.3.10.10    wasplex2.itso.ibm.com
```
2. Make sure that the FTP server on z/OS or OS/390 is running.
3. After installation, before starting the SMEUI, initialize the WebSphere runtime server instances.
4. Define the BBONPARM workstation environment variable to pre-assign various login options for the SMEUI. These values may be overwritten when you start the Administration or Operations application.

Example:

Perform the following steps to set the default options for a bootstrap server called wasplex2.itso.ibm.com and a default user ID of CBADMIN when running under Windows NT:

- a. Open **Start->Settings->Control Panel->System**.
- b. Click the Environment tab.
- c. Define a new variable, BBONPARM.
- d. Assign the values -loginuser CBADMIN -bootstrapserver wasplex2.itso.ibm.com.
- e. Click **Set** to set the new variable.
- f. Close the dialog.

The default values will be primed in the login panel the next time the application is started.

For more information about SMUI, see *WebSphere Application Server V4.0.1 for z/OS and OS/390: System Management User Interface, SA22-7838*.

2.7.1 Administration application

The Administration application allows you to display and modify the Application Server configuration, that is, the WebSphere for z/OS applications and the environment in which they run.

It can be used system-wide (sysplex) or for a specific system.

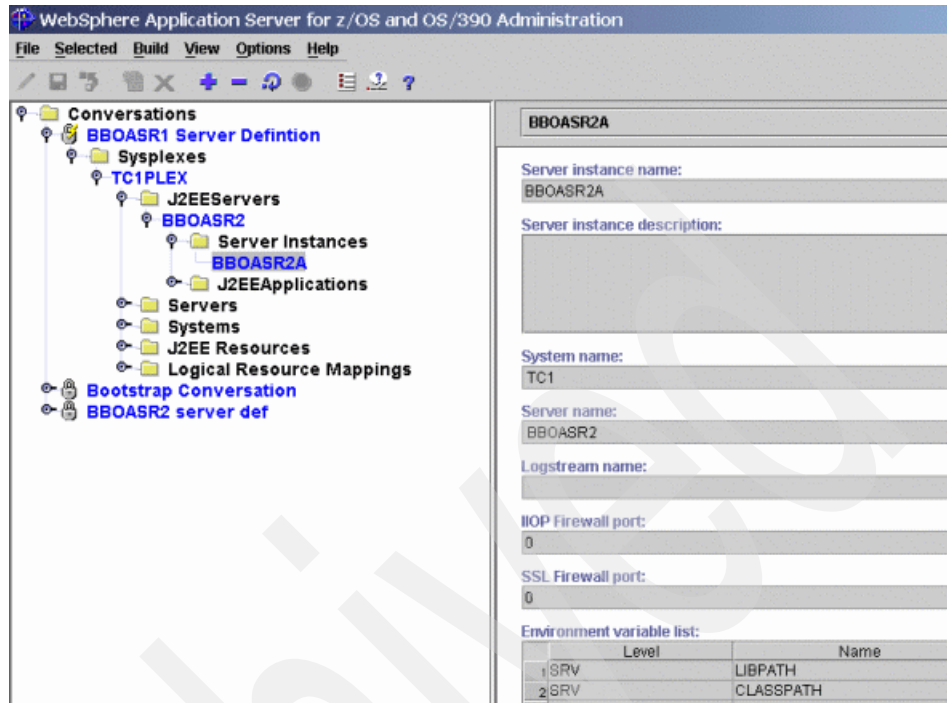


Figure 2-7 Example of the SMUI Administration window

Recommendations:

- ▶ Do not use the same administrator ID to log on to multiple concurrent sessions of the application, from either a single workstation or from more than one workstation. For example, if you start the Administration application on your workstation using CBADMIN as the user ID, you should not start another session using CBADMIN from either your own or a different workstation.
- ▶ If you define several administrator user IDs, they all may be logged on simultaneously, but only *one* should update and activate a conversation at a time. If more than one administrator attempts to activate a conversation, unexpected results will occur.

Important: We strongly recommend that you follow these recommendations, as there is currently no way to prevent multiple concurrent activations of conversations.

Server environment configuration file current.env

All of the environment variables for a server in WebSphere for z/OS and OS/390 are defined in a configuration file within the HFS called current.env. The location of this file is:

/WebSphere390/CB390/controlinfo/envfile/<Sysplex>/<servername>/current.env

Example:

/WebSphere390/CB390/controlinfo/envfile/SANDBOX/DAEMON01/current.env

Important: Normally, you would define the settings of environment variables via the SMEUI, but if you change these files by editing them directly, two things should be noted:

- ▶ Take great care when doing this because you might introduce new, very hard-to-find errors.
- ▶ Your changes will be overwritten with the settings in the SMUI when the next activation of the system is done.

2.7.2 Operations application

The Operations application manages operating tasks. It lets you manage the WebSphere servers and server instances. Each server and server instance in the currently active configuration is represented by an icon. You can display the status of server instances and the properties of each object. You can start and stop servers and server instances.

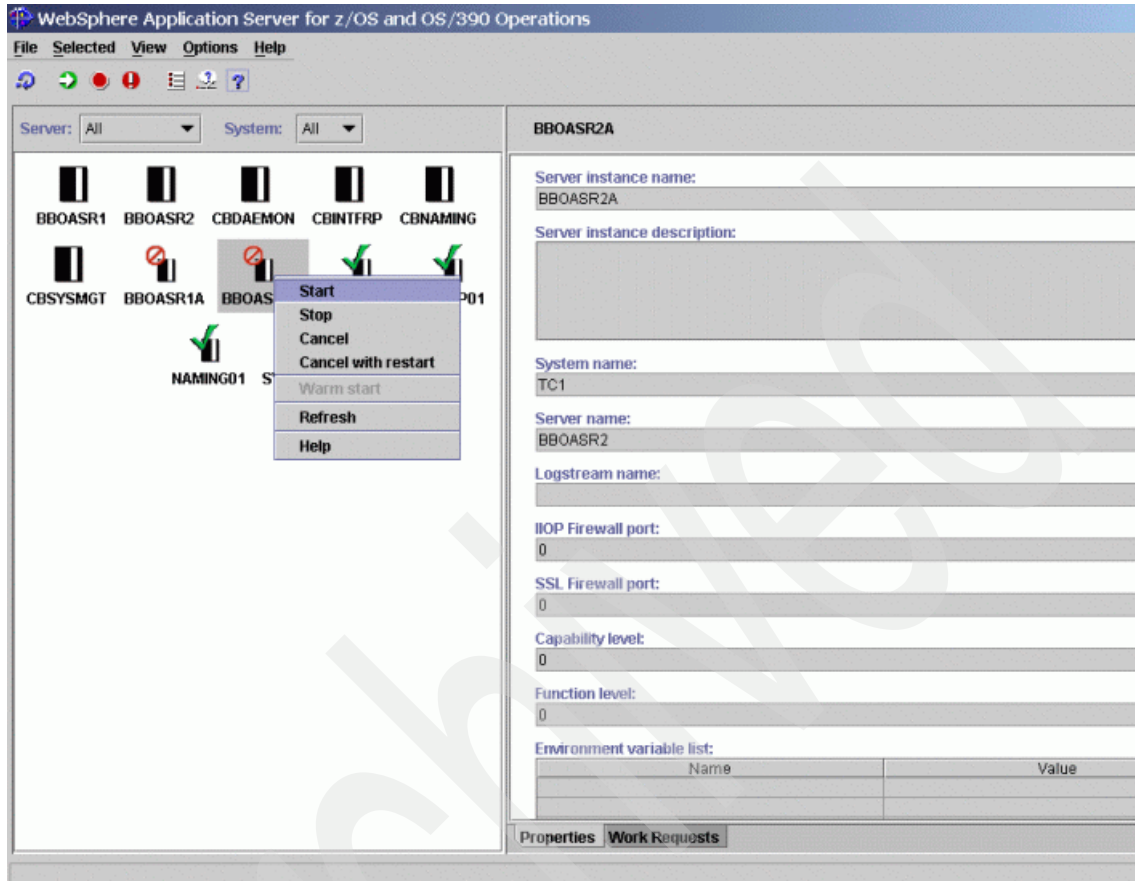


Figure 2-8 Example of the SMUI Operations window

2.8 System Management Scripting API (SMAPI)

The SM Scripting API (SMAPI) is an additional feature of WebSphere for z/OS Systems Management. It provides almost the same functions as the SMEUI. To use the SMAPI, you need to know how to write a REXX script, because REXX is the scripting language that is supported by the SM Scripting API.

A typical SMAPI script consists of (like any other procedural program) three parts:

- ▶ One part to generate the input file
- ▶ One part to call the function

- One part to process the result

There are predefined REXX functions available:

- CB390CMD(...) for calling an operations function
- CB390CFG(...) for calling an administrations function
- XMLGEN(...) for generating the input file
- XMLPARSE(...) for parsing the result
- XMLFIND(...) for finding a special attribute or value
- XMLEXTRACT(...) for extracting a known attribute

To use one of the administration functions, the default xml files must be present. Each default xml file lists all valid attributes for the corresponding administrative function. The style of the document is given by the Document Type Definition (DTD).

Installing SMAPI

The SMAPI is a fully-integrated part of WebSphere for z/OS, so it comes pre-installed. But you have to set up the client environment. Chapter 2 in *WebSphere Application Server V4.0.1 for z/OS and OS/390: System Management Scripting API*, SA22-7839, explains how to install the SMAPI.

Running SMAPI in Batch

Example 2-8 shows how to run an SMAPI script located in /u/cook1/SMAPI004, using BPXBATCH.

Example 2-8 Running BPXBATCH

```
//BPXBAT04 JOB (999,POK),'AMR',CLASS=A,REGION=OM,MSGCLASS=H,
// MSGLEVEL=(1,1),USER=CBADMIN,PASSWORD=
//*
//*****
/* Step1 - Run a SMAPI in batch
//*****
//STEP1 EXEC PGM=BPXBATCH,
// PARM='SH /u/cook1/SMAPI004'
//*-----
//STDIN DD DUMMY
//STDENV DD DUMMY
//STDOUT DD PATH='/tmp/&SYSUID..out',
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
// PATHMODE=SIRWXU
//STDERR DD PATH='/tmp/&SYSUID..err',
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
// PATHMODE=SIRWXU
```



```

//*****
//* Step2 - Send results to SYSOUT
//*****
//STEP2 EXEC PGM=IKJEFT01
//*-----
//SYSTSPRT DD SYSOUT=*
//HFSOUT DD PATH='/tmp/&SYSUID..out'
//HFSERR DD PATH='/tmp/&SYSUID..err'
//STDOUTL DD SYSOUT=*,DCB=(RECFM=VB,LRECL=133,BLKSIZE=137)
//STDERRL DD SYSOUT=*,DCB=(RECFM=VB,LRECL=133,BLKSIZE=137)
//SYSPRINT DD SYSOUT=*
//SYSTSIN DD *
OCOPY INDD(HFSOUT) OUTDD(STDOUTL)
OCOPY INDD(HFSERR) OUTDD(STDERRL)
/*

```

For samples of SMAPI REXX scripts, see *WebSphere Application Server V4.0.1 for z/OS and OS/390: System Management Scripting API*, SA22-7839 and *Migrating WebSphere Applications to z/OS*, SG24-6521.

2.9 WebSphere troubleshooting

This section has information to help with the troubleshooting of WebSphere Application Server V4.0.1 for z/OS and OS/390 problems.

2.9.1 WebSphere plug-in validation and basic debugging

These instructions apply to both the WebSphere Application Server 3.5 plug-in and the WebSphere Application Server 4.0.1 plug-in:

- ▶ Start the Web server.
- ▶ Browse the SYSOUT of the Web Server started task and find the following string:
:-)

Yes, that's a “smiley face”, and it's an indication that the plug-in initialized okay. The plug-in sometimes takes a few moments to initialize, and may not be up even though the Web server is operational. Give it a few moments and try again if you don't see it the first time.

If you still can't find the smiley face, search on the “frowny face” :-(. The Web server will write that message if something prevented the plug-in from initializing. Common causes for plug-in initialization failure:

- The JAVA_HOME variable in httpd.envvars is not set correctly.
 - A mistyped directory or in the ServerInit statement in httpd.conf. Check for case problems.
 - A second parameter on ServerInit statement points to a was.conf file and/or a directory that does not exist.
- Once you've verified the smiley face, issue the following URL from your browser:

`http://<host>/webapp/examples/index.html`

You should see the initial screen for “webapp/examples” that validates basic operation of plug-in code a screen.

If you receive this, it is an indication that your URL was successfully mapped to the WebSphere Application Server 4.0.1 plug-in using the Service statements in httpd.conf. You have successfully invoked the plug-in's function to serve the static page.

2.9.2 WebSphere Application Server validation and basic debugging

After the IVP process the most basic form of validation is to point your browser at the Web server and issue the URL used to invoke the application. But there are a few things you can do before that to insure success, and there's a methodology used afterwards when things don't work.

Preliminary validation

The following two “activities” help you determine if things are in proper working order prior to issuing a URL against the system.

Server region's SYSPRINT

Check the SYSPRINT of the application server region. It has two pieces of key information that will tell you whether things are working properly:

- An indication of which webcontainer.conf file is in use

If the server can't locate the webcontainer.conf you pointed to in the jvm.properties file, it'll take the default webcontainer.conf file located in the /usr/lpp/WebSphere/bin directory. If that happens, your virtual host won't be defined and things won't work. The server can fail to find your webcontainer.conf file with something as small as a typo in the long directory pointing to the private directory of the server.

- An indication of which applications are bound to which virtual hosts

If for whatever reason the application you deployed didn't get bound to a virtual host, your request to run the application will fail. Failure to bind to a virtual host can occur if the string you provide on the `host.<name>.contextroots=` property doesn't match the `<context-root>` setting in the deployment descriptor for the application. If you code the single slash, this problem isn't likely to occur. But an explicit coding of the `contextroots=` property might lead to a mismatch.

Do the following:

- Check the SYSPRINT of the server region. Near the top you'll find the following statement:

```
Web Container:Configuration File Name: <directory and file name of
config file>
```

The key is to make certain the directory and file name are what you specified in the `jvm.properties` file. A typo will result in the server using the `/usr/lpp/WebSphere/bin` directory.

- If the Web Container Configuration File Name doesn't point to your `webcontainer.conf`, then go back and check your `jvm.properties` file and make certain your pointer is correct.
- Now look further down in SYSPRINT and locate the following string:

```
VirtualHost Web Application Context Root Bindings:
/
```

Note: You'll find a separate block of information for each “context root binding” in the `webcontainer.conf` file. If you coded just the single forward slash, then you'll see only one such block of information. More context root bindings result in more blocks of information presented in the SYSPRINT.

- Now scroll down just a bit within the block of information for the context root binding (the single slash in this example) and look for the following:

```
VirtualHost Bound Web Applications:
  Web Application Context Root: /PolicyIVP
```

This indicates which applications matched the context root binding value. In this example, the application with a context root of `/PolicyIVP` has matched the single slash.

Now make sure that the application is bound to the virtual host you intended.

- Scroll down a bit further and look for the following:

VirtualHost Alias List:
wsc4.washington.ibm.com:8080

This indicates the virtual host, defined on the host.<name>alias= property in the webcontainer.conf file, to which the application has been bound. Make certain this is the virtual host you intended.

Check plug-in Application dispatching information

The WebSphere 4.0.1 plug-in provides a program that will tell you what applications it sees bound to virtual hosts in the WebSphere 4.0.1 runtime. This is a very handy way to verify that your plug-in sees things the way you intended them to be.

Do the following:

- ▶ With the Web server and application server up and running, point your browser to:

`http://<your host>/webapp/examples/index.html`

That will bring up a screen that looks like Figure 2-9 on page 51.

Note: Receiving the screen shown here is a good way to validate that the plug-in is working, but it does not in itself mean the Web container configuration information is correct.

- ▶ Click on the **Show server configuration** link and when the next screen comes up, scroll down and find the heading Application Dispatching Information. Click on that link. That will bring up a screen that looks like Figure 2-9 on page 51.

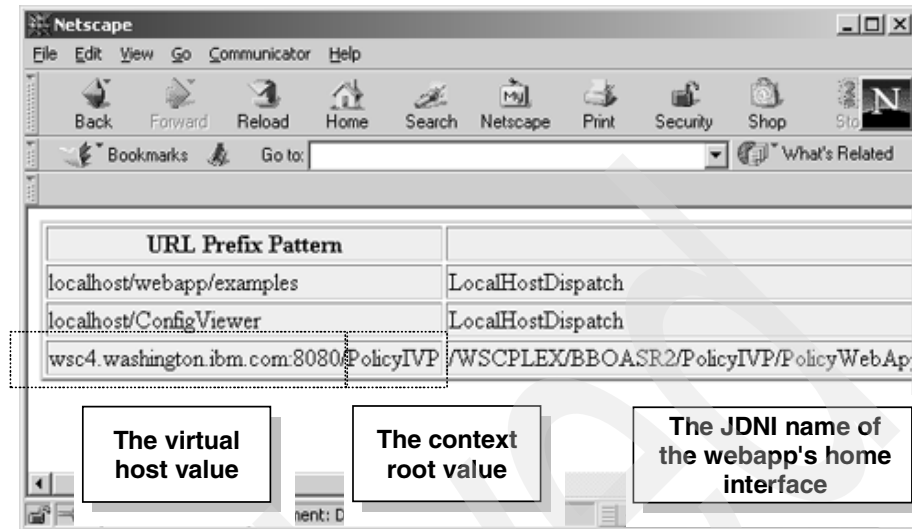


Figure 2-9 Application dispatching information screen

The appearance of your virtual host and your context root indicates the plug-in has knowledge of your deployed application over in the WebSphere 4.0.1 runtime environment. If you see only the two “localhost” values, things aren't working right.

Note: The information exchange between the plug-in and the WebSphere Application Server 4.0.1 runtime is not instantaneous. If you just started your server region, it may take a few moments for the information about what Web Applications are in the Web container to work its way to the plug-in. If you don't see your Web application immediately, give it a minute or so and refresh the screen. The asynchronous update interval can be controlled using the environment variable:

```
com.ibm.ws390.wc.serverCheckInterval=<interval in min>
```

The server region may take up to 10 minutes to start.

Basic debugging

When something goes wrong (and something always goes wrong), there's a way you can methodically walk through the system and determine what's failing (or at least determine what's not failing). Figure 2-10 on page 52 illustrates this methodology.

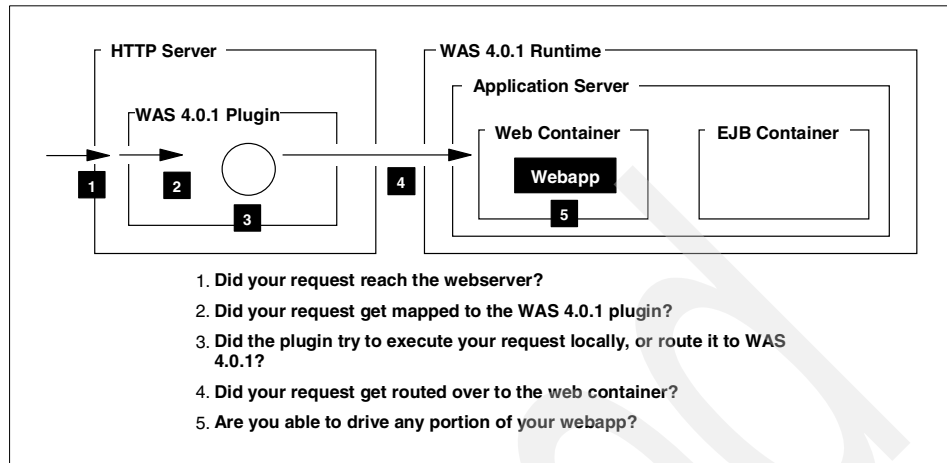


Figure 2-10 Basic debugging methodology

Validate that your request reached the Web server

Typing a URL at the browser and pressing Enter doesn't guarantee that the request will hit the server you think it will. The best way to validate that your request is getting to your server is to start the “-vv” trace and see if your request is recognized by the Web server.

For example, a URL of:

`http://<your_host>[:port]/webapp/examples/index.html`

will show up in the “-vv” trace as:

`Client sez.. GET /webapp/examples/index.html HTTP/1.0`

This validates that your request hit your Web server and is being acted upon.

Validate that your request was mapped to the plug-in

A request received by your Web server doesn't guarantee that the request will be mapped to the WebSphere Application Server 4.0.1 plug-in. That requires a properly coded Service statement. Here again, the “-vv” trace validates that the request is being mapped over to the plug-in.

Note: a properly coded Service statement is of little use if the plug-in itself isn't initialized. You can make certain the plug-in is up and going by reviewing the information found at “WebSphere plug-in validation and basic debugging” on page 47.

Again, a URL of:

```
http://<your_host>[:port]/webapp/examples/index.html
```

with a Service statement of:

```
Service /webapp/examples/* /usr/lpp/WebSphere/WebServerplug-in/...
```

will show the following in the “-vv” trace when the request is matched to the plug-in:

```
Service..... /webapp/examples/* matched "/webapp/examples/index.html" ->
:
Pattern..... match SUCCEEDED.
:
APIClassExec Calling function "service_exit"
```

This validates that your request was mapped to the plug-in. If you see your request mapping to a Pass statement, then the plug-in is not in control.

Validate that the plug-in isn't trying to run the Web application locally

The WebSphere Application Server 4.0.1 plug-in will attempt to run a request locally whenever it gets a match on a *DeployedWebApp* statement in the was.conf statement.

The best way to check is to review your was.conf file and make sure no `deployedwebapp rooturi` statement is defined that will map to your inbound request. If it matches, the plug-in will try to run the request locally. Remember: It is the absence of definitions in the was.conf file that causes it to try to map the request to the WebSphere Application Server 4.0.1 runtime.

The other way to validate this is by seeing whether the request is mapped to the WebSphere Application Server 4.0.1 runtime. That is explained next.

Validate that the request has been mapped to WebSphere Application Server 4.0.1 runtime

There are two ways you can validate that a request was mapped to the WebSphere Application Server 4.0.1 runtime:

- ▶ Perform a visual comparison of the URL against information in Application Dispatching Information (this proves nothing, but it'll eliminate obvious errors of typing and such).
- ▶ Enable tracing of the plug-in and interrogate the trace file for evidence of the request getting mapped.

This is covered here. Do the following:

- ▶ Edit the was.conf file and set the following properties:

```
appserver.tracelevel=com.ibm.*=all=enabled
appserver.loglevel=WARNING
appserver.logdirectory=(directory to which logging will occur)
```

Note: Setting the appserver.tracelevel to all=enabled like this produces a tremendous amount of output. You would never have this running on a production system because it would drain away too much system resource. If you set this property on your test system, remember that it is set and turn it off (comment out the line and restart the Web server) after you have done your debugging. Otherwise, you will likely quickly fill the HFS in which the logging is done.

- ▶ Stop and restart the Web server.
- ▶ Verify that the plug-in initialized (see 2.9.1, “WebSphere plug-in validation and basic debugging” on page 47).
- ▶ Clear your browser cache.
- ▶ Issue the URL that produces the failure indication.
- ▶ Browse the ncf log, which should be quite large with tracing enabled.
- ▶ Issue a **find** command on the URL you issued. For example, if your entire URL was:

```
wg31.washington.ibm.com:8080/SimpleJSP/simple.jsp
```

then do a find on only the /SimpleJSP/simple.jsp portion of it.

- ▶ When you find the first occurrence of the string, scroll down just a bit. You should see the String Matcher Table as well as the Basic Rules and Exact Rules tables. If the plug-in sees a match on the URI vs. the String Matcher Table, you should see the following immediately afterwards in the trace:

```
WS390Redirect < localDispatch
WS390Redirect D Matched JNDI Name : "/WSLPLEX/APSRV3/SimpleJSP/...
InProcNativeS D ConnectionStub.getRequestURI: instance = 2
WS390Redirect D Remotely Dispatching Request URI "/SimpleJSP/simple.jsp"
```

What this means is that the request (/SimpleJSP/simple.jsp in this example) matched an entry in the String Matcher Table, the plug-in was able to determine the JNDI name from the table, and the plug-in is dispatching the request remotely.

After this there is a great deal of trace activity showing the plug-in flowing requests to invoke the home interface of the remote Web container. All that will augment the validation provided by the Remotely Dispatching Request URI message. Your request has been sent to the Web container.

The servletmapping string and execution of Web application class files

The activity just discussed simply validates that your request got to the Web container. But that does not guarantee that it will execute a Web application. That's because the specific Web application to be executed is implied with the servletmapping string, which is part of the URL to be sure, but not something the plug-in worries about. So a request you send could have the correct context root, which will allow the plug-in to map the request to the runtime, but have an incorrect servletmapping, which will result in an error.

The servletmapping string is defined in the deployment descriptor for the Web application (the web.xml file in the WAR file), and has an XML tag of `<url-pattern>`, see Figure 2-11.

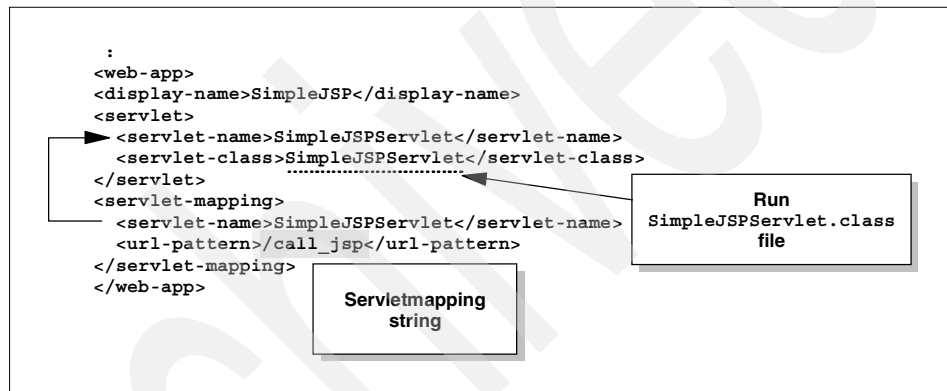


Figure 2-11 Servletmapping string in the Web application deployment descriptor

The interesting thing is this: If the Web container doesn't see a match with a defined `<url-pattern>` servletmapping string, it'll then determine whether the request is for a JSP. It does this by checking the end of the URI for a string of `.jsp`. If the URI doesn't end in `.jsp`, the Web container assumes the request is for a simple file and goes looking for a file to send out.

Key: If you accidentally mistype the servletmapping value on the URL, but the context root value is proper, the request flows over to the Web container. But the Web container won't find any Web applications with a servletmapping equal to your mistyped URL value, so it'll eventually consider it a request for a simple file. The error you get on the browser screen will tell you which file type the Web container was trying to service. These are described next.

Key error indicators found on the browser screen

If your URL has the proper context root value and the request makes it over to the Web container, there are plenty of opportunities for errors. The first step is to look at the error message on the browser to narrow the possibilities. Consider the following examples:

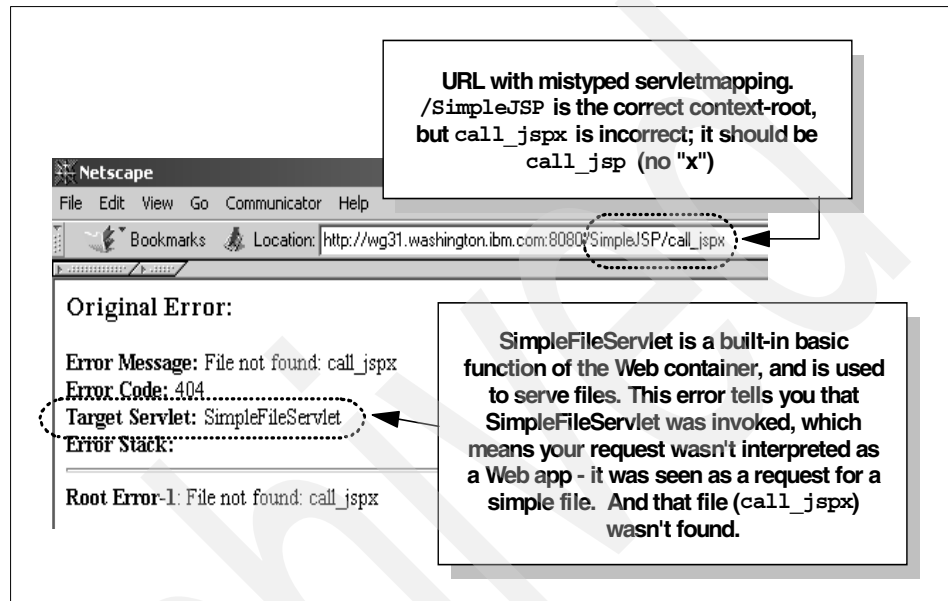


Figure 2-12 URL with mistyped servletmapping string

In Figure 2-12, the person issuing the URL mistyped the servletmapping value by adding an extraneous "x" at the end. The plug-in properly interpreted the correctly typed context-root value of SimpleJSP and dutifully passed the request over to the Web container. But the Web container looked through its set of known servletmapping values and didn't find a call_jsp servletmapping string, so it assumed the request was for a simple file. Having failed to find call_jsp in the root of the WAR file in the HFS, it gave up and issued the File not found error message.

Now consider an example where a JSP is requested directly, but the JSP name is mistyped. The .jsp extension is correct, but the first part of the name is not:

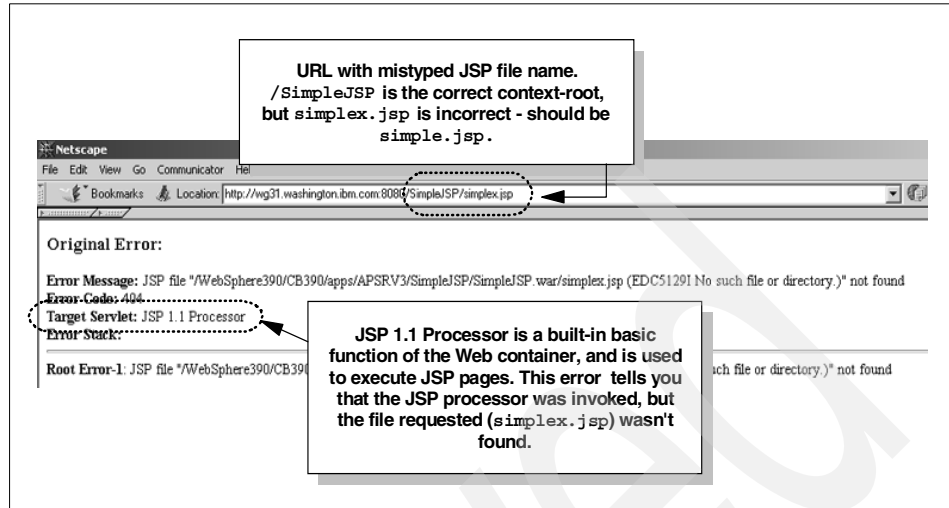


Figure 2-13 URL with mistyped JSP file name

So the bottom line is this: If what you're trying to do is invoke a servlet and you see either the SimpleFileServlet error message or JSP 1.1 Processor error message, you know you've probably got something wrong with your servletmapping string.

Determine if you can serve any portion of your Web application

Web applications consist of not just servlet class files, but static files such as HTML and JPG/GIF files, and JSP pages. You should be able to request and be served those files directly. To insure your path to the Web application in the Web container is open, you could, for example, request an image file with the following URL:

`http://<your host>/SimpleJSP/banner.gif`

If you can get the GIF, and you're certain that GIF is not being served by the HTTP server directly (validate this in "vv" trace), then you know the following things have worked: Service in `httpd.conf`; String Matcher Table function in WebSphere Application Server 4.0.1 plug-in; receipt of request by Web container in WebSphere Application Server 4.0.1; recognition of request as simple file; and the locating and serving of the file itself.

Validate request results in execution of desired Web application

So how can you validate that your Web application was in fact run in the Web container? By routing the `TRACEALL=1` output to `SYSPRINT` and looking at the results. Do the following:

- ▶ Edit the current.env file of your application server instance and make sure the TRACEALL property is set to at least 1.
- ▶ Set TRACEBUFFLOC=SYSPRINT.
- ▶ Stop and restart your application server.
- ▶ Clear your browser cache.
- ▶ Issue the URL that you wish to test.
- ▶ Review the contents of the application server instance's SYSPRINT. What you will find is something that looks like this:

```
Trace: 2001/09/12 18:09:47.991 01 t=8E15C0 c=1.27 key=P8 (13007002)
  FunctionName: com.ibm.servlet.engine.webapp.ServletInstance
  SourceId: com.ibm.servlet.engine.webapp.ServletInstance
  Category: AUDIT
  ExtendedMessage: Loading.servlet:."SimpleJSPServlet"
Trace: 2001/09/12 18:09:48.066 01 t=8E15C0 c=1.27 key=P8 (13007002)
  FunctionName: com.ibm.servlet.engine.srt.WebGroup
  SourceId: com.ibm.servlet.engine.srt.WebGroup
  Category: AUDIT
  ExtendedMessage: [Servlet.LOG]:."SimpleJSPServlet: init"
Trace: 2001/09/12 18:09:48.090 01 t=8E15C0 c=1.27 key=P8 (13007002)
  FunctionName: com.ibm.servlet.engine.webapp.ServletInstance
  SourceId: com.ibm.servlet.engine.webapp.ServletInstance
  Category: AUDIT
  ExtendedMessage: Servlet.available.for.service:."SimpleJSPServlet"
```

Figure 2-14 Contents of SYSPRINT showing loading and making ready of a servlet in the Web container

With this indication appearing in the application server instance's SYSPRINT, you know the request has been received by the Web container, has been recognized, and the servlet class file is being loaded.

2.9.3 Common configuration errors and the symptoms displayed

This section shows some error symptoms and discusses the common configuration errors that cause them.

The first error symptom you see will be on the screen of your browser. Unfortunately, these symptoms are almost never enough to pinpoint the problem. So further digging into the various traces is necessary.

Note: The browser error symptoms shown in this document are based on what Netscape displays. Internet Explorer may sometimes display different things, particularly if you have the Show Friendly Error Messages option turned on.

Browser error messages

- ▶ Error 404 - File was not found.
See “No Service directive coded that matches the URL received” on page 61.
- ▶ Error 500 - Service handler performed no function. See the following:
 - “Plug-in not initialized” on page 61
 - “Service directive has error in directory or file name of plug-in code” on page 63
 - “Service directive has error in the exit routine named on the directive” on page 63
- ▶ Error 500 - Failed to load target servlet.
See “Plug-in tries to run the code locally” on page 69.
- ▶ Virtual Host or Web Application Not Found. See the following:
 - “WebSphere Application Server 4.0.1 not started” on page 65
 - “Web container not configured in WebSphere Application Server 4.0.1 application server” on page 67
 - “URL doesn't contain a value that matches the defined context root or virtual host” on page 71
 - “Your application didn't bind to a virtual host” on page 73
 - “Plug-in not connected to the WebSphere Application Server 4.0.1 runtime you think it is” on page 74
- ▶ Recursive Error Detected - File Not Found. See the following:
 - “Servlet's mapping string doesn't match” on page 76
 - “Mismatch in servlet name in deployment descriptor” on page 78
 - “Class file is incorrect” on page 79

Errors related to request not reaching plug-in

A request received by the HTTP Server is passed to the WebSphere Application Server 4.0.1 plug-in with the Service directive in the httpd.conf file. There are quite a few reasons why that request might not make it “over the wall” into the plug-in, such as:

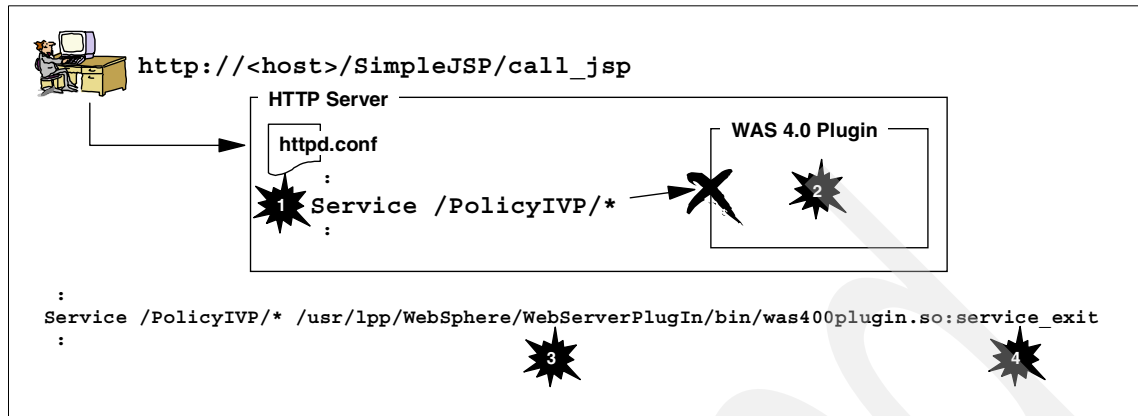


Figure 2-15 Some reasons why a request may not reach the plug-in

1. No Service directive coded that matches URL received.

For a request to make it to the plug-in, a Service statement needs to be coded with a URL pattern that matches the request. It is very easy to overlook adding a new Service statement when adding a new Web application. And if no Service statement matches, the request falls through and probably matches on some Pass directive later, and the plug-in is never invoked.

2. Plug-in not initialized.

You could have a perfectly coded Service statement, but if the plug-in itself isn't initialized, then the Service statement has nowhere to send the request. The plug-in may fail to initialize for several reasons. You should always check to make sure that the plug-in has initialized before testing any new Web applications.

3. Service directive has error in directory or file name of plug-in code.

The Service directive has a rather lengthy portion where the directory and file name of the plug-in code are specified. If you mistype any portion of that, the Web server will try to invoke the plug-in, but will fail because no such directory or file exists. This error will not be caught at Web server startup; it only becomes evident when a request matches and IHS invokes the associated Service Routine.

4. Service directive has error in the "exit" routine named on the directive.

The plug-in has three different "exit routines" and the one invoked on the Service statement is the service_exit. That is, unless you mistype that value. Then problems occur.

Each of these is discussed next, with the error symptom associated with the error.

No Service directive coded that matches the URL received

The request fails to map to a coded Service statement in the httpd.conf file. The Web server continues to evaluate the request against other directives, and eventually the request maps to a Pass statement (probably the Pass /* statement), or fails to map at all.

Browser error symptom



Figure 2-16 Error 404 message

Log or trace symptom

This problem will show itself in the Web server's "vv" trace. There will be no matches on Service directives, and you will probably see it match against the Pass /* directive, but the file implied won't be found.

How to correct

Make sure the URL being sent from the browser maps against one of your defined Service statements. If necessary, code another Service statement and restart the Web server.

Plug-in not initialized

A Service statement matched, but the plug-in to which the request is intended is not initialized. The request has nowhere to go.

A plug-in not initializing can be due to errors in the was.conf file, an improperly coded ServerInit statement, or a missing JAVA_HOME variable in the httpd.envvars file. You check for initializing by searching for the smiley face in the SYSOUT of the Web server's started task (see 2.9.1, "WebSphere plug-in validation and basic debugging" on page 47).

Browser error symptom

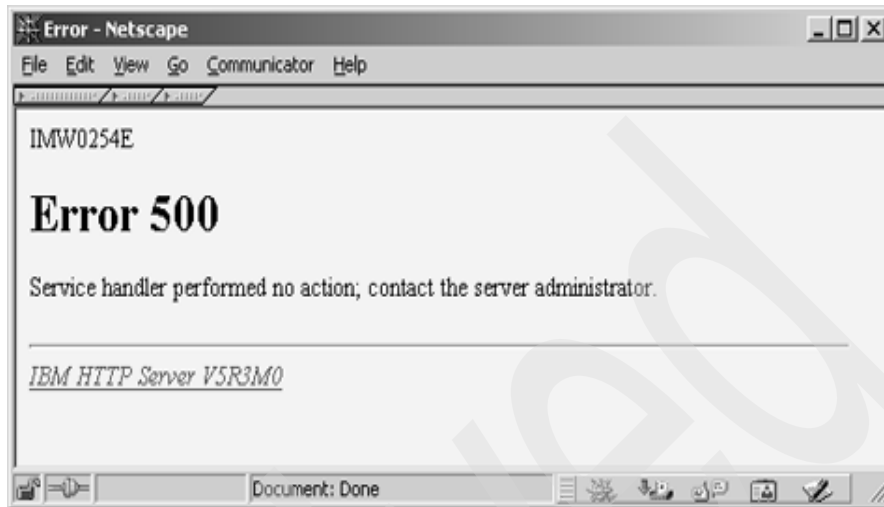


Figure 2-17 Error 500 message

The key here is the phrase “Service handler performed no action.” That means the Service handler was invoked, but did nothing. So a Service statement was matched.

Log or trace symptom

This problem shows itself in the Web server’s “vv” trace, but it is very obscure. There will be a match on a Service directive and then you’ll see the Web server trying to match on other Service directives. But no indication of why the first match wasn’t honored is given. Ultimately you see the ERROR 500 message in the trace.

How to correct

Check the SYSOUT of the Web server’s started task and look for the smiley face or the sad face [:-(]. You will likely see the sad face or neither. Some initialization failures will be cited in the “V.” trace (for example, a was.conf file not found will be flagged there). Other causes for plug-in initialization failures can be found in the plug-in’s “native” log.

Service directive has error in directory or file name of plug-in code

In this scenario the plug-in is initialized, and a Service directive is coded to match the URL. But the directory or file name of the WebSphere Application Server 4.0 plug-in code has some error in it that causes the Web server to fail to find the plug-in. If it can't find the plug-in, it can't invoke the plug-in with your request.

Browser error symptom

See Figure 2-17 on page 62. This is the same error symptom displayed when the plug-in isn't initialized.

Log or trace symptom

Unfortunately, the “vv” trace for this problem displays the same information displayed when the plug-in wasn't initialized. You see a match on the Service statement, then without explanation the Web server starts trying to match the other Service statements. It finally ends with a 500 error.

How to correct

First, verify that the plug-in initialized. If it has, then visually inspect the Service statement for typos. Some common problems:

- ▶ Lowercase “s” in Websphere rather than WebSphere
- ▶ Lowercase “i” in WebServerplugin rather than WebServerplugIn

Fix and restart the Web server.

Service directive has error in the exit routine named on the directive

The Service directive has an exit routine of `service_exit`, which points to the portion of the plug-in code to be invoked when a request is received. If you made a mistake in coding that, the plug-in will fail to initialize. (A common mistake is replicating the `ServerInit` statement to make a Service statement, and forgetting to change the exit routine from `init_exit` to `service_exit`.)

The external symptom is similar to other things that cause the plug-in to initialize, but the indication in the “vv” trace is different.

Browser error symptom

See Figure 2-17 on page 62.

Log or trace symptom

In the “vv” trace you will find a string that identifies the problem:

Failed to load function <exit routine>:
EDC5214I Requested function not found in this DLL

You will find no smiley face and no sad face. This will be the only indication of this problem.

How to correct

Inspect the exit routines specified after the colon on the Service statement. Each one should have: service_exit. Any variations on this cause a failure.

Errors related to plug-in not passing request to Web container

Once the request has been passed to the plug-in, the plug-in has to pass it to the WebSphere Application Server 4.0.1 runtime. There are several things that can keep that from happening:

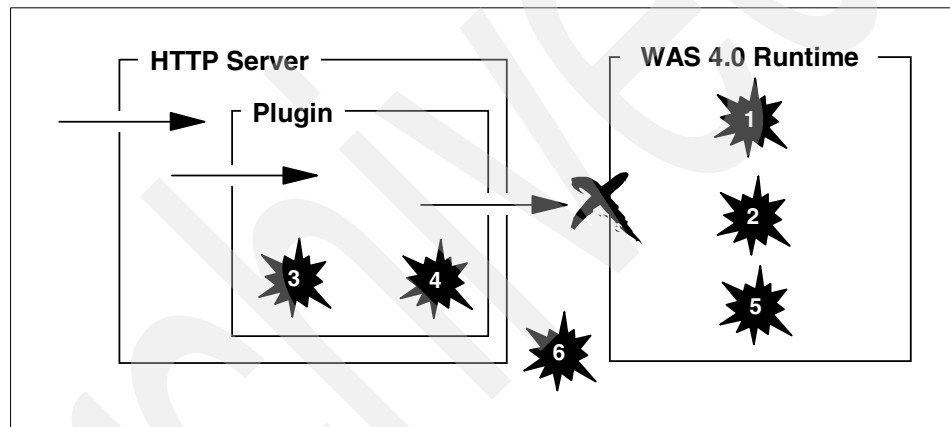


Figure 2-18 Some reasons why requests may not be sent to runtime environment

1. WebSphere Application Server 4.0.1 application server not started.

Your Web application might be perfectly configured and deployed into the application server, but if that server isn't started then the plug-in won't have much success routing the request.

2. WebSphere Application Server 4.0.1 Web container not configured.

The process of configuring the Web container is a manual one, and it's easy to forget to do if you are in test mode and creating many different application servers. With no Web container configured, the deployed Web application will be recognized (sort of) but it won't be bound to any virtual host. Therefore, your attempts to access it will result in failure.

3. The plug-in tried to run the Web application locally.

If in the past you ran the Web application in the plug-in, but are in the process of migrating your Web application to the WebSphere Application Server 4.0.1 Web container, you might forget to remove the definitions from the local was.conf file. That means the plug-in will try to run the Web application locally, probably with no success.

4. URL doesn't map to any defined context root or virtual host.

Your URL might match a Service statement and get passed to the plug-in properly, but if the URL as received doesn't match any *context roots* found in the String Matcher Table (see “Validate that the request has been mapped to WebSphere Application Server 4.0.1 runtime” on page 53 for an explanation of what that table is), then the request won't have any place to go.

5. Application doesn't bind to a virtual host.

If the context root setting for your Web application isn't able to bind to a virtual host defined in the Web container, your application will be unrecognized. If you're using a contextroots= statement of a single slash (/), your Web application will always bind. But if you're using a contextroots= value of something more specific, it might not bind.

6. The plug-in isn't connected to the WebSphere Application Server 4.0.1 system you think it is.

The plug-in connects to the Systems Management Server (SMS) based on the RESOLVE_IPNAME and RESOLVE_PORT environment variables in the httpd.envvars file for the Web server in which the plug-in runs. If you have multiple WebSphere Application Server 4.0.1 runtime environments, it's possible that your plug-in isn't connected to the runtime you think it is. If that's the case, your Web application might not be recognized by the plug-in.

WebSphere Application Server 4.0.1 not started

The WebSphere Application Server 4.0.1 plug-in works in conjunction with the WebSphere Application Server 4.0.1 application server. If that server isn't started, then the plug-in has nowhere to redirect the request.

Browser symptom



Figure 2-19 Application server not started

Log or trace symptom

You'll see the following error on the console:

```
+BBOU0516E LOCATE REQUEST FAILED FOR SERVER - (server name)
```

The plug-in found its way to the SMS server (via the `RESOLVE_IPNAME` and `RESOLVE_PORT` variables in `httpd.envvars`) and was given the names of the application servers that deployed Web applications. Then when the plug-in went to communicate with the application server, it was unable to locate the server. If the server was not started, then the locate will of course fail.

Any requests for applications deployed in the not-yet-started application server will fail with the Virtual Host or Web Application Not Found message.

If you then look in the plug-in's `ncf` log, you'll see an indication of the error:

```
ServletHost    W Web.Group.Not.Found:."/SimpleJSP/simple.jsp"  
ServletReques X Web Group Not Found  
The web group /SimpleJSP/simple.jsp has not been defined
```

How to correct

Start the application server, provide enough time for the plug-in to communicate with the server, and issue the URL again.

Web container not configured in WebSphere Application Server 4.0.1 application server

It is possible to deploy a Web application into a WebSphere Application Server 4.0.1 application server even though the Web container has not been configured. WebSphere Application Server will place the Web application code and files into the HFS and will even recognize that a Web application has been deployed. WebSphere Application Server will then make use of the default webcontainer.conf file, but that default copy has no virtual host defined. Failing to bind to a virtual host means the plug-in will see the Web application as a “localhost” dispatch, but will fail to load it because the executable code isn't available to the plug-in.

Configuring the Web container is covered in “Configure the Web container” on page 29.

Browser symptom

See Figure 2-20 on page 68.

Log or trace symptom

A peek in the ncf log will reveal the following:

```
ServletHost    W Web.Group.Not.Found:."/SimpleJSP/simple.jsp"  
ServletReques X Web Group Not Found  
The web group /SimpleJSP/simple.jsp has not been defined
```

Further, the Application Dispatching panel will show:

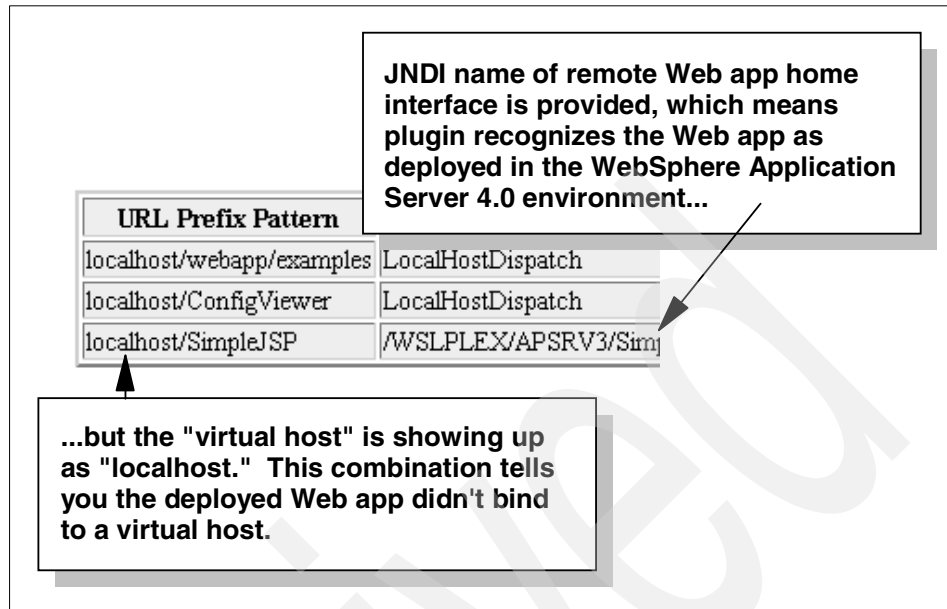


Figure 2-20 Symptom when Web container not configured

The final proof of this can be found in the SYSPRINT of the server region for the application server:

```
Web Container:Configuration File Name:
    /usr/lpp/WebSphere/bin/webcontainer.conf
:
VirtualHost Web Application Context Root Bindings
    /
:
VirtualHost Alias List
    localhost
```

The key information here is the use of the default webcontainer.conf file (located at /usr/lpp/WebSphere/bin) and the localhost alias. A properly configured Web container will have your custom copy of webcontainer.conf, and the virtual host will be your IP host name for the server.

How to correct

This problem can be caused by several things:

- ▶ You simply forgot to configure the Web container.

If this is the problem, then follow the instructions at “Configure the Web container” on page 29.

- ▶ You made a mistake in the pointer to the webcontainer.conf.

This pointer is found in the `jvm.properties` file, and a mistake in typing of any portion of this pointer will cause the server to use the default `webcontainer.conf` file (which contains no virtual host definitions). Check the `SYSPRINT` of the server region and see if the default configuration file is in use. Then check the pointer out of `jvm.properties` and make sure everything—case, spelling—is correct.

Plug-in tries to run the code locally

The general rule of thumb is this: If the plug-in sees a match on “rooturi” in the local `was.conf` file, it'll try to run the Web application locally (in the plug-in). Otherwise, the request will be passed over to the Web container if the Web application is defined over there. It is good practice to follow this rule of thumb and make certain no `was.conf` definitions for your Web application exists if you intend is to run the Web application in the WebSphere Application Server 4.0.1 Web container.

However, it's very easy to imagine a scenario where, in the act of migrating a Web application from the plug-in environment to the Web container environment, you accidentally forget to remove the definitions from `was.conf`. Somewhat surprisingly, this may or may not result in an error. It all depends on whether you have an explicitly coded virtual host in your local `was.conf`.

The version of WebSphere Application Server found in the plug-in has the concept of a “virtual host” but the coding is different. All Web applications defined in the local `was.conf` file must have a `deployedwebapp.<name>.host=` statement. The value found on that statement points to a `host.<name>.alias=` statement also found in the `was.conf` file. What follows the `alias=` on that statement is the virtual host. By default the value is the keyword `localhost`, but you may also have an explicitly coded IP name. If the virtual host is an explicitly coded IP name, and that IP name is identical to a virtual host IP name coded in the WebSphere Application Server 4.0.1 `webcontainer.conf` file, then the plug-in ignores the Web container in favor of the local definition.

The best way to view this is to look at the following example of the Application Dispatching information from the supplied configuration viewer:

URL prefix pattern	JNDI name
<code>wg31.washington.ibm.com:8080/SimpleJSP</code>	<code>WSLPLEX/APSRV3C...</code>
<code>localhost/SimpleJSP</code>	<code>LocalHostDispatch</code>

This shows the URL pattern `/SimpleJSP` being defined in the Web container and the plug-in's local `was.conf`. The difference is this: The Web container's version is bound to virtual host `wg31.washington.ibm.com:8080`, while the local plug-in is using the default `localhost` virtual host.

In this example, the request flows to the Web container only if the URL's host value matches the virtual host of wg31.washington.ibm.com:8080. Any other host value will be run local to the plug-in.

If, however, you have coded a virtual host of wg31.washington.com:8080 in your was.conf and your webcontainer.conf, and the Web application /SimpleJSP is bound to that virtual host in both locations, the plug-in will only recognize the local copy. It'll see the conflict and reject the Web container's definition.

The following discussion shows the error when the plug-in tries to run it locally and fails.

Browser symptom

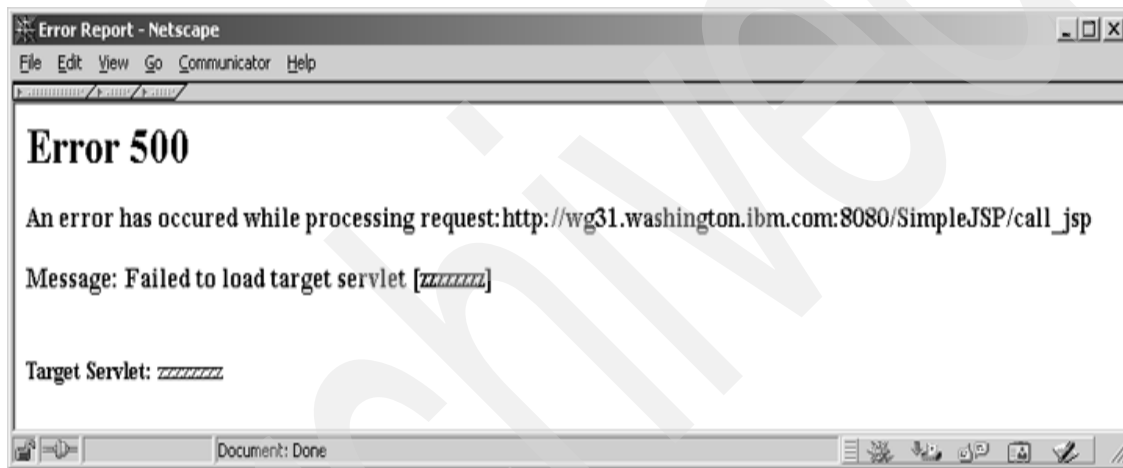


Figure 2-21 Error 500 message

Log or trace symptom

Look in the ncf trace of the plug-in. With appserver.loglevel=WARNING set, you'll see the following:

```
"zzzzzzzz"
"Failed to load servlet"
javax.servlet.ServletException: Servlet [zzzzzzzz]:
    Could not find required servlet class - SimpleJSPServlet.class
```

In this example, the problem is that the servlet class file is not found. This occurs when a servlet is moved from the plug-in environment to the WebSphere Application Server 4.0.1 Web container environment.

Other problems could occur: servlet class file invalid, permission bits too restrictive, etc. The point is that the plug-in is trying to run the servlet, when it should be routing the request over to the WebSphere Application Server 4.0.1 environment.

How to correct

Edit the was.conf file and remove (or comment out) the definitions for the Web application that you wish to run in the WebSphere Application Server 4.0.1 environment.

URL doesn't contain a value that matches the defined context root or virtual host

This problem has two forms:

- ▶ URL doesn't match any Service statement and therefore doesn't get “over the wall” to the plug-in. This problem manifests itself as described in “No Service directive coded that matches the URL received” on page 61.
- ▶ URL gets “over the wall” but doesn't match any context root settings in the Web container. This problem is very similar to that described “WebSphere Application Server 4.0.1 not started” on page 65. The difference here is that the application server is up and running.

Browser symptom

See Figure 2-19 on page 66.

Log or trace symptom

This problem is caused by the received URL not matching an entry in the String Matcher Table maintained by the plug-in. The contents of the ncf trace will be the same as illustrated for “WebSphere Application Server 4.0.1 not started” on page 65.

The Web applications deployed into the WebSphere Application Server 4.0.1 server will show on the Application Dispatching panel (see “Check plug-in Application dispatching information” on page 50).

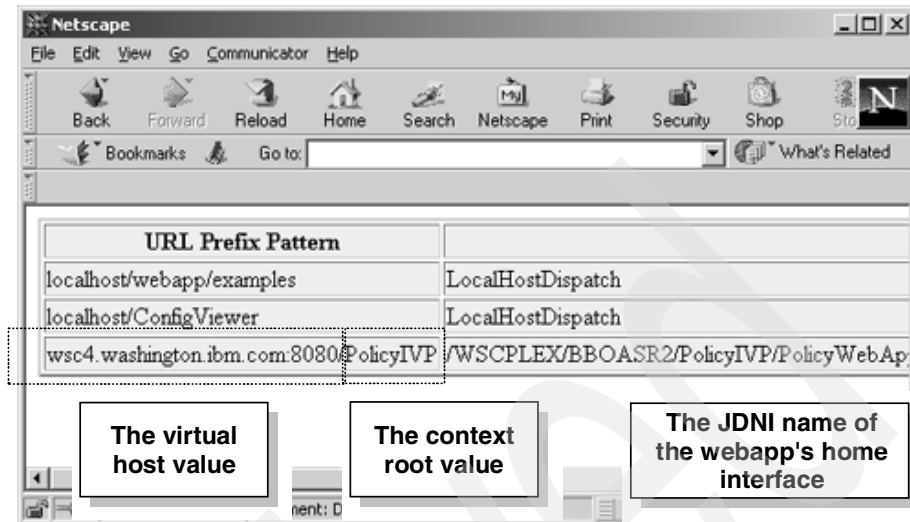


Figure 2-22 Application Dispatching Information screen

If you see the application you're trying to invoke on this screen, then the WebSphere Application Server 4.0.1 application server is up and the plug-in has successfully communicated with the WebSphere Application Server 4.0.1 Web container. Most probably there is a typo in your URL: Either the virtual host is incorrect, or the context root string doesn't match.

How to correct

Provided the Application Dispatching screen shows your application and verifies that the plug-in is talking to the WebSphere Application Server 4.0.1 Web container, visually inspect your URL and make certain of the following two things:

- ▶ The IP host name on the URL is identical (including port information, if any) to the virtual host shown on the Application Dispatching screen. Without an exact match here the plug-in will not associate your URL with the application.
- ▶ The string that follows the first slash matches the characters on your URL exactly, including matching the case of the characters. This is the context root value, and WebSphere Application Server uses that string to match your URL request with a deployed Web application.

Anything not matching between the URL and the information shown on the Application Dispatching screen will prevent the request from being honored and will result in the Web group not defined message.

Your application didn't bind to a virtual host

This is different from the preceding problem. In that one your URL was incorrect. In this one the application you deployed into the Web container contains a context root that didn't bind to any virtual host defined in the webcontainer.conf file. If it doesn't bind to any virtual host, then the plug-in has no knowledge of the application at all.

If your webcontainer.conf file is making use of the single-slash “catch-all” contextroots= setting, then this problem will not occur (that's because the single slash allows any and all Web applications to bind to the virtual host). But this problem may pop up if you are coding more explicit contextroots= values. For example, consider the setting:

```
host.default_host.contextroots=/PolicyIVP
```

and a Web application <context-root> setting of /SimpleJSP. There's no match possible there. The Web application will not bind to the virtual host.

Browser symptom

See Figure 2-19 on page 66.

Log or trace symptom

The key indicator of this problem is the content of the Application Dispatching panel, which will fail to show your application:

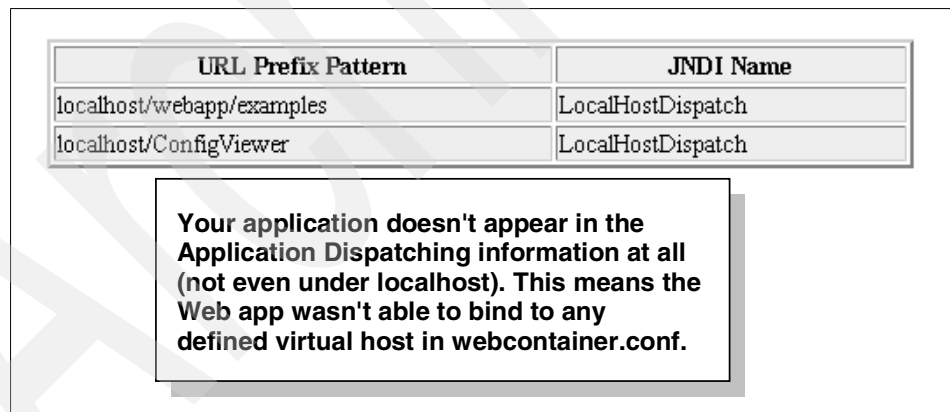


Figure 2-23 Application dispatching when Web application doesn't bind to any virtual host

Furthermore, if you look in the SYSPRINT of the application server region, you'll see something like this:

```
VirtualHost Web Application Context Root Bindings:  
/SimpleJSP
```

```
VirtualHost Bound Web Applications: <none>
VirtualHost Alias List:
  wg31.washington.ibm.com:8080
```

In this example, the test Web application used to force this condition was the only Web application in your Web container, so the value <none> is appearing in Bound Web Applications. If you had other Web applications that did bind properly, they would appear, but the Web application you're debugging would not. Look for the Web application you're debugging. If it's not showing up as bound, then this particular problem is occurring.

How to correct

Modify your webcontainer.conf file and update the contextroots= statement so your Web application will bind. This statement will allow multiple strings, separated by commas:

```
host.default_host.contextroots=/SimpleJSP, /PolicyIVP, /XYZ
```

Your solution may be something as simple as adding another string to the statement. Changing the webcontainer.conf file requires a restart of the application server.

Plug-in not connected to the WebSphere Application Server 4.0.1 runtime you think it is

The plug-in will attempt to communicate with whatever WebSphere Application Server 4.0.1 Systems Management Server (SMS) it finds based on the RESOLVE_IPNAME and RESOLVE_PORT variables in httpd.envvars. If you fail to code those environment variables, the plug-in will by default go to port 900 on the TCP/IP stack on which the plug-in itself is operating. If you have multiple WebSphere Application Server 4.0.1 systems running, it's possible to make a mistake and point your plug-in to the wrong server. If that happens, the Web application you think should be deployed in the Web container might not be accessible by the plug-in.

This problem will show itself in ways nearly identical to "WebSphere Application Server 4.0.1 not started" on page 65 and "URL doesn't contain a value that matches the defined context root or virtual host" on page 71.

Browser symptom

See Figure 2-19 on page 66.

Log or trace symptom

The problem here is that the URL won't match what's found in the String Matcher Table. Therefore, the message you'll see in the ncf trace is:

```
ServletHost    W Web.Group.Not.Found:./SimpleJSP/simple.jsp"
ServletReques X Web Group Not Found
The web group /SimpleJSP/simple.jsp has not been defined
```

You'll not see anything in the logs that indicates that the plug-in is pointed to the wrong IP name and port. You must simply review the configuration of `httpd.envvars` and make sure you have it coded to the proper values. Checking the Application Dispatching information helps isolate this problem.

How to correct

Visually inspect the `RESOLVE_IPNAME` and `RESOLVE_PORT` values in `httpd.envvars` and correct if necessary.

Errors related to request not resolving to Web application class file

The URL request may map to a Service request and be thrown “over the wall” to the plug-in; it may map to a virtual host and context root in the String Matcher Table and be routed over to the Web container, and then still fail.

The ability of a URL request to make its way over to the Web container is based on your coding of the `webcontainer.conf` file and the value of the `<context-root>` XML tag found in `application.xml` of the deployed EAR file. But there's more to the Web application puzzle than that. There is the `web.xml` file inside the Web application's WAR file, and that's where all manner of problems can be introduced:

```
<servlet>
  <servlet-name>Was40Ivp</servlet-name>
  <servlet-class>com.ibm.ws390.samples.ivp.servletclient.Was40Ivp</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>Was40Ivp</servlet-name>
  <url-pattern>/PolicyServlet</url-pattern>
</servlet-mapping>
</web-app>
```

Annotations: 1 points to `<url-pattern>`, 2 points to `<servlet-name>`, 3 points to `<servlet-class>`.

The Web app creation tool would normally be responsible for making sure this was all correct. But that doesn't mean problems can't occur!

Figure 2-24 Where problems can be introduced into the Web applications deployment descriptor

1. The `<url-pattern>` tag contains the servlet mapping string. If the value on the URL doesn't match any `<url-pattern>` string defined in any Web application deployed in the container, then the request fails.

2. The `<servlet-name>` string is what ties together the `<servlet-mapping>` section of the XML file with the `<servlet>` section. If the values don't match one-for-one, then the request fails. (Surprisingly, neither the AAT tool nor the SMS GUI checks for this error.)
3. The `<servlet-class>` tag points to the actual class file that is to be invoked. It's quite possible that the class file name is incorrect, or the class file itself is corrupt or otherwise invalid.

Servlet's mapping string doesn't match

Between the time you create the Web application and set its `<url-pattern>` and the time you issue your first URL against that servlet, you might forget the exact notation of the servlet mapping string. The request will be passed to the plug-in based on a match to a Service statement, and it'll get routed to the Web container based on the `<context-root>` match. But without a `<url-pattern>` match, WebSphere Application Server won't know what specific servlet to invoke.

What happens is that the request will eventually be considered a request for a static file, and the name on the URL won't be the name of a file WebSphere Application Server sees in the HFS. So it'll issue the following error:

Browser symptom

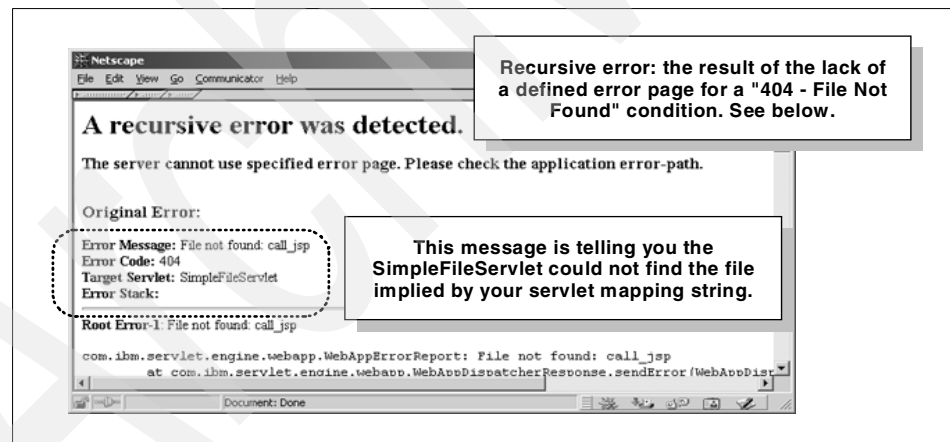


Figure 2-25 Default error page when servletmapping not found and WebSphere Application Server looks for static file without success

This error is somewhat ugly in that the page is really telling you two things: The static file wasn't found (resulting in a 404 error), and the error page for the 404 condition wasn't found either. Error pages for Web applications are defined in the web.xml file, and up to this point the sample web.xml files in this document have not included that XML coding. Here's what that XML stanza would look like to specify a custom 404 error page:

```
<webapp>
:
<servlet>
  <servlet-name>SimpleJSPServlet</servlet-name>
  <servlet-class>SimpleJSPServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>SimpleJSPServlet</servlet-name>
  <url-pattern>/call_jsp</url-pattern>
</servlet-mapping>
<error-page>
  <error-code>404</error-code>
  <location>/404.html</location>
</error-page>
</web-app>
```

The server looks for this page in the root of the HFS directory structure that represents the deployed WAR file. If the <location> string was /subdir/404.html, it would look in the subdirectory /subdir. If you don't have an error page defined, or WebSphere Application Server can't find the error page you specify, you get the "recursive error" symptom.

Log or trace symptom

The symptom of this problem occurs in the WebSphere Application Server 4.0.1 application server region's SYSPRINT. The HTTP Server's "vv" trace shows the request being passed to the plug-in based on a match to a Service statement. The plug-in's ncf trace will show normal operation because the URL will match to a virtual host and context root string. The browser symptom shows enough detail of the problem so that looking in the SYSPRINT is not required.

How to correct

Correct the format of your URL, or change the <url-pattern> value in the web.xml file, refresh your WAR file, regenerate your EAR file, and redeploy the application.

Mismatch in servlet name in deployment descriptor

This problem is fairly obscure, and would occur only if you're hand-building the web.xml file in the WAR file. A Web application construction tool would likely not create this problem. Nevertheless, this symptom would occur any time the `<url-pattern>` value is defined and found by WebSphere Application Server, but no associated servlet can be found in the web.xml file.

Browser symptom

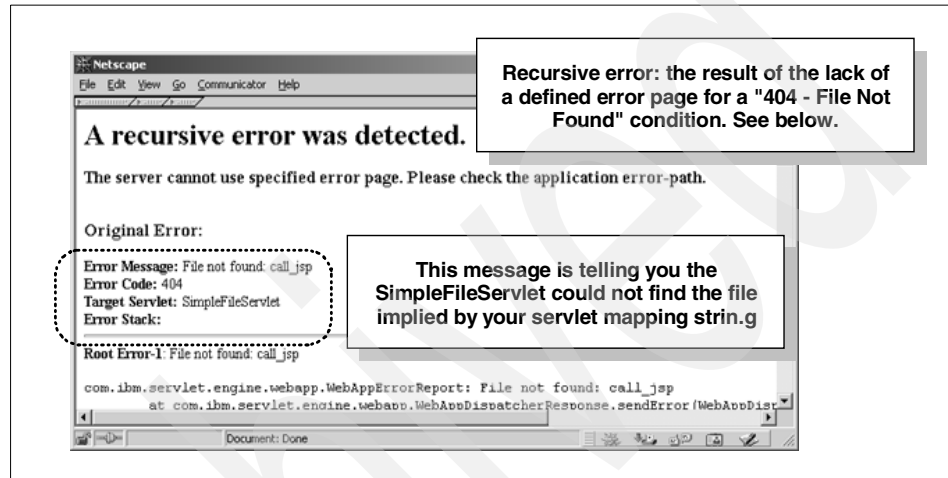


Figure 2-26 Default error page when servletmapping found, but no associated servlet defined

This problem occurs when WebSphere Application Server can't get a match for a servletmapping string: It steps back and assumes the request is for a static file. It'll then go looking for the file, and if it fails it'll throw the default "recursive error" page.

Log or trace symptom

The symptom of this problem occurs in the WebSphere Application Server 4.0.1 application server region's SYSPRINT. The HTTP Server's "vv" trace shows the request being passed to the plug-in based on a match to a Service statement. The plug-in's ncf trace will show normal operation because the URL will match to a virtual host and context root string. The browser symptom shows enough detail of the problem so that looking in the SYSPRINT is not required.

How to correct

Reconstruct the WAR file with a corrected web.xml file. Make certain the `<servlet-name>` string is present in both the `<servlet-mapping>` section as well as the `<servlet>` section.

Class file is incorrect

In this case the servlet mapping string is correct, but the class file referenced in the <servlet-class> tag of the web.xml file isn't correct, and therefore the class file can't be found. The WebSphere Application Server Web container will try to locate the class file, but will fail.

Browser symptom

Unfortunately, the symptom for this problem is very cryptic. What you will see on the browser screen is this:



Log or trace symptom

This problem lies entirely within the WebSphere Application Server 4.0.1 runtime, so the Web server's "vv" trace and the plug-in's ncf trace are of no use. The output provided to the server region's SYSPRINT is by default minimal. If your trace settings in current.env are the default, you will not see any evidence of this problem in the SYSPRINT.

However, if in your current.env file you have the following coded:

```
:  
TRACEALL=1  
TRACEBUFFLOC=SYSPRINT  
TRACEPARAM=00  
:
```

then you'll get some information out to your SYSPRINT that will indicate the problem. Here's what you'll see:

```
:  
"Failed to load servlet": javax.servlet.ServletException:  
Servlet [SimpleJSPServlet]:  
:  
Could not find required servlet class - SimpleJSPServletx.class  
:  
Unexpected internal engine error while sending error to client:  
"/SimpleJSP/call_jsp"  
:
```

How to correct

Inspect your web.xml file and make sure the pointer to the class file in the <servlet-class> tag is correct. Watch for misspellings of the file name, or possibly an error in any of the qualifiers of a longer package name. If you spot an error, correct it, re-assemble the WAR/EAR file and redeploy the application.

2.9.4 Turning on CTRACE

Many subsystems on z/OS write their component trace (CTTRACE) to IPCS. WebSphere Application Server V4.0.1 for z/OS and OS/390 uses this system to write detailed traces, but it can also be used for applications.

The customization jobs allocate the TRACE data, copy CTIBBO00 to SYS1.PARMLIB and copy the trace writer BBOWTR to SYS1.PROCLIB.

Example 2-9 Sample SYS1.PARMLIB(CTIBBO00) member

```
/* ===== */
/* DEFAULT CTIBBO00 MEMBER */
/* ===== */
TRACEOPTS
/* ----- */
/* -Start a ctrace writer. Remove comments to start the PROC */
/* during CB Series address space initialization. */
/* ----- */
/* WTRSTART(BBOWTR) */
/* */
/* ----- */
/* -Indicate that tracing is active for CB Series: */
/* ----- */
ON
/* ----- */
/* -Connect to ctrace external writer (BBOWTR): */
/* (Note that the WTR value must match the value for the started */
/* ctrace external writer, wtrstart.) */
/* ----- */
WTR(BBOWTR)
```

The following steps need to be taken to run a CTRACE:

1. Set the environment variable TRACEALL.
Use “1” unless you have a problem, then use “3”.
2. If the trace writer was not started via the SYS1.PARMLIB(CTIBBOxx) member, start it by entering the following z/OS command:

```
TRACE CT,WTRSTART=BBOWTR
```

2.9.5 The WebSphere logstream

WebSphere writes all critical errors to the defined logstream. To find the name of the logstream, check the syslog or the BBOSMS log for:

```
BB0U0129I          LOGSTREAMNAME: your_Logstream_name
```

Example

```
BB0U0129I          LOGSTREAMNAME: WAS.ERROR.LOG
```

and

```
BB0U0025I ERRORS WILL BE WRITTEN TO WAS.ERROR.LOG LOG STREAM BBOSMS.
```

How to define the logstream

- ▶ See *WebSphere Application Server V4.01 for z/OS and OS/390 Installation and Customization*, GA22-7834.
- ▶ The parameter LOGSTREAMNAME in the environment configuration file defines which logstream a server shall use.
- ▶ Make sure that the LOGSTREAMNAME matches the actual name of a logstream (which can be DASD or CF resident).
- ▶ Set up the necessary RACF authorizations to access the logstream (the RACF class LOGSTRM is being used for this purpose).
- ▶ Client errors can be separated from server errors by defining a CLIENTLOGSTREAM.

How to format a logstream

To look at the error logs you should use the Logstream Browser Utility (BBORBLOG). This utility can be invoked from a TSO command line ("Option 6" in TSO) with the following:

```
EX 'BBO.SBBOEXEC(BBORBLOG)' 'WAS.ERROR.LOG'
```

The system then displays the message IDC0550I ENTRY (A) SYSPROG.WAS.ERROR.LOG DELETED (because it removes the old output from the previous run of the logstream browser), formats the logstream, and then displays the contents of the logstream, which looks like Example 2-10.

Example 2-10 Output of the Logstream Browser utility

```
BROWSE    SYSPROG.WAS.ERROR.LOG                      Line 00000000 Col 001 080
Command ===>                                           Scroll ===> PAGE
***** Top of Data *****
2001/11/13 17:27:13.133 01 SYSTEM=TC1 SERVER=INTFRP01 JobName=BB0IR
                   ASID=0X0034 PID=0X0100000E TID=0X08FB9890 00000000 c=UNK
                   ./bborcarm.cpp+281 ... BB0U0618W ARM READY FAILED - SERVER NOT
```

```
REGISTERE
2001/11/13 17:27:13.049 01 SYSTEM=TC1 SERVER=NAMING01 JobName=BBONM
ASID=0X0033 PID=0X0100000D TID=0X08FB8C38 00000000 c=UNK
./bborcarm.cpp+281 ... BBOU0618W ARM READY FAILED - SERVER NOT
REGISTERE
2001/11/13 17:27:07.395 01 SYSTEM=TC1 SERVER=SYSMGT01 JobName=BBOSMS
ASID=0X0032 PID=0X0100000C TID=0X08FB7FE0 00000000 c=UNK
./bborcarm.cpp+281 ... BBOU0618W ARM READY FAILED - SERVER NOT
REGISTERE
```

Notes:

- ▶ The Logstream Browser writes its output to a data set called <userid>.logstream_name. Because this data set is overwritten each time you invoke the Logstream Browser, you might want to copy/rename it if it contains valuable information before you invoke the Logstream Browser again.
- ▶ You might need to adjust the space allocation in the BBORBLOG script if you want to browse big logstreams.
- ▶ The most recent entry in the logstream is displayed at the top.
- ▶ The timestamp for the traces is based on the GMT time zone.

WebSphere in a sysplex

This chapter describes how to enable WebSphere on the second and subsequent systems in a sysplex. Running WebSphere in a sysplex environment provides you with workload balancing, redundancy, and non-disruptive upgrades. To further enhance your workload management and availability, you can enable Sysplex Distributor. This chapter also describes how to configure Sysplex Distributor.

The following topics are covered:

- ▶ Configuring WebSphere in a sysplex
- ▶ Migration to a new version in a sysplex
- ▶ Configuring Sysplex Distributor
- ▶ The big picture

3.1 Configuring WebSphere in a sysplex

To set up the second system in a sysplex to run WebSphere, do the following:

1. Duplicate the environment you used for the first system, e.g., LINKLIST, LPA, and so forth.
2. Use the Administration application to configure the instances that will run on the second system.
3. Start the instances.

We describe the steps we took to get WebSphere running on the second system. Third and subsequent systems would follow the same steps.

3.1.1 Environment

Chapter 5 of *WebSphere Application Server V4.01 for z/OS and OS/390 Installation and Customization*, GA22-7834 has been used as a reference for configuring WebSphere in a sysplex. We will not continue to make specific references to this document but suggest that you use it in conjunction with the information presented here.

We assumed that the sysplex was configured and running and that DB2 was set up in data-sharing mode. Refer to Appendix C, “DB2 quick setup” on page 359 for assistance. We installed WebSphere on the first system in the sysplex as described in Chapter 2, “WebSphere Application Server” on page 9. There must be an LDAP server running on the second system. We planned to configure all the server instances to run on the second system as shown in Figure 1-2 on page 8.

There are a number of issues to consider when planning to run WebSphere in a sysplex. These need to be addressed when configuring the first system. Refer to Section 2.3, “Planning the installation” on page 13 for details.

3.1.2 Environment configuration checklist

The following checklist shows the configuration we had to establish prior to configuring new WebSphere Application Server instances. Modify these steps according to your environment.

- ▶ Update SCHEDxx to specify BBODAEMN, BBOCTL and BBOSMS in the program properties table.
- ▶ APF authorize:
 - hlq.SBBOLOAD
 - hlq.SBBOLD2

- hlq.SBBOLPA
- SCEERUN
- SDSNLOAD
- ▶ Add to LPA:
 - hlq.SBBOLPA
 - hlq.SBBOMIG
 - hlq.SBBOLOAD - if there is sufficient virtual storage
- ▶ Add to LINKLIST:
 - hlq.SBBOLD2
 - hlq.SBBOLOAD - if it is not placed in LPA due to virtual storage constraints
- ▶ Catalog all WebSphere and LDAP data sets.
- ▶ Update BLSCUSER to allow IPCS processing of WebSphere dumps.
- ▶ Start SMF recording (optional).
- ▶ Update TCP/IP.
 - Add an IP name for the second system to the DNS for the second daemon server instance to use (assuming that the Sysplex Distributor has not been configured yet).
 - Update your TCP/IP profile to add the following ports:
 - 900 TCP SYSMGT02
 - 5555 TCP DAEMON02 SHAREPORT
 - 5556 TCP DAEMON02 SHAREPORT ; SSL port number
- ▶ Configure LDAP.

LDAP needs to run on any system where the Naming Repository is running. It should already be using shared DB2 tables that contain your existing configuration. The configuration files for LDAP are held in the shared WebSphere configuration HFS in /WebSphereCFG/sysplex/etc/ldap/. You need to copy and modify these files to refer to the DB2 subsystem running on the second system.

 - Copy *sysid1.bboslaped.conf* to *sysid2.bboslaped.conf* and change *servername* to the DB2 location name.

Attention: The location name is specified when DB2 is installed. It is the location of the entire data sharing group. It is *not* the DB2 subsystem ID, although it may have the same value on the first DB2 system installed in the sysplex.

- Configure LDAP to run in multiserver mode.

If multiple LDAP server instances are using the same DB2 database to store directory information, all LDAP server instances using that database must be operating in multi-server mode, either with or without dynamic workload management enabled. More information on this can be found in Chapter 8 of *SecureWay Security Server LDAP Server Administration and Use*, SC24-5923, - “Operating in Multi-server Mode With Dynamic Workload Management Enabled”.

- Update *sysid1.bboslapi.conf* and *sysid2.bboslapi.conf* as follows:

```
sysplexGroupName BBOSANDBOX
sysplexServerName SC63      <===== change to SC64 for sysid2
multiserver      y
tbspacemutex     BBOMUTX
```

- Copy *sysid1.dsnaoini* to *sysid2.dsnaoini* and change MVSDEFAULTSSID, DB2 subsystem name, and the DATA SOURCE to the DB2 subsystem ID on the second system.
- Copy *sysid1.bboidif.cb* to *sysid2.bboidif.cb* (no changes required).
- ▶ Copy your *db2sqljdb.properties* file, giving it a new name, and update it to specify DB2SQLJSSID= the DB2 subsystem ID on the second system.
- ▶ Copy started task procedures from the first system.
- ▶ Activate your changes either dynamically with the commands shown in E.1, “General commands” on page 392, or with an IPL.

If you dynamically changed your environment, be sure to make the changes permanent. We recommend that you update your automation to load the WebSphere modules into LPA after an IPL.

3.1.3 Defining the new system to WebSphere

We followed the directions in Chapter 5 of *WebSphere Application Server V4.01 for z/OS and OS/390 Installation and Customization*, GA22-7834, to define the new system (SC64) and the server instances that would run on it. Figure 3-1 on page 87 shows the definitions we created.

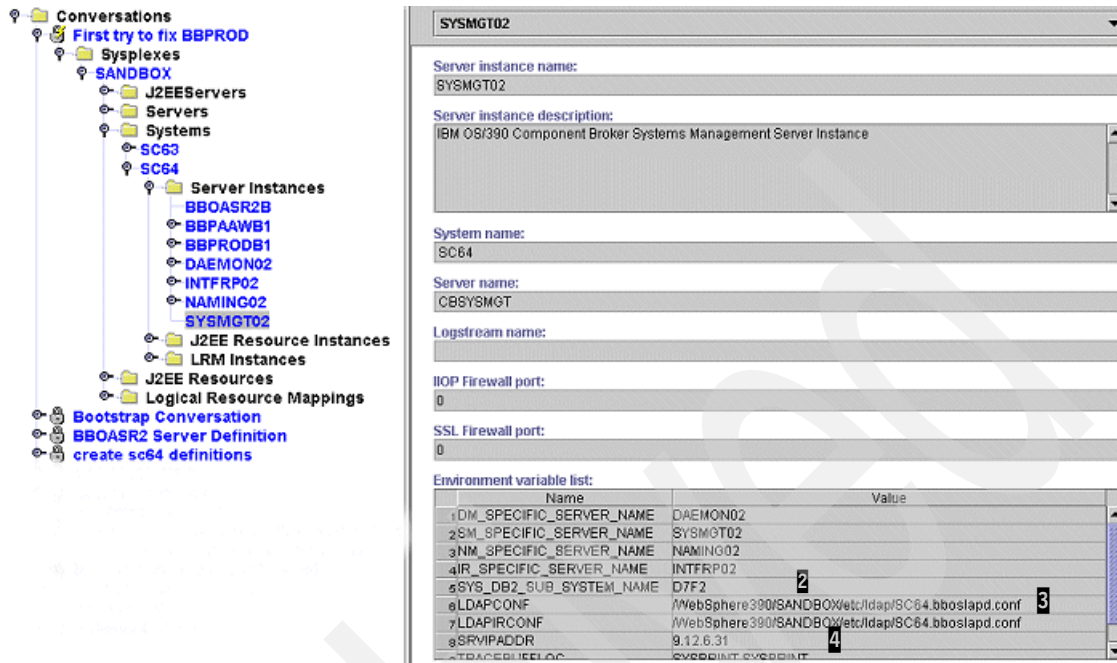


Figure 3-1 Server definition

We are looking at the configuration of the system management server instance on the second system in the sysplex.

1. The server instance names change for the second and subsequent systems in the sysplex.
2. The DB2 subsystem name is the subsystem ID on this system. It is not the location name for the data sharing group.
3. We need to use modified LDAP configuration files that refer to DB2 on the second system.
4. We are using two TCP/IP stacks. SRVIPADDR specifies the IP address of the system from the appropriate stack.

Next we define a J2EE server instance and a J2EE resource instance for the second system. Our configuration is shown in Figure 3-2.

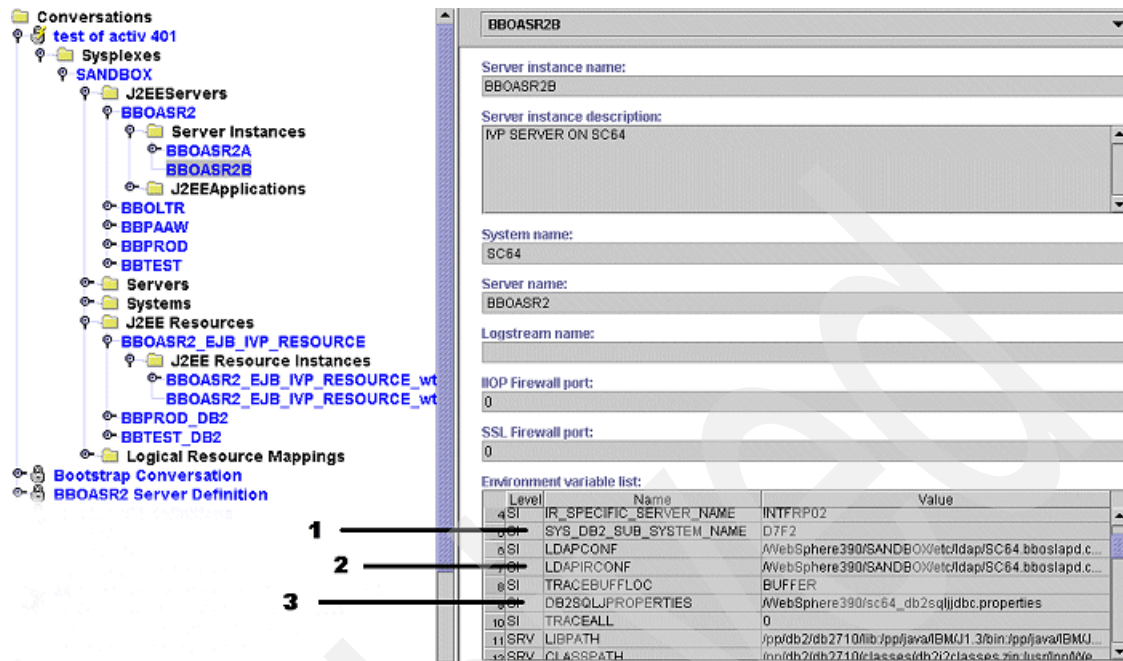


Figure 3-2 J2EE server instance definition

1. The DB2 subsystem name for the second system (D7F2).
2. We need to use the new LDAP configuration files we created earlier (see environment variables).
3. We need to use a new db2sqljdb.properties file that sets DB2SQLJSSID to the db2subsystem ID on the second system.

The variables will overwrite the values specified for the first system in the J2EE server definition.

You should now be able to start WebSphere on the second system. There is no workload balancing until the Sysplex Distributor is running, but you can test your configuration by running the supplied IVPs. Refer to “Steps for setting up the Web application IVPs” in *WebSphere Application Server V4.01 for z/OS and OS/390 Installation and Customization*, GA22-7834 for the customization required for the application server BBOASR2B on the second system.

When all your regions are started, **netstat** shows some of the connections between them. It is only a snapshot. Connections can be created, then disconnected, and these will not be displayed. Be patient, startup can take a long (15 min.) time. Example 3-1 shows how our system looks with an IVP running, an administration application connected, and a Web transaction running on the system.

Example 3-1 netstat command output

User Id	Conn	Local Socket	Foreign Socket	State
-----	----	-----	-----	----
BBOASR2A	00001609	0.0.0.0..1052	0.0.0.0..0	Listen
BBOASR2A	00001626	201.3.10.10..1055	201.3.10.10..900	Establish
BBOASR2A	00001606	0.0.0.0..1051	0.0.0.0..0	Listen
BBOASR2S	0000163B	192.168.70.4..1057	192.168.70.4..1389	Establish
BBOIVP63	00001662	201.3.10.10..1058	201.3.10.10..900	Establish 1
BBOLDAP	000015AE	0.0.0.0..1389	0.0.0.0..0	Listen
BBOLDAP	0000163C	192.168.70.4..1389	192.168.70.4..1057	Establish
BBOLDAP	00001622	192.168.70.4..1389	192.168.70.4..1054	Establish
BBWEBSRV	00001617	201.3.10.10..1053	201.3.10.10..900	Establish 2
BBWEBSRV	00001675	201.3.10.10..80	9.12.2.119..1132	Establish 3
BBWEBSRV	0000160A	0.0.0.0..80	0.0.0.0..0	Listen
BBWEBSRV	00001621	192.168.70.4..1054	192.168.70.4..1389	Establish
DAEMON01	000015F9	201.3.10.10..5555	9.12.2.119..3266	Establish 4
DAEMON01	000015B8	0.0.0.0..5555	0.0.0.0..0	Listen
INTFRP01	000015CE	0.0.0.0..10003	0.0.0.0..0	Listen
INTFRP01	000015CA	0.0.0.0..10002	0.0.0.0..0	Listen
NAMING01	000015C8	201.3.10.10..1049	0.0.0.0..0	Listen
NAMING01	000015CD	201.3.10.10..1050	0.0.0.0..0	Listen
NAMING01	000015FB	201.3.10.10..1049	9.12.2.119..3270	Establish 4
SYSMGTO1	000015C2	201.3.10.10..1048	0.0.0.0..0	Listen
SYSMGTO1	000015F6	201.3.10.10..900	9.12.2.119..3265	Establish 4
SYSMGTO1	000015C0	201.3.10.10..900	0.0.0.0..0	Listen
SYSMGTO1	00001618	201.3.10.10..900	201.3.10.10..1053	Establish 2
SYSMGTO1	00001663	201.3.10.10..900	201.3.10.10..1058	Establish 1
SYSMGTO1	00001627	201.3.10.10..900	201.3.10.10..1055	Establish
BBOIVP63	00001665	0.0.0.0..3668	*.*.*	UDP
SYSMGTO1	00001664	0.0.0.0..3667	*.*.*	UDP

1. This is the connection between the batch IVP and a system management server instance.

2. This is the connection between the Web server and a system management server instance.

3. This is the connection between the browser and the Web server. This connection is transient and will disappear when the transaction completes.

4. This is the connection between the Administration application and the daemon on this system. A new system management server instance and the naming server are also connected to the Administration application.

3.2 Migration to a new version in a sysplex

We migrated from 4.0.0 to 4.0.1 to demonstrate a non disruptive upgrade in a sysplex environment. This requires a warm start of WebSphere. The instructions we followed were in Chapter 6 of *WebSphere Application Server V4.01 for z/OS and OS/390 Installation and Customization*, GA22-7834, and Part 3 of *WebSphere Application Server V4.01 for z/OS and OS/390 Migration*, GA22-7860. This migration strategy assumes that you have shared HFSs configured and that your root file system can be shared between systems. A prerequisite is that you are running OS/390 2.9, z/OS1.1 or later.

3.2.1 HFS structure

The HFS structure we used is described in detail in Appendix B of *WebSphere Application Server V4.01 for z/OS and OS/390 Installation and Customization*, GA22-7834. We needed a design that would allow us to upgrade WebSphere without impacting other users on the system. We also needed to be able to change from the old WebSphere HFS to the new version on one system, without impacting the other WebSphere Application Servers running in the sysplex. We discuss this relative to WebSphere—the technique is applicable to other products, such as JDK, DB2 etc. Our file structure is partially shown in Figure 3-3 on page 91.

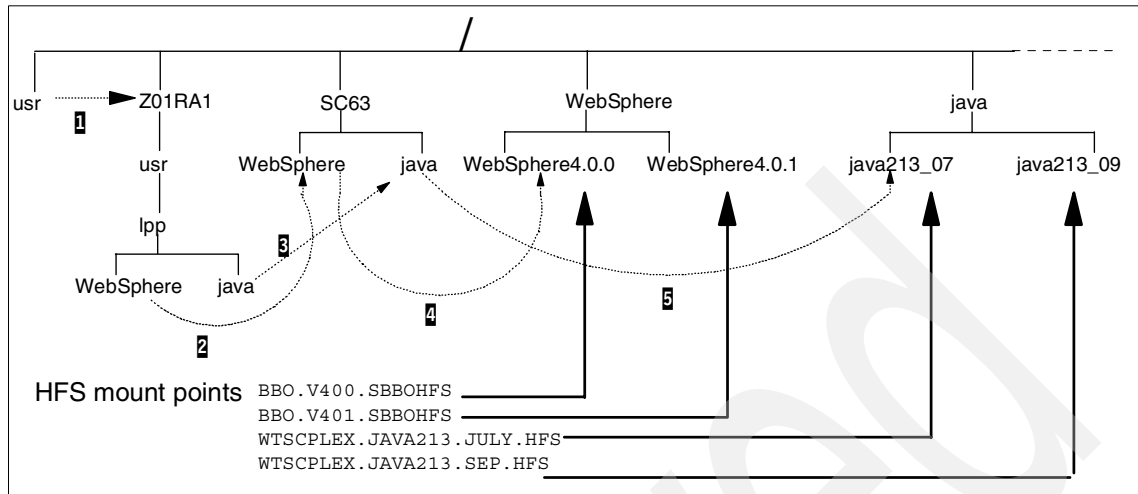


Figure 3-3 HFS structure

On this system, \$VERSION resolved to Z01R01, \$SYSNAME to SC63. Our root file system was shared. We used the following commands to build the file system:

```
SC63: /> cd /
1 SC63: /> ln -s \ $VERSION /usr usr
SC63: /> cd Z01RA1
2 SC63: /Z01RA1 /usr /lpp> ln -s \ $SYSNAME /WebSphere WebSphere
3 SC63: /Z01RA1 /usr /lpp> ln -s \ $SYSNAME /java java
SC63: /> cd SC63
4 SC63: /SC63> ln -s /WebSphere /WebSphere4.0.0 WebSphere
5 SC63: /SC63> ln -s /java /java213_07 java
```

1 First, the \$VERSION symbolic is used to link to a specific version of the root file system—in our case Z01RA1. All or some of the systems in the sysplex share the files in this file system. Other systems could be at higher or lower levels, depending on the compatibility levels. Our system, SC63, ran at the Z01RA1 level of the root file system.

2 If we had mounted the WebSphere product HFS directly to this mount point (/Z01RA1/usr/lpp/WebSphere), then when we upgraded WebSphere we would have had to stop WebSphere on *every* system using this root file system. Instead, we created a symbolic link to a file system that contained all our system-specific information.

3 We created a similar link for the Java for z/OS product HFS.

4 Our system-specific information was in /SC63. Here we created a link to the version of WebSphere that we were using.

5 We created a link to the version of Java we were using.

Each system in a sysplex has its own link to WebSphere. All links will be the same if WebSphere is the same version everywhere. When you want to upgrade, a link can be changed without impacting any WebSphere users on other systems.

Every version of the WebSphere product HFS has its own mountpoint, which is shared by all the systems in the sysplex. Thus, there are not multiple copies of the HFS, with one mounted on each system.

On a second system, \$VERSION resolves to Z02RC1, and \$SYSNAME resolves to SC64. It ran with a different version of WebSphere and Java.

```
SC63: />cd SC64
SC63:/SC64> ln -s /WebSphere/WebSphere4.0.1 WebSphere
SC63:/SC64> ln -s /java/java213__09 java
```

Finally, we mounted our product HFSs as shown. The following commands should be entered on a single line:

```
MOUNTFILESYSTEM('BBO.V400.SBB0HFS') MOUNTPOINT('/WebSphere/WebSphere4.0.0)
type(HFS) MODE(RDW)
MOUNTFILESYSTEM('BBO.V401.SBB0HFS') MOUNTPOINT('/WebSphere/WebSphere4.0.1)
type(HFS) MODE(RDW)

MOUNTFILESYSTEM('WTSCPLEX.JAVA213.JULY.HFS') MOUNTPOINT('/java/java213_07)
type(HFS) MODE(RDW)

MOUNTFILESYSTEM('WTSCPLEX.JAVA213.SEP.HFS') MOUNTPOINT('/java/java213_09)
type(HFS) MODE(RDW)
```

3.2.2 Migration procedure

To migrate from WebSphere Application Server 4.0.0 to WebSphere Application Server 4.0.1 we needed to upgrade the JDK and perform a warmstart of WebSphere. Details of the warmstart process can be found in Chapter 6 of *WebSphere Application Server V4.01 for z/OS and OS/390 Installation and Customization*, GA22-7834.

We are describing a general migration strategy. For other version migrations, the specifics will change. We recommend that you read the appropriate migration documentation before proceeding. Your implementation may also vary slightly because your z/OS environment could differ from ours.

These are the steps we performed to migrate:

1. Install the new JDK and WebSphere code.
2. Implement the HFS structure described in 3.2.1, “HFS structure” on page 90.
Mount the new file systems.
3. Back up HFS configuration files, System Management and LDAP databases, WebSphere installation data sets (WAS.CNTL and WAS.DATA), and the WebSphere procedures.
4. Stop WebSphere and LDAP on the system being migrated.
5. Run the *new* customization dialog.
 - Load old variables.
 - Update references to the new load libraries etc., if required.
 - Update references to reflect DB2 on this system.
6. Update the z/OS environment as required.

Refer to E.1, “General commands” on page 392 for examples of the following tasks.

- APF-authorize new libraries.
- Delete old libraries from LPA.

This can be done dynamically but is cumbersome, as each member requires an individual DELETE command. We provided a sample job in E.1, “General commands” on page 392.

- Add new libraries to LPA.
- Delete old libraries from the LINKLIST.
- Add new libraries to the LINKLIST.
- If you make these changes dynamically, be sure to update permanent PARMLIB members.

7. Perform the required migration actions (4.0.0 to 4.0.1). Before submitting these jobs, check that they refer to the new libraries. Submit BBOMCFG, BBOMPAT2, and BBOBIND. These jobs need only be run once for the sysplex. You will need a uid of 0 to run BBOMCFG.
8. Go to the system-specific directory (/SC63) and delete the old symlinks. Add the new links for Java and WebSphere, e.g.:

```
SC63:/SC63> ln -s /WebSphere/WebSphere4.0.1 WebSphere
```
9. Start LDAP.
10. Start the Daemon and Application servers.

After these have started, you should see a “ready for warm start” message. Do not perform a warmstart at this time.

11. Download and install a new version of the Administration and Operations applications.

The upgrade is complete for the first system in the sysplex. The other systems should continue to operate without problems.

12. For all of the other systems in the sysplex, perform steps 4, 6, 8, 9, and 10.

Note that for other version migrations, where step 7 differs from here, other local customization may be required.

All systems in the sysplex should now be running and have “ready for warm start” messages.

13. For each system, do this:

- Shut down the Daemon and Application servers.

- Warmstart the daemon.

```
S BBOBMN,SRVNAME=DAEMON01',PARMS='-ORBCBI WARM'
```

- Warmstart all the application servers that run on this system.

The migration is now complete; all systems have been upgraded with minimal disruption.

3.3 Configuring Sysplex Distributor

Here we provide a short description of the three main approaches available to TCP/IP users in a sysplex to perform load balancing and to accomplish high availability.

3.3.1 Sysplex objectives

The increasing demands of network servers led to the creation of techniques to address performance requirements when a single server is not capable of providing the availability and scalability demands placed on it by its clients. Specifically, network solutions make use of what is referred to as the *clustering technique*, whereby multiple servers are associated into a group to provide sufficient processing power and availability characteristics to handle the demands of their clients.

You can utilize the clustering approach to increase server availability and processing capability. This approach attempts to provide mechanisms by which you can ensure the viability of the cluster in an environment containing a large number of clients generating a potentially high number of requests. To do so, the clustering technique can provide for two main objectives: high availability and load balancing. In some cases, clustering techniques address only high availability, as is the case with Dynamic VIPA. It provides for availability in the case of potential TCP/IP stack or z/OS image failures.

In other cases, the intent is to provide for both high availability and load balancing. This is done by the Domain Name System/Workload Manager solution (DNS/WLM), Network Dispatcher, or the Sysplex Distributor, which is part of z/OS Communication Manager.

In general, load balancing refers to the ability to utilize different systems within the cluster simultaneously, thereby taking advantage of the additional computational function of each. Further, clustering techniques addressing load balancing lead to other system requirements, such as a single system-wide image (one identity by which clients access the system), horizontal growth, and ease of management.

We now discuss the three main approaches available to TCP/IP users in a sysplex to perform load balancing and to accomplish high availability. These techniques can (and usually do) make use of the MVS Workload Manager to distribute IP traffic across a number of servers, but they differ in the approach they take.

DNS/WLM solution

The DNS solution is based on the DNS name server and the z/OS Workload Manager. This solution is only available with the BIND 4.9.3 name server and not with the BIND 9 name server. Intelligent sysplex distribution of connections is provided through cooperation between WLM and DNS. For customers who elect to place a name server in a z/OS sysplex, the name server can utilize WLM to determine the best system to service a given client request.

In general, DNS/WLM relies on the hostname to IP address resolution for the mechanism by which to distribute load among target servers. Hence, the single system image provided by DNS/WLM is that of a specific hostname. Note that the system most suitable to receive an incoming client connection is determined only at connection setup time. Once the connection is made, the system being used cannot be changed without restarting the connection. This solution has the general problem of client caching. In most cases clients cache the address, and so there is no DNS resolution any more.

The DNS approach works only in a sysplex environment. If the server applications are not all in the same sysplex, then there can be no single WLM policy and no meaningful coordination between WLM and DNS.

External IP workload balancing

The IBM Network Dispatcher (part of WebSphere Edge Server) and Cisco's Multi-Node Load Balancer (MNLB) are examples of external IP workload balancing solutions. Such solutions exist outside the sysplex, but may route work into the sysplex. Where DNS/WLM resolves a domain name to different IP addresses as a means of balancing work, external IP workload balancing solutions define a single IP address representing all instances of the server, and then balance new work requests (new TCP connection requests) among available servers. These external solutions rely on an agent in the sysplex to deliver Workload Manager information for nodes on which application instances reside. All hosting application instances have the same IP address defined as a hidden or loopback address. This means that normal IP routing cannot be used between the decision point and the target stack, so that either the decision point must be directly connected with the target server—with no intervening routers, same LAN segment—or another solution such as Generic Routing Encapsulation, or other proprietary solutions, must be used.

Although the Network Dispatcher solution functions with separate hosts (not part of the same sysplex), load balancing relies on WLM data from individual systems or individual sysplexes and the Network Dispatcher's own perception of the workload, which is based on the TCP/IP traffic.

There is an important performance advantage related to Network Dispatcher, which supports session affinity (HTTP session), which means that it can route work to the server address space that was used by earlier client/server interactions. The session data of the conversation can be reused from the server address space. This is much faster than obtaining the session data from external storage.

Sysplex Distributor

Sysplex Distributor (SD) is one solution for connection-dispatching technology among z/Series IP servers. Essentially, SD extends the notion of Dynamic VIPA and Automatic VIPA takeover to allow for load distribution among target servers within the sysplex. It combines technology used with Network Dispatcher for the distribution of incoming connections with that of Dynamic VIPAs to ensure high availability of a particular service within the sysplex.

Technically speaking, the functionality of SD is similar to that of Network Dispatcher in that one IP entity advertises ownership of some IP address by which a particular service is known. In this way, the single system image of SD is also that of a special IP address. However, in the case of SD, this IP address (known as the cluster address in Network Dispatcher) is called a distributed DVIPA. The IP entity advertising the distributed DVIPA (and dispatching connections destined for it) is itself a system image within the sysplex, and is referred to as the distributing stack. An inbound packet destined for that DVIPA flows through the primary distributing stack, which then forwards the packet over an internal link (XCF, IUTSAMEH, or IQDIO) to the selected target stack.

Like Network Dispatcher and DNS/WLM, SD also makes use of Workload Manager (WLM) and its ability to gauge server load. In this paradigm, WLM informs the distributing stack of this server load so that the distributing stack may make the most intelligent decision regarding where to send incoming connection requests. Additionally, SD can specify certain policies within the Policy Agent so that it may use QoS information from target stacks in addition to the WLM server load. Further, these policies can specify which target stacks are candidates for clients in particular subnetworks.

As with Network Dispatcher, connection requests are directed to the distributing stack of SD. The stack selects which target server is the best candidate to receive an individual request and routes the request to it. It maintains state so that it can forward data packets associated with this connection to the correct stack, but it does not yet support HTTP sessions, like Network Dispatcher. Additionally, data sent from servers within the sysplex need not travel through the distributing stack.

For more information, see Chapter 14 in *WebSphere Application Server V4.01 for z/OS and OS/390 Assembling Java 2 Platform, Enterprise Edition (J2EE) Applications*, SA22-7836.

Load Balancing using Sysplex Distributor with CISCO routers

The Cisco Multi-Node Load Balancer (MNLB) provides a workload balancing function that distributes traffic through Cisco routers across multiple destination TCP/IP stacks. The MNLB consists of a Service Manager (the Cisco Local Director, which is denoted by a cluster IP address) and a set of Forwarding Agents (Cisco routers). For a TCP connection to the cluster IP address, the Forwarding Agent sends the SYN packet to the Service Manager, which then selects a target stack and notifies the Forwarding Agent of this decision. The Forwarding Agent then sends all future packets for that TCP connection directly to the target stack. This solution also uses WLM to gather target server performance statistics.

You can also use a combination of the Sysplex Distributor and the MNLB to provide workload balancing.

The scope of a cluster IP address managed by SD is a single sysplex, and integration with Cisco forwarding agents merely allows the SD distributing stack to be bypassed for inbound traffic. If workload balancing across nodes in multiple clusters (sysplexes) is desired, MNLB using Cisco Local Director as the service manager can be used. Sysplex Distributor will continue to advertise network ownership of the cluster IP address with any attached routing daemon so that SD appearance and behavior toward the attached routing network is unchanged except for its new relationship with Cisco forwarding agents.

This offers the choice of providing the workload distribution inside the sysplex, outside the sysplex, or a combination of both.

3.3.2 Sysplex Distributor implementation

We chose to implement the Sysplex Distributor so clients receive the benefits of workload distribution provided by both Workload Manager (WLM) and Quality of Service (QoS) Policy Agent. In addition, SD ensures high availability of the IP applications running on the sysplex cluster, even if one physical network interface fails or an entire IP stack or z/OS is lost. Also, it can be implemented without the need of additional software or hardware. Remember the HTTP session limitations.

Here are the steps needed to implement SD:

1. Choose which IP stack is going to execute the SD distributing function.
2. Select which IP stacks are going to be the backup stack for the Sysplex Distributor stack and in which order.
3. Ensure that WLM GOAL mode is enabled in all the LPARs participating in the Sysplex Distributor.
4. Enable sysplex routing in all the IP stacks participating in the Sysplex Distributor with the `sysplexROUTING` statement.
5. For those IP stacks that are active in a multi-stack environment, the same host links have to be created dynamically. In general, code `DYNAMICXCF` in all the IP stacks participating in the Sysplex Distributor.

Note: For Sysplex Distributor, you cannot specify the XCF address using the `IUTSAMEH` `DEVICE`, `LINK`, and `HOME` statements. XCF addresses have to be defined through `IPCONFIG DYNAMICXCF`.

6. Code `DATAGRAMFWD` in all IP stacks participating in the Sysplex Distributor.

7. Select, by port numbers, the applications that are going to be distributed using the SD function. Note that if the application chosen requires data and control ports, both ports have to be considered.
8. Code the VIPADYNAMIC/ENDVIPADYNAMIC block for the distributing IP stack:
 - Define the dynamic VIPA associated to the distributing IP stack with the VIPADEFINE statement.
 - Associate the sysplex Dynamic VIPA to the application's port number with the VIPADISTRIBUTE statement.
9. Code the VIPADYNAMIC/ENDVIPADYNAMIC block for the distributor's backup IP stack. Define the IP stack as backup for the sysplex DVIPA address with the VIPABACKUP statement.
10. VTAM must have XCF communications enabled by specifying XCFINIT=YES as a startup parameter or by activating the VTAM major node, ISTLSXCF.

Implementation example

We implemented SD on two LPARs, SC63 and SC64, each running z/OS V1R2.0 Communication Server. We decided to run SD on a different IP stack, which we called TCPIPF. Using different IP stacks is again because of availability reasons. The following are our implementation steps:

Sysplex Distributor definitions

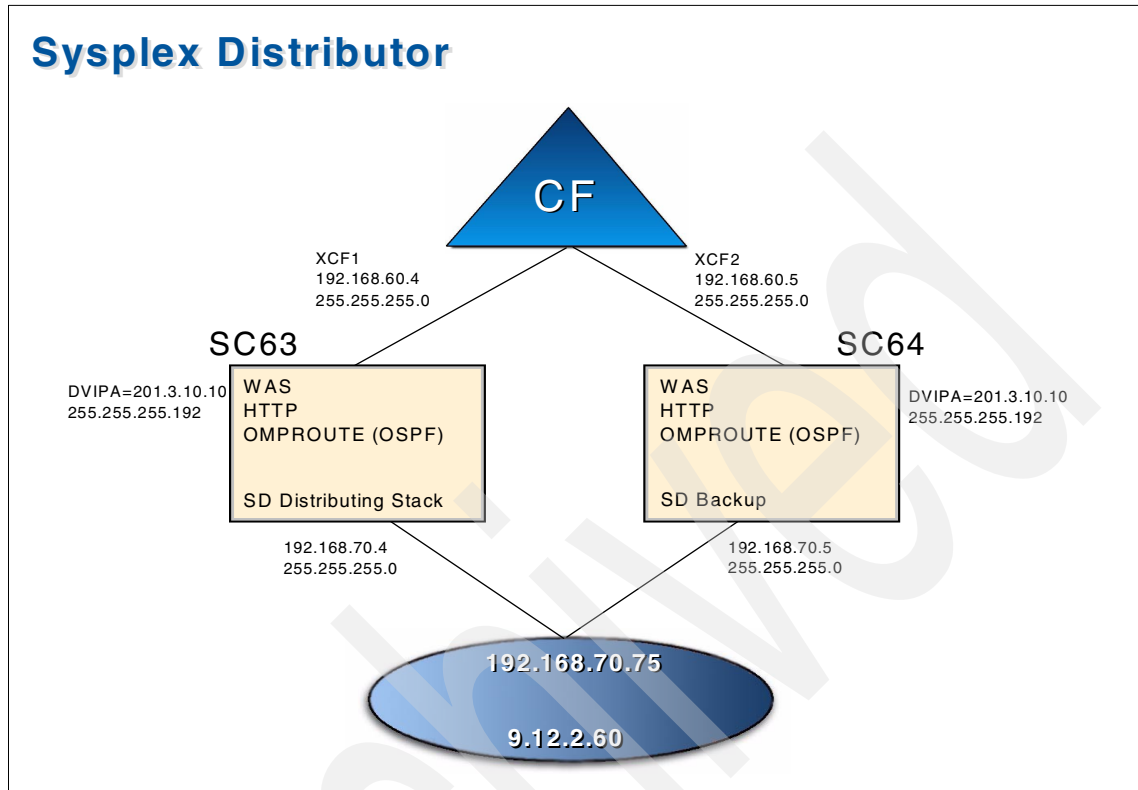


Figure 3-4 Sysplex Distributor configuration

1. BPXPRMxx PARMLIB changes:

- Create a temporary BPXPRMCC, and we added:

```
SUBFILESYSTYPE NAME(TCPIPF)
TYPE(CINET)
ENTRYPOINT(EZBPFINI)
```

Issue SETOMVS RESET=(CC) to add dynamically.

- Update BPXPRMxx for the next IPL.

2. We used the command D WLM,SYSTEMS to ensure that all the LPARs participating in the Sysplex Distributor were running in GOAL mode.

Example 3-2 WLM command output

```
D WLM,SYSTEMS
IWM025I 11.18.31 WLM DISPLAY 624
ACTIVE WORKLOAD MANAGEMENT SERVICE POLICY NAME: DAYTIME
```

```

ACTIVATED: 2001/11/06 AT: 16:37:17 BY: COOK1 FROM: SC64
DESCRIPTION: Policy desgined for DB2 SP
RELATED SERVICE DEFINITION NAME: DAYTIME
INSTALLED: 2001/11/06 AT: 16:36:37 BY: COOK1 FROM: SC64
WLM VERSION LEVEL: LEVEL011
WLM FUNCTIONALITY LEVEL: LEVEL013
WLM CDS FORMAT LEVEL: FORMAT 3
STRUCTURE SYSZWLM_WORKUNIT STATUS: CONNECTED
STRUCTURE SYSZWLM_OECB2064 STATUS: CONNECTED
*SYSNAME* *MODE* *POLICY* *WORKLOAD MANAGEMENT STATUS*
SC63 GOAL DAYTIME ACTIVE
SC64 GOAL DAYTIME ACTIVE

```

3. We chose system SC63 to be the SD distributing stack and SC64 to be the backup:

a. TCPIP profile definitions on SC63

Example 3-3 TCPIP profile on SC63

```

;
IPCONFIG DATAGRAMFWD VARSUBNETTING sysplexROUTING
        DYNAMICXCF 192.168.60.4 255.255.255.0 2

VIPADYNAMIC
VIPADefine MOVE IMMEDIATE 255.255.255.192 201.3.10.10
VIPADistribute DEFINE 201.3.10.10 PORT 900 5555 80 1389
        DESTIP 192.168.60.4 192.168.60.5
ENDVIPADYNAMIC
;
DEVICE OSA2880 MPCIPA AUTORESTART
LINK OSA2880LNK IPAQNET OSA2880
START OSA2880
;
HOME
        192.168.70.4 OSA2880LNK
;
GATEWAY
;
        192.168.70 = OSA2880LNK 1492 0
;
DEFAULTNET 192.168.70.75 OSA2880LNK 1492 0
;
AUTOLOG 5
OMPROUTF ; OMROUTE
ENDAUTOLOG
;
PORT
        900 TCP SYSMGT01 ; port number for SYSMGT01
        5555 TCP DAEMON01 SHAREPORT ; port number for DAEMON01

```

```

5555 TCP BBODMN                ; port number for BBODMN
5556 TCP DAEMON01 SHAREPORT    ; SSL port number for DAEMON01
5556 TCP BBODMN                ; SSL port number for BBODMN
1389 TCP BBOLDAP               ; port number for BBOLDAP

```

b. TCPIP profile definitions on SC64

Example 3-4 TCPIP profile on SC64

```

;
IPCONFIG DATAGRAMFWD VARSUBNETTING sysplexROUTING
        DYNAMICXCF 192.168.60.5 255.255.255.0 2

VIPADYNAMIC
VIPABACKUP 90 201.3.10.10
ENDVIPADYNAMIC
;
DEVICE OSA2880 MPCIPA AUTORESTART
LINK OSA2880LNK IPAQNET OSA2880
START OSA2880
;
HOME
192.168.70.5 OSA2880LNK
;
GATEWAY
;
192.168.70 = OSA2880LNK 1492 0
;
DEFAULTNET 192.168.70.75 OSA2880LNK 1492 0
;
AUTOLOG 5
OMPROUTF ; OMPROUTE
ENDAUTOLOG
;
PORT
900 TCP SYSMGT02                ; port number for SYSMGT02
5555 TCP DAEMON02 SHAREPORT    ; port number for DAEMON02
5556 TCP DAEMON02 SHAREPORT    ; SSL port number for DAEMON02

```

Note:

- ▶ We chose 201.3.10.10 as Dynamic VIPA address assigned to a sysplex cluster for the WebSphere Application Server.
 - ▶ We coded the WebSphere Application Server, LDAP, and HTTP ports in VIPADISTRIBUTE to be distributed.
 - ▶ We wanted the Dynamic VIPA to be taken back as soon as the original IP stack was restarted (MOVE IMMEDIATE) in the event of a failure.
4. We updated the TCPDATA with our Hostname and DOMAINORIGIN.

a. TCPDATA definition on SC63

Example 3-5 TCPDATA data set on SC63

```
TCPIPJOBNAME TCPIPF
HOSTNAME WTSC63F
DOMAINORIGIN ITS0.IBM.COM
NSINTERADDR 9.12.14.7
NSPORTADDR 53
RESOLVEVIA UDP
RESOLVETIMEOUT 20
RESOLVERUDPRETURNS 1
DATASETPREFIX TCPIPF
MESSAGECASE MIXED
```

b. TCPDATA definition on SC64

Example 3-6 TCPDATA data set on SC64

```
TCPIPJOBNAME TCPIPF
HOSTNAME WTSC64F
DOMAINORIGIN ITS0.IBM.COM
NSINTERADDR 9.12.14.7
NSPORTADDR 53
RESOLVEVIA UDP
RESOLVETIMEOUT 20
RESOLVERUDPRETURNS 1
DATASETPREFIX TCPIPF
MESSAGECASE MIXED
```

5. Add IP addresses and the hostnames for SC63, SC64, and the DVIPA in /etc/hosts.

Example 3-7 /etc/hosts definition on SC63 and SC64

```
192.168.70.4 wtsc63f.itso.ibm.com wtsc63f
192.168.70.5 wtsc64f.itso.ibm.com wtsc64f
201.3.10.10 wasplex2.itso.ibm.com
```

6. Customize OMPROUTE.

a. OMPROUTE started task

The following is the OMPROUTE started task we used on both systems:

```
//OMPROUTE EXEC PGM=BPXBATCH,REGION=4096K,TIME=NOLIMIT,
//          PARM='PGM /usr/lpp/tcpip/sbin/omproute'
//STDOUT DD PATH='/tmp/omproutf.stdout',
//          PATHOPTS=(OWRONLY,OCREAT,OAPPEND),
//          PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//STDERR DD PATH='/tmp/omproutf.stderr',
//          PATHOPTS=(OWRONLY,OCREAT,OAPPEND),
//          PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
```

```
//STDENV DD PATH='/etc/omproutf.env',
//          PATHOPTS=(ORDONLY)
//CEEDUMP DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
```

b. OMPROUTE files

i. ENV file

The following is the env file /etc/omproutf.env we used on both systems:

```
OMPROUTE_FILE=/etc/omproutf.conf
RESOLVER_CONFIG=/etc/omproutf.resolv
_BPXX_SETIBMOPT_TRANSPORT=TCPIPF
```

ii. Configuration file

Example 3-8 and Example 3-9 on page 104 show the OMPROUTE configuration file for each stack.

Example 3-8 OMPROUTE configuration file for SC63

```
; SC63 omproutf
;
Area      Area_Number=0.0.0.0
          Stub_Area=yes
          Authentication_type=None;
OSPF_Interface IP_Address=192.168.70.4
          Subnet_mask=255.255.255.0
          Name=OSA2880LNK
          MTU=1500;
OSPF_Interface IP_Address=201.3.10.*
          Subnet_mask=255.255.255.192
          MTU=1500;
OSPF_Interface IP_Address=192.168.60.*
          Subnet_mask=255.255.255.0
          MTU=1500;
AS_Boundary_routing
  Import_RIP_Routes=YES
  Import_Direct_Routes=YES
  Import_Static_Routes=YES;
```

Example 3-9 OMPROUTE configuration file for SC64

```
; SC64 omproutf
;
Area      Area_Number=0.0.0.0
          Stub_Area=yes
          Authentication_type=None;
OSPF_Interface IP_Address=192.168.70.5
          Subnet_mask=255.255.255.0
```

```

Name=OSA2880LNK
MTU=1500;
OSPF_Interface IP_Address=201.3.10.*
Subnet_mask=255.255.255.192
MTU=1500;
OSPF_Interface IP_Address=192.168.60.*
Subnet_mask=255.255.255.0
MTU=1500;
AS_Boundary_routing
Import_RIP_Routes=YES
Import_Direct_Routes=YES
Import_Static_Routes=YES;

```

7. ADD a DNS entry to add the DVIPA address with the host name. This maps DAEMON_IPNAME and RESOLVE_IPNAME with the DVIPA.

The DNS entry looks like the following in a Windows 2000 server:

Name	TYPE	Data
wasplex2	a	201.3.10.10

8. Add the host names to the PC host file (if your PC does not use a DNS):

```

192.168.70.4 wtsc63f.itso.ibm.com
192.168.70.5 wtsc64f.itso.ibm.com
201.3.10.10 wasplex2.itso.ibm.com

```

9. You can add a Quality of Services (QoS) policy, but we did not use it in our examples.

WebSphere Application Server definitions

1. During the WebSphere installation on both systems, we defined DAEMON_IPNAME and RESOLVE_IPNAME as the following:

```

DAEMON_IPNAME=wasplex2.itso.ibm.com
RESOLVE_IPNAME=wasplex2.itso.ibm.com

```

2. We added the following to all WebSphere procedures:

```
//SYSTCPD DD DISP=SHR,DSN=TCPIPF.&SYSNAME..TCPparms(TCPDATA)
```

3. We update the current.env files (these files should be located in /WebSphere390/controlinfo/envfile/<sysplex>/<servername>/current.env) with the following:

- a. DAEMON01 and DAEMON02:

```
_BPXX_SETIBMOPT_TRANSPORT=TCPIPF
```

- b. NAMING01:

```

_BPXX_SETIBMOPT_TRANSPORT=TCPIPF
SRVIPADDR=201.3.10.10          DVIP Addr
com.ibm.ws.naming.ldap.masterurl=ldap://wtsc63f.itso.ibm.com:1389

```

Note: We added ldap://wtsc63f.itso.ibm.com, although configuring LDAP to run in multiserver mode should be sufficient.

c. NAMING02:

```
_BPXK_SETIBMOPT_TRANSPORT=TCPIPF
SRVIPADDR=201.3.10.10
com.ibm.ws.naming.ldap.masterurl=ldap://wtsc64f.itso.ibm.com:1389
```

d. SYSMGT01 and SYSMGT02:

```
_BPXK_SETIBMOPT_TRANSPORT=TCPIPF
SRVIPADDR=201.3.10.10
```

e. BBOASR2A and BBOASR2B (our IVP Appl server):

```
_BPXK_SETIBMOPT_TRANSPORT=TCPIPF
```

f. LDAP:

```
_BPXK_SETIBMOPT_TRANSPORT=TCPIPF
```

g. BBPAAWA1 and BBPAAWB1 (our Appl server):

```
_BPXK_SETIBMOPT_TRANSPORT=TCPIPF
```

Note: We edited the env files manually, because we could not add _BPXK_SETIBMOPT_TRANSPORT=TCPIPF as an environment variable through the SMEUI. We got an error message. PMR PQ54078 was opened to fix this problem, because our changes were overwritten with the settings in the SMUI when doing the next activation of the system. APAR PQ56455 provides a solution to this problem, so characters are accepted.

Checking the Sysplex Distributor configuration

Here are some commands we issued from SDSF to check the configuration:

- Example 3-10 on page 106 shows the output from the NETSTAT VIPADCFG command in all the IP stacks participating in the sysplex cluster. As you can see, only the distributing IP stack (TCPIPF on SC63) shows the VIPA DEFINE (1) and VIPA DISTRIBUTE (2) configurations with the port numbers (3) and the XCF IP addresses (4) assigned.

The output from the IP stack in SC64 shows that this stack is the primary backup (5) with rank 90.

Example 3-10 Netstat VIPADCFG for SC63 and SC64 IP stacks

```
SC63 D TCPIP,TCPIPF,N,VIPADCFG
EZZ2500I NETSTAT CS V2R10 TCPIPF 366
DYNAMIC VIPA INFORMATION:
VIPA DEFINE: 1
  IP ADDRESS      ADDRESSMASK      MOVEABLE
  -----
  201.3.10.10     255.255.255.192 IMMEDIATE
```

VIPA DISTRIBUTE: 2		
IP ADDRESS	PORT	XCF ADDRESS
-----	----	-----
201.3.10.10	00080	192.168.60.5
201.3.10.10	00080	192.168.60.4
201.3.10.10	00900	192.168.60.5
201.3.10.10	00900	192.168.60.4
201.3.10.10	01389	192.168.60.5
201.3.10.10	01389	192.168.60.4
201.3.10.10	05555	192.168.60.5
201.3.10.10 3	05555	192.168.60.4 4

```
SC64 D TCP/IP,TCPIP,N,VIPADCFG
EZZ2500I NETSTAT CS V1R2 TCP/IP 344
DYNAMIC VIPA INFORMATION:
```

VIPA BACKUP:	
IP ADDRESS	RANK
-----	----
201.3.10.10	000090 5

- Example 3-11 shows the output from the NETSTAT VDPT command for all the IP stacks participating in the sysplex cluster. The output for this command shows the Dynamic VIPA destination port table. You can see the destination IP address (1), which is the Sysplex Distributor IP address, the port numbers to which connections are being distributed (2), the destination XCF address (3), the number of applications listening on the port number selected (4), and the total number of connections that have been forwarded by the Sysplex Distributor (5).

Note that the output for the stacks in SC64 does not show any record because these stacks are not distributing workload. They are considered the target stacks by the distributing IP stack.

Example 3-11 NETSTAT VDPT command for SC63 and SC64

```
SC63 D TCP/IP,TCPIP,N,VDPT
EZZ2500I NETSTAT CS V2R10 TCP/IP 377
DYNAMIC VIPA DISTRIBUTION PORT TABLE:
```

DEST IPADDR	DPORT	DESTXCF ADDR	RDY	TOTALCONN
1	2	3	4	5
201.3.10.10	00080	192.168.60.4	001	0000000008
201.3.10.10	00080	192.168.60.5	000	0000000000
201.3.10.10	00900	192.168.60.4	001	0000000015
201.3.10.10	00900	192.168.60.5	000	0000000000
201.3.10.10	01389	192.168.60.4	001	0000000000
201.3.10.10	01389	192.168.60.5	000	0000000000
201.3.10.10	05555	192.168.60.4	001	0000000003
201.3.10.10	05555	192.168.60.5	000	0000000000

8 OF 8 RECORDS DISPLAYED

```
SC64 D TCPIP,TCPIPF,N,VDPT
EZZ2500I NETSTAT CS V1R2 TCPIPF 349
DYNAMIC VIPA DISTRIBUTION PORT TABLE:
DEST IPADDR      DPORT DESTXCF ADDR      RDY TOTALCONN  WLM
0 OF 0 RECORDS DISPLAYED
```

- Example 3-12 shows the output from the sysplex VIPADYN command. It displays the information for all the stacks participating in the sysplex at once. The command shows the MVS (system) name (1) and the actual status (2) of each stack. If the stack is defined as a backup, you can also see the rank (3) value defined for it. The display also indicates whether each stack is defined as the distributor or a destination (4) or both.

Example 3-12 sysplex VIPADYN for all stacks participating in the sysplex

```
SC63 D TCPIP,TCPIPF,sysplex,VIPADYN
EZZ8260I sysplex CS V2R10 385
VIPA DYNAMIC DISPLAY FROM TCPIPF AT SC63
IPADDR: 201.3.10.10 LINKNAME: VIPLC9030A0A
ORIGIN: VIPADEFINE
TCPNAME MVSNAME STATUS RANK ADDRESS MASK NETWORK PREFIX DIST
-----1-----2-----3-----4-----
TCPIPF SC63 ACTIVE 255.255.255.192 201.3.10.0 BOTH
TCPIPF SC64 BACKUP 090 DEST
2 OF 2 RECORDS DISPLAYED
```

3.3.3 Sysplex Distributor functionality

According to our configuration, SC63 is configured as the distributing IP stack with 201.3.10.10 as the Dynamic VIPA (DVIPA) assigned to the sysplex cluster. SC64 is configured as backup. WebSphere Application Server and HTTP server are running on both systems.

This is how Sysplex Distributor works:

1. When an IP stack on SC63 is activated, the definitions for the local XCF1 link are created dynamically because DYNAMICXCF is coded in the SC63 profile. Through this new link, SC63 recognizes the other IP stacks that belong to the same sysplex cluster and their XCF-associated links: XCF2.
2. The DVIPA assigned to the sysplex cluster, and the application ports that this DVIPA serves, are read from the VIPADISTRIBUTE statement in the profile data set. An entry is added in the home list with the distributed IP address in all the IP stacks. The home list entry on the target stacks is actually created with a message that SC63 sends to all the stacks read from the

VIPADISTRIBUTE statement. Only one stack advertises the DVIPA through the RIP or OSPF routing protocol. In this case it is the one that resides in SC63, the host in charge of load distribution.

3. SC63 monitors whether there is at least one application (HTTP and WebSphere) with a listening socket for the designated port and DVIPA. Actually, SC64 sends a message to SC63 when a server (in our case HTTP/WebSphere Application Server) is bound to either INADDR_ANY or specifically to the DVIPA (and, of course, the designated port). With that information, SC63 builds a table with the name of the application and the IP stacks that could serve any connection request for it. The table matches the application server listening port with the target XCF IP address.
4. When a client in the network requests a service from WebSphere Application Server for example, through an HTTP request, the DNS resolves the IP address for the application with the DVIPA address. This DNS could be any DNS in the IP network and does not need to register with WLM.
5. As soon as SC63 receives the connection request (TCP segment with the SYN flag), it queries WLM and/or QoS (we did not use QoS in our example) to select the best target stack for HTTP and forwards the SYN segment to the chosen target stack. In our example, it is HTTP in SC64 that best fits the request.
6. One entry is created in the connection routing table (CRT) in SC63 for this new connection with XCF2 as the target IP address. SC63 also adds the connection to its connection routing table.
7. The SC63 IP stack will forward subsequent incoming data for this connection to the correct target stack.
8. When the SC64 IP stack decides that the connection no longer exists, it informs the SC63 IP stack with a message so SC63 can remove the connection from its connection routing table.

Our implementation scenarios

In our tests, we shut down TCP/IP while HTTP and WebSphere were active. This simulates a system failure. Our implementation scenarios were as follows:

Table 3-1 Implementation scenarios

	SC63 Status	SC64 Status	Result
Scenario One	TCPIP, WebSphere and HTTP are running	TCPIP, WebSphere and HTTP are running	SD distributes the work between SC63 and SC64.
Scenario Two	TCPIP, WebSphere and HTTP are down	TCPIP, WebSphere and HTTP are running	DVIPA is switched to SC64 and SD sends the requests to SC64.

	SC63 Status	SC64 Status	Result
Scenario Three	TCPIP, WebSphere and HTTP are running	TCPIP, WebSphere and HTTP are down	DVIPA is back to SC63 and SD sends the requests to SC63.
Scenario Four	Only TCPIP is running	TCPIP, WebSphere and HTTP are running	DVIPA is on SC63 and SD sends the requests to SC64.

3.3.4 The big picture

Figure 3-5 on page 111 shows an overall picture of the flow of a client request until it is executed by WebSphere.

1. The DNS receives the client request and maps the name to an IP address.
2. Sysplex Distributor receives this request, queries WLM/QoS to select the best system available, and sends the request to it.
3. On the selected system, the HTTP queue manager assesses all incoming requests based on directives in http.conf. If the request is to be served in scalable mode, it is put on a queue for WLM to manage, otherwise the request is served by the queue manager itself. WLM processes the queued requests and passes them to a queue server.
4. In the selected HTTP server WebSphere plug-in, the local was.conf file is checked by the plug-in. If the plug-in finds a `deployedwebapp.<name>.rooturi=` statement whose value matches the received URL, it will try running the Web app locally. If it can't find a definition for the requested application in the was.conf file, it will route the request over to the WebSphere 4.0.1 runtime.
5. When the request is passed over to WebSphere, the control region receives the request and queues it to WLM. WLM starts additional server regions if required

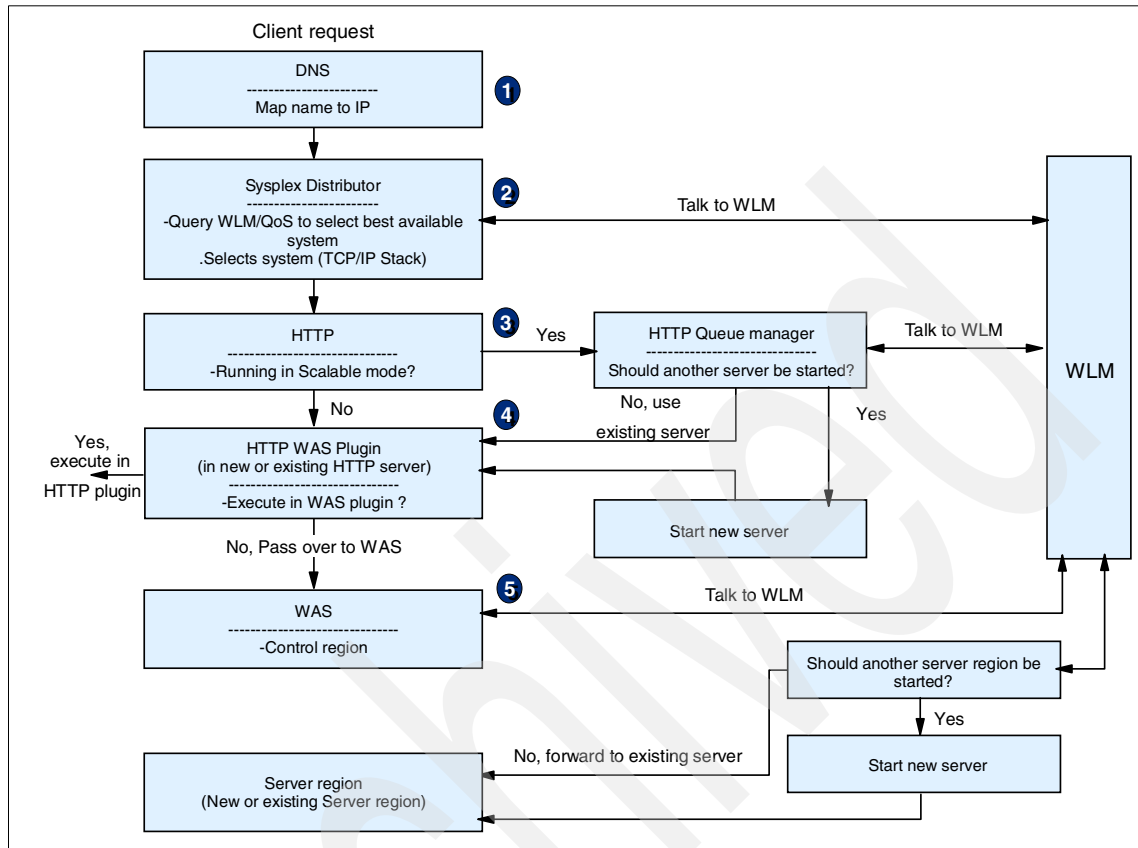


Figure 3-5 Client request flowchart

3.4 Multiple WebSphere nodes in a sysplex

APAR PQ55866 introduced a new function to WebSphere V4.0.1 that allows a test and a production WebSphere node to exist in the same sysplex. Although this configuration is not suggested because of possible hazards to the production system, customers may implement this solution.

In the following figures we show the various possibilities of how WebSphere can be set up in sysplexes. The levels of separation are:

1. Server instances
2. Systems
3. HFSs and SM repositories

4. Separate sysplexes

The required level of separation depends on the risk the installation is willing to take when running test and production applications.

- Test and production on different server instances with shared SM data and HFS on the same system, same sysplex

In this setup (Figure 3-6) the test and the production system share the system management server data and the HFS files for the application. Risk is highest.

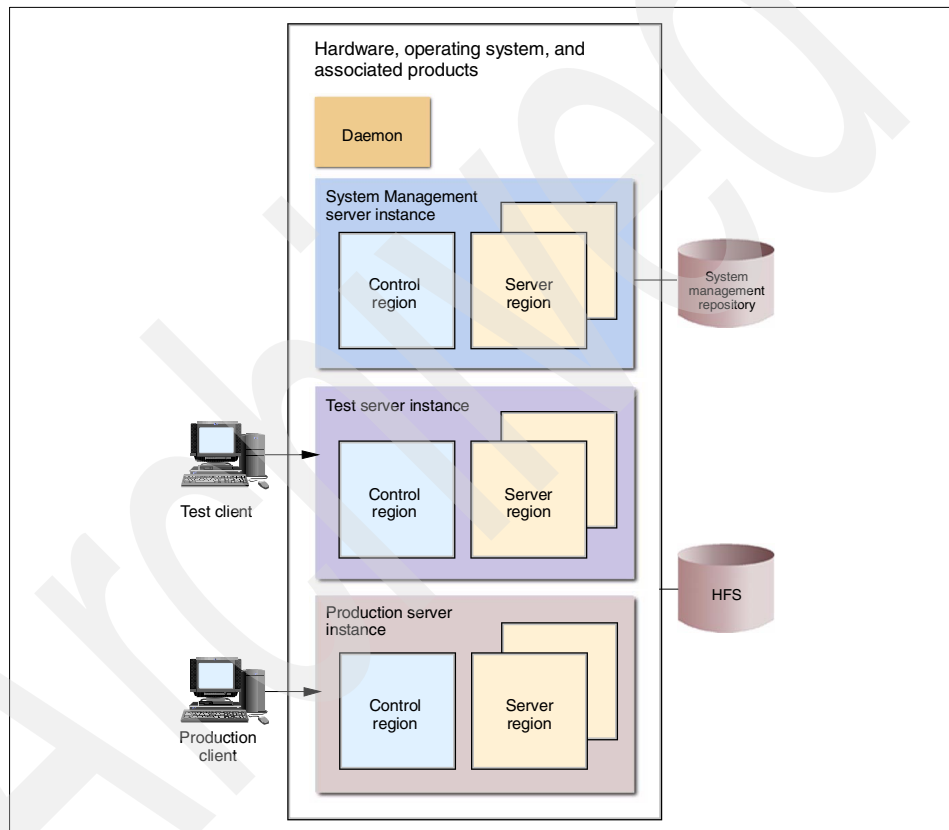


Figure 3-6 Test and production system with shared SM and HFS

- A separate WebSphere server instance for test and production on two separate systems, but in the same WebSphere node

This solution (Figure 3-7) gives somewhat better separation because test and production do not run on the same system, but still on the same WebSphere node and on the same sysplex, so the SM repository and the HFS are shared.

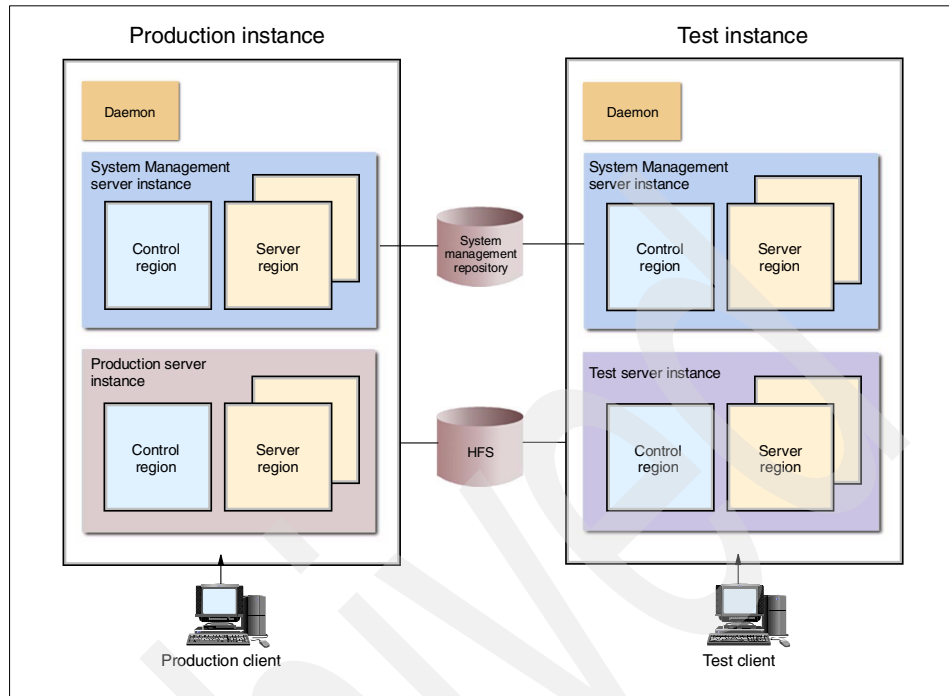


Figure 3-7 Same WebSphere but separate server instances

- Separate WebSphere node for production and test

In this environment (Figure 3-8) we separated the test and production WebSphere environments, separated the SM repositories and the HFSs, but were still running in the same sysplex.

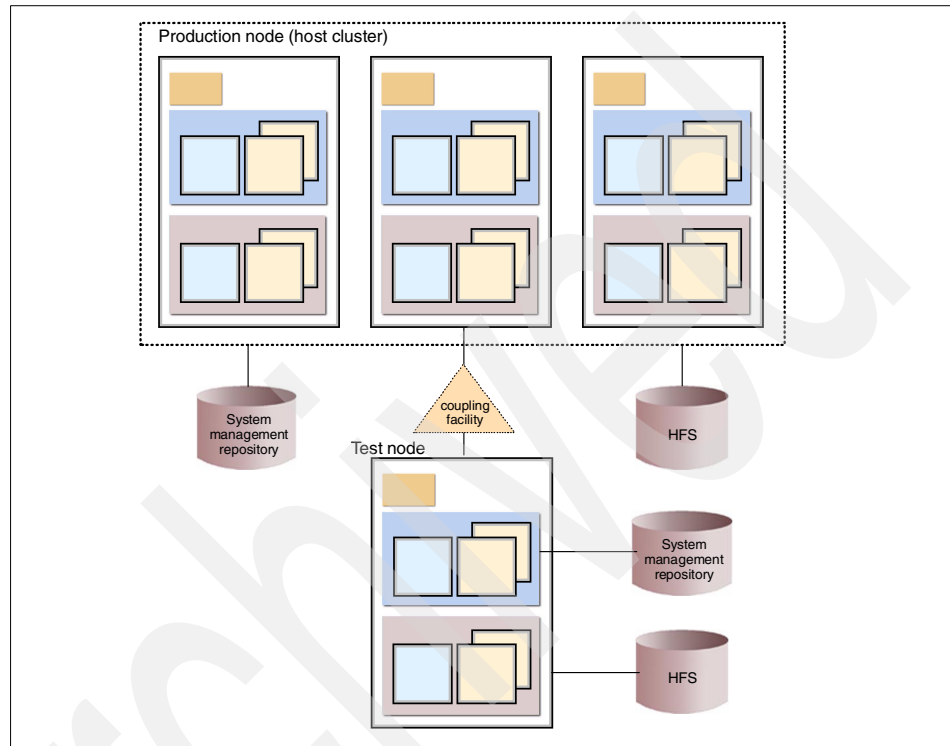


Figure 3-8 Same sysplex, but separate WebSphere nodes

► Separate sysplexes

This is the safest solution (Figure 3-9), where everything is separated.

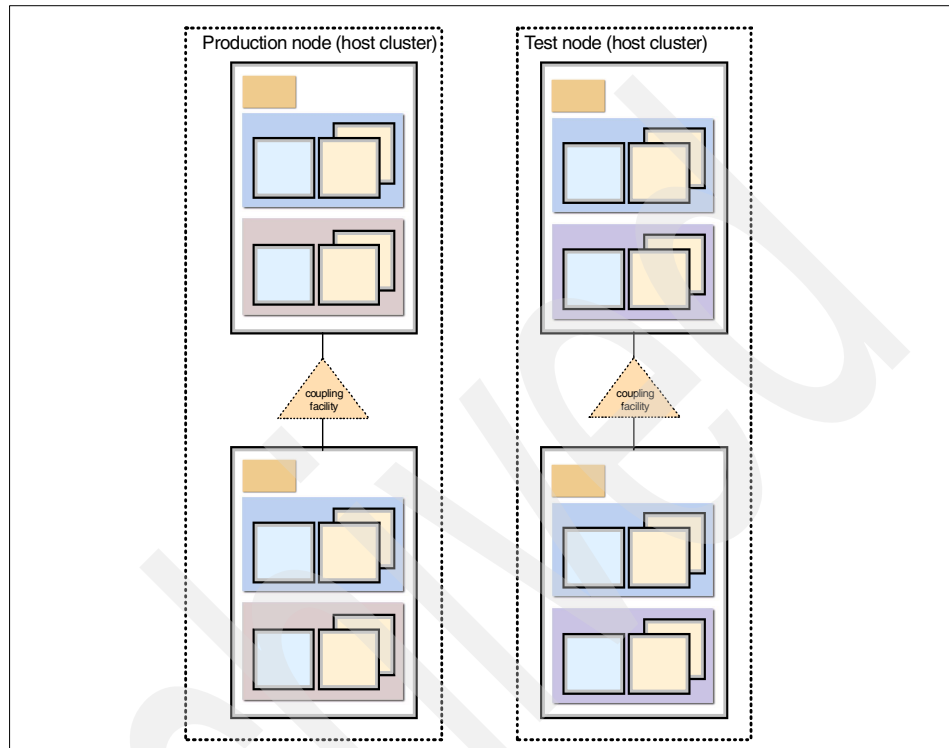


Figure 3-9 Separate sysplexes for test and production

Java-based access to DB2 for z/OS

This chapter describes the connectivity between Java programs and DB2. Its focus is on the "2-tier" solution where the Web Server, the Web Application Server, existing applications, and data reside on a zSeries server (S/390).

Application developers have two alternative APIs for accessing DB2 on OS/390 and z/OS from within a Java application:

- ▶ Java Database Connectivity (JDBC)
- ▶ Structured Query Language for Java (SQLJ)

JDBC is a component of the core Java API standard as defined by Sun, as it is an integral part of the SDK. It implements a set of methods that allow SQL statements to be *passed* to a database, and results to be accessed and manipulated. SQLJ is an ANSI standard set of extensions to the core Java classes that allow SQL to be directly *embedded* with Java applications.

The connectors based on these technologies are designed to support development and deployment on middle-tier servers and portability to z/OS without change.

The chapter includes the following:

- ▶ JDBC and SQLJ and their drivers are explained with their prerequisites, installation and configuration tasks, and a comparison between the two in sections 4.1 on page 119 to 4.4 on page 125.
- ▶ The steps to prepare and execute Java applications using JDBC and SQLJ drivers are described in sections 4.5 on page 137 and 4.6 on page 152.
- ▶ Special considerations for using JDBC with WebSphere for z/OS V4 are discussed in section 4.7 on page 155.
- ▶ Security and performance issues are discussed in sections 4.8 on page 170 and 4.9 on page 171.

The term DB2 in this chapter refers to DB2 UDB for OS/390 and z/OS, and WebSphere for z/OS V4 refers to WebSphere Application Server for z/OS and OS/390 Version 4.

4.1 JDBC and SQLJ—an introduction

DB2 supports many types of Java programs. It provides driver support for client applications and applets written in Java using Java Database Connectivity (JDBC). It also supports embedded SQL for Java (SQLJ), Java user-defined functions (UDFs), and Java stored procedures.

For general information on stored procedures, see Part 6 of *DB2 Universal Database for OS/390 and z/OS: Application Programming Guide and SQL Guide Version 7*, SC26-9933, and for general information on user-defined functions, see Part 3 of the same book. For information on defining and writing Java user-defined functions and stored procedures and setting up their environment, see Chapter 4, and for information on their preparation for execution, see Chapter 6 of *DB2 Universal Database for OS/390 and z/OS: Application Programming Guide and Reference for Java Version 7*, SC26-9932 and *e-business Cookbook for z/OS Volume III: Java Development*, SG24-5980.

4.1.1 JDBC

JDBC is a component of the core Java application programming interface (API) standard as defined by Sun, that Java applications use to access any relational database. The support of DB2 for OS/390 and z/OS for JDBC enables you to write Java applications that access local DB2 data or remote relational data on a server that supports Distributed Relational Database Architecture (DRDA). JDBC does not directly embed SQL in the Java code, but implements a set of methods (much like ODBC) that allow SQL statements to be passed to the database, and results to be accessed and manipulated.

JDBC offers a number of advantages for accessing DB2 data:

- ▶ It combines the benefit of running your applications in an OS/390 environment with the portability and ease of writing Java applications. Using the Java language, you can write an application on any platform and execute it on any platform for which the Java Development Kit (JDK) is available.
- ▶ The ability to develop an application once and execute it anywhere offers the potential benefits of reduced development, maintenance, and systems management costs, and flexibility in supporting diverse hardware and software configurations.
- ▶ The JDBC interface offers the ability to change between drivers and access a variety of databases without recoding your Java program.
- ▶ JDBC applications do not require precompiles.

4.1.2 SQLJ

SQLJ is an ANSI (and soon to be ISO) standard set of extensions to the core Java classes that allow SQL to be directly embedded within Java applications, in much the same way it can be embedded in other languages, such as COBOL and PL/1.

DB2 SQLJ allows you to create, build and run embedded SQL for Java applications, applets, servlets and stored procedures. These contain embedded SQL statements that are precompiled and bound to a DB2 database. The use of static SQL allows all authority, syntax, access strategy, and logic checking to be done at SQL compile time. This unique ability provides performance increases for applications that repeatedly use the same SQL. SQLJ also supports calling user-defined functions (UDFs).

The SQLJ standard has three components: embedded SQLJ, a translator, and a runtime environment. The translator translates SQLJ files to produce Java source files and profiles. The runtime environment performs the SQL operations in JDBC, and uses the profile to obtain details about database connections.

With SQLJ, Java applications containing SQLJ clauses written to the ANSI specification are translated by a Java precompiler to produce modified Java code and a platform-independent description of the SQLJ clauses called a profile. The profile is then customized to produce a DB2 for OS/390 and z/OS-dependent DBRM (Database Request Module), which is then bound into a package or plan and the application is executed through the JVM provided by Java for OS/390 and z/OS. The profiles are portable across platforms. Thus a profile generated by Java application-developed Oracle, compiled in NT, may be moved directly to DB2 for OS/390 and z/OS, customized and bound into a DB2 package or plan without modification of any source code. SQLJ applications have all the benefits of their JDBC counterparts and, in addition, enjoy the advantages of the static model.

For more information about SQLJ, see the SQLJ Web site at:

<http://www.sqlj.org>

and the IBM documentation in *DB2 Universal Database for OS/390 and z/OS: Application Programming Guide and SQL Guide Version 7*, SC26-9933.

4.1.3 JDBC versus SQLJ

In general, Java applications use JDBC for dynamic SQL and SQLJ for static SQL. However, because SQLJ includes JDBC, an application program can create a JDBC connection and then use that connection to execute dynamic SQL statements through JDBC and embedded static SQL statements through SQLJ.

Some of the major differences between SQLJ and JDBC are:

- ▶ SQLJ follows the static model, and JDBC follows the dynamic SQL model.
- ▶ SQLJ source programs are smaller than equivalent JDBC programs, because certain code that the programmer must include in JDBC programs is generated automatically by SQLJ.
- ▶ SQLJ execution is faster than JDBC in most cases. Since SQLJ computes the DB2 access path at compilation time, the execution of complex queries can be significantly faster than JDBC. Note that JDBC has to compute the access path at runtime.
- ▶ SQLJ can verify data types during the program preparation process (precompilation) to determine whether statements only contain valid expressions at runtime. In JDBC, no analysis or checking of the SQL statements is done until the database received them at execution time.
- ▶ In SQLJ programs, you can embed Java host expressions in SQL statements. JDBC requires a separate call statement for each bind variable and specifies the binding by position number.
- ▶ SQLJ provides the advantages of static SQL authorization checking. With SQLJ, the authorization ID under which SQL statements execute is the plan or package owner. DB2 checks table privileges at bind time. Because JDBC uses dynamic SQL, the authorization ID under which SQL statements execute is not known until runtime, so no authorization checking of table privileges can occur until runtime.
- ▶ Disadvantages of SQLJ compared to JDBC are that it is necessary to have the precompiler installed, which must be provided by the database vendor and is different for each database and each platform, and there is no integration with DB2 development (like DCLGEN). The whole development process for SQLJ is more complex because of the additional steps needed for precompilation and binding.

4.2 SQLJ/JDBC driver implementation

To allow the generic functions and methods implemented by JDBC to be translated into database-specific calls, each database vendor must deliver a JDBC driver to act as an interface.

Figure 4-1 on page 122 shows the four types of JDBC drivers. Type 1 drivers are widely replaced by Type 3 drivers. IBM delivers a Type 2 with DB2 for OS/390 and z/OS, which translates JDBC calls into calls to a DB2 language interface module. The following sections go into detail on how to install the driver and customize the environment.

Type 4 drivers are the most desired drivers, since they only require client software to communicate with the database directly and do not require any intermediate software. A Type 4 driver is all Java, similar to a Type 3 driver. The difference is that the driver uses a protocol that is understood by the DBMS rather than a network protocol like TCP/IP to communicate with the database server. In IBM's case, the protocol is Distributed Relational Database Architecture, commonly known as DRDA. Vendor products leverage IBM's DRDA protocol to achieve these Type 4 drivers. Vendors for Type 3 and 4 drivers for example are MERANT, NEON Systems, Inc., and i-net Software.

For more information about SQLJ and JDBC drivers and their vendors refer to *JDBC Links: Drivers* on the Java Sun Web site:

<http://www.java.sun.com/products/jdbc>

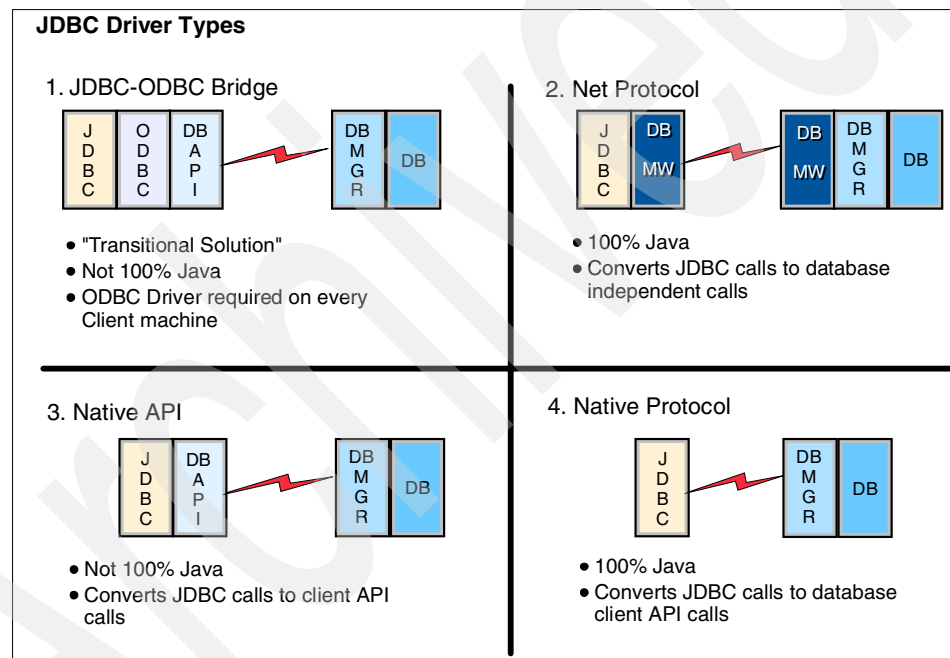


Figure 4-1 JDBC driver types

DB2 for OS/390 and z/OS provides the following implementations of JDBC and SQLJ (Type 2 drivers):

- The SQLJ/JDBC driver with JDBC 1.2 support is fully compliant with the JDBC 1.2 and SQLJ - Part 0 specification. This driver can be used with WebSphere for z/OS and it's similar in function and capabilities to the SQLJ/JDBC driver that was shipped with DB2 V5 and V6. To use this version of JDBC, you need JDK for OS/390, Version 1.1.6 or higher. The installation

will not be described in this book because there are no changes to the previous edition of this cookbook.

- The SQLJ/JDBC driver with JDBC 2.0 support is fully compliant with the JDBC 1.2 and SQLJ - Part 0 specification and includes most of the functions of the JDBC 2.0 and SQLJ Part 1 specification. To use this version of JDBC, you need the SDK for OS/390, Version 1.3 or higher. This driver includes support for SQLJ invocation of static methods as stored procedures, and JDBC 2.0 DataSource, Java Naming and Directory Interface (JNDI), and Connection Pooling, as well as for Java Transactions (JTA) when used in conjunction with WebSphere for z/OS V4 and above.

You can now create Java-stored procedures and user-defined functions that run in a Java Virtual Machine, as specified in Part 1 of the ANSI standard, Database Languages - SQL - Part 10: Object Language Bindings (SQL/OLB). These routines can contain JDBC calls, SQLJ statements, or a combination of both. You can create a JAR file that contains all the methods for your routine, install that JAR file in DB2, and then run the routine from the JAR file. Java routines have special runtime requirements, so they need to run in a WLM-established stored procedure address space that is tailored for Java routines (like the runtime environment in WebSphere Application Server Version 4).

When you connect to a data source using the SQLJ/JDBC driver with JDBC 2.0 support, your application can reference a data source using a logical name, rather than an explicit driver class name and URL. In addition, you can define or modify the data source attributes without changing the JDBC application program.

JDBC 2.0 data source support is a complete replacement for the previous JDBC driver manager support. You can use both types of support in the same application, but it is recommended that you use DataSource support to obtain connections, regardless of whether you use connection pooling or Java transactions. With DataSource support, an application uses JNDI to associate a logical name with a specific data source implementation. This DataSource object contains all of the information that is necessary to determine the correct JDBC driver and return a `java.sql.Connection` object to the specified data source.

Connection pooling is part of DataSource support. Connection pooling is a framework for caching physical data source connections, which are equivalent to DB2 threads. When JDBC reuses physical data source connections, the expensive operations that are required for the creation and subsequent closing of `java.sql.Connection` objects are minimized. Connection pooling is a built-in part of JDBC 2.0 data source support. It lets a single physical data source connection be serially reused by logical `java.sql.Connection`

instances. The application can use the logical `java.sql.Connection` object in exactly the same manner as it uses a `java.sql.Connection` object when there is no connection pooling support. Connection pooling support is completely transparent to the JDBC application.

JDBC and SQLJ global transaction support lets Enterprise Java Beans (EJB) and Java servlets that run under WebSphere for z/OS V4 or later access DB2 for OS/390 and z/OS relational data within global transactions. WebSphere for z/OS provides the environment to deploy EJBs and servlets, and RRS provides the transaction management. With global transactions you do not execute the *commit* or *rollback* methods on a `Connection` object. You let WebSphere for z/OS manage when transactions begin and end. Alternatively you can use specialized Java Transaction API (JTA) interfaces to indicate the boundaries of transactions. Although DB2 for OS/390 and z/OS does not implement the JTA specification, the methods for delimiting transaction boundaries are available.

To select a version of the SQLJ/JDBC driver, specify the associated file name in your CLASSPATH environment variable. See 4.4.4, “Setting the environment variables” on page 127 for details.

You can find the JDBC 1.2 and JDBC 2.0 specifications at the JDBC Web site:

<http://www.java.sun.com/products/jdbc>

4.3 Software prerequisites

The following are the software prerequisites for JDBC and SQLJ.

4.3.1 Prerequisites for JDBC

JDBC has an FMID of JDB7712 and is available with DB2 for OS/390 and z/OS Version 7.

JDBC for DB2 for OS/390 V7 requires the SDK for OS/390 (Version 1.3) or higher. The IBM Developer Kit for OS/390, Java™ 2 Technology Edition provides a complete Java 2 Technology Development Kit at the SDK 1.3.0 level for the S/390 platform. For additional information, refer to the Web site at:

<http://www.ibm.com/s390/java>

Refer to the *DB2 Universal Database for OS/390 and z/OS: Program Directory Version 7*, GI10-8216, for the requirements.

4.3.2 Prerequisites for SQLJ

SQLJ requires the installation of JDBC and shares the same FMID of JDB7712 as JDBC for DB2 for OS/390 and z/OS Version 7, which has a Type 2 driver available.

4.4 Installation and configuration

For the installation of the SQLJ/JDBC 1.2 driver based on JDK 1.1.8 refer to 6.2.1, “Java Database Connectivity” in the previous version of this cookbook: *Volume II: Infrastructure for Java-based Solutions*.

The README file delivered with DB2 V7 in your DB2 directory (default is /usr/lpp/db2/db2710) covers the MINIMUM steps that are required to complete the SQLJ/JDBC driver installation and configuration. Other files delivered with DB2 V7 describe and run tests for the SQLJ/JDBC driver.

You can refer to *DB2 Universal Database for OS/390 and z/OS: Application Programming Guide and Reference for Java Version 7*, SC26-9932, and *DB2 Universal Database for OS/390 and z/OS: Program Directory Version 7*, GI10-8216, for a comprehensive description of all of the installation and configuration options. The current versions may be downloaded in PDF format from the DB2 for OS/390 library, which can be accessed from the DB2 for OS/390 home page at:

<http://www.software.ibm.com/data/db2/os390>.

JDBC is a prerequisite for SQLJ as SQLJ uses the same runtime environment as JDBC. Once JDBC has been set up you can begin configuring SQLJ.

Periodically, service updates are made to the SQLJ/JDBC driver. Check the DB2 Preventive Service Planning (PSP) Bucket documentation on IBMLink for the most current information on APAR fixes and service updates. To access IBMLink on the web, go to URL:

<http://www.ibm.com/ibmlink/>

While installing SQLJ/JDBC driver 2.0, you should consider that parts of the customization/configuration have been done when you installed WebSphere for z/OS V4 as the application server uses DB2 for its control information. If you want to deploy applications using SQLJ/JDBC into WebSphere for z/OS V4, you should customize and modify the parameters and files either with the System Management User Interface (see *WebSphere Application Server V4.01 for z/OS and OS/390 System Management User Interface (SMUI)*, SA22-7838) or the System Management Application Programming Interface (see *WebSphere Application Server V4.01 for z/OS and OS/390 System Management Scripting*

API, SA22-7839) provided with the application server. This gives you the possibility to define the environment for your Sysplex or to each application separately according to the requirements in a simple way. For deployment of applications see *e-business Cookbook for z/OS Volume III: Java Development*, SG24-5980 and *WebSphere Application Server V4.01 for z/OS and OS/390 Assembling Java 2 Platform, Enterprise Edition (J2EE) Applications*, SA22-7836.

Based on our experience, to prepare your OS/390 and z/OS environment for installing the JDBC Type 2 driver support, you need to perform the steps described in the following sections of this chapter. Some definitions have to be modified due to setting up JDBC and others due to the requirements of the sample applications. You can find our sample applications (newsample01.java, newsample02.sql) in Appendix D, "Sample files" on page 377.

4.4.1 Loading the JDBC and SQLJ libraries

When you install DB2, include the steps that allocate the HFS directory structure and (using SMP/E) load the JDBC and SQLJ libraries. The jobs that perform these functions are :

- ▶ DSNISMKD
- ▶ DSNDEF2
- ▶ DSNRECV3
- ▶ DSNAPPL2
- ▶ DSNACEP2

You will find the jobs in your installation data set for DB2.

4.4.2 Database and system administration

Make sure you have the most current PTFs available. The DB2 Database Administrator should be contacted to determine the proper DB2 subsystem and location names to be used, as well as the required DB2 authorities.

Log on to TSO. Specify a maximum region size of at least 200 MB. Make sure that you have super user authority (UID=0).

4.4.3 Setting the program control extended attribute (optional)

Set the program control extended attribute for SQLJ/JDBC driver DLLs if a program that uses the driver requires a program-controlled environment, i.e., WebSphere for z/OS:

```
extattr +p /usr/lpp/db2/db2710/lib/*
```


4.4.4 Setting the environment variables

As mentioned, you have several ways to customize your environment:

1. Via SMUI for deploying Java applications into WebSphere for z/OS V4 with a graphical user interface
2. Via SMAPI for deploying Java applications into WebSphere Application Server for OS/390 and z/OS with XML files (prepared and customized) and REXX scripts
3. Via .profile to test sample applications within the Unix System Services environment and JVM
4. Via shell scripts to test sample applications without changing the test environment permanently

SMUI

The steps and options to change environment variables with the SMUI are described in detail in *WebSphere Application Server V4.01 for z/OS and OS/390 System Management User Interface (SMUI)*, SA22-7838.

SMAPI

The steps and options to customize the environment with the SMAPI are described in detail in *WebSphere Application Server V4.01 for z/OS and OS/390 System Management Scripting API*, SA22-7839.

.profile

Another way of customizing your environment is by editing your *profile* file (.profile) or the systems profile (in the /etc/ directory) in OS/390 UNIX System Services (USS). The following paragraphs explain which modifications have to be made to customize the environment for JDBC and SQLJ.

► STEPLIB

Modify STEPLIB to include the SDSNEXIT, SDSNLOAD, and SDSNLOAD2 data sets (unless the data sets are linklisted).

In the WEBSEVER started task of your SYS1.PROCLIB include:

```
//STEPLIB DD DISP=SHR,DSN=DB7A.SDSNEXIT
//          DD DISP=SHR,DSN=DB7A.SDSNLOAD2
//          DD DISP=SHR,DSN=DB7A.SDSNLOAD
```

or include the following line in your profile file:

```
export STEPLIB=DB7A.SDSNEXIT:DB7A.SDSNLOAD:DB7A.SDSNLOAD2:$STEPLIB
```

where DB7A is your DB2 V7 High Level Qualifier.

These data sets are required for executing the DB2 JDBC driver. If they are not in a STEPLIB and not in linklist, you will receive an 806-04 ABEND for module DSNHDECP.

► **PATH**

Modify PATH to include the directory that contains the shell scripts that invoke the JDBC and SQLJ program preparation and debugging functions. Since our install directory for JDBC and SQLJ is /usr/lpp/db2, the modified PATH results in:

```
export PATH=/usr/lpp/java/IBM/J1.3/bin:/usr/lpp/db2/db2710/bin:$PATH
```

► **LIBPATH and LD_LIBRARY_PATH**

The LIBPATH variable specifies the DLL search paths for Java and JDBC in the hierarchical file system. The DB2 for OS/390 and z/OS SQLJ/JDBC driver contains several dynamic load libraries (DLL). Modify LIBPATH and LD_LIBRARY_PATH to include the directory that contains these DLLs:

```
export LIBPATH=/usr:/usr/lib:/usr/lpp/db2/db2710/lib:$LIBPATH
export LD_LIBRARY_PATH=/usr/lpp/db2/db2710/lib:$LD_LIBRARY_PATH
```

► **CLASSPATH**

To use the JDBC 2.0 driver, modify CLASSPATH to include db2j2classes.zip. This file contains all the classes necessary to prepare and run JDBC and SQLJ programs with the JDBC 2.0 driver.

```
export CLASSPATH=/usr/lpp/db2/db2710/classes/db2j2classes.zip:$CLASSPATH
```

Note: If you deploy your application into WebSphere for z/OS V4, you do not need to include the name of the class file in CLASSPATH; defining the directory is sufficient (SMUI, SMAPI). If you run the application in your JDK environment of Unix System Services, you need to define the exact name of the class file. If CLASSPATH contains any of the JDBC 1.2 class files, db2sqljclasses.zip, db2sqljruntime.zip, or db2jdbcclasses.zip, delete them.

► **DB2SQLJPROPERTIES**

This environment variable specifies the fully-qualified name of the runtime properties file for the DB2 for OS/390 and z/OS SQLJ/JDBC driver. For customizing this file, see 4.4.5, “Customizing the SQLJ/JDBC runtime properties file (optional)” on page 130.

Modify the DB2SQLJPROPERTIES variable with the location of your db2sqljjdbc.properties file (all in one line). If you do not set the DB2SQLJPROPERTIES environment variable, the driver uses the default name ./db2sqljjdbc.properties. You may want to keep the customized version in a separate directory, such as /etc. In any case, because you will need this information during the WebSphere for z/OS customization process, note the name and location of this file.

```
export DB2SQLJPROPERTIES=/usr/lpp/db2/db2710/classes/db2jdbc.properties
```

Since we were testing JDBC and SQLJ with different plan names we decided to create a separate properties file for the SQLJ newsample02 application. We created a file called db2sqlj.properties in /usr/lpp/db2/db2710/classes, so we coded:

```
export DB2SQLJPROPERTIES=/usr/lpp/db2/db2710/classes/db2sqlj.properties
```

Shell script

The easiest way to customize the environment is to modify the shell script simple.sh (see Example 4-1 and a general file in Appendix D, "Sample files" on page 377) and use it for compiling and running sample Java applications. This gives you the possibility to modify all necessary parameters in one place and does not affect your other application settings.

Example 4-1 A modified simple.sh

```
#!/bin/sh
export STEPLIB=DB7A7.SDSNEXIT:DB7A7.SDSNLOAD:DB7A7.SDSNLOD2
PATH=/usr/lpp/java/IBM/J1.3/bin:/usr/lpp/db2/db2710/bin:$PATH
export PATH=/usr/lpp/java/IBM/J1.3/bin/classic:$PATH
export JAVA_HOME=/usr/lpp/java/IBM/J1.3
export LIBPATH=/usr/lpp/db2/db2710/lib:$LIBPATH
export LD_LIBRARY_PATH=/usr/lpp/db2/db2710/lib:$LD_LIBRARY
export CLASSPATH=./usr/lpp/db2/db2710/classes/db2j2classes.zip:$CLASSPATH
export DB2SQLJPROPERTIES=/usr/lpp/db2/db2710/classes/db2jdbc.properties
#
if [ "$1" = "c" ];
then
    echo "$2" " will be compiled."
    javac "$2"
else
    if [ "$1" = "r" ];
    then
        echo "$2" "will be executed."
        java "$2" "$3"
```

```

else
    echo
    echo "To compile the file : simple.sh c <java_file>"
    echo "To run the file      : simple.sh r <class_file> <db2_subsystem_name>"
    echo
fi
fi

```

You need to update the script with your specific directories and files if you have not used the default values for installation of DB2 and WebSphere for z/OS. For explanations of the variables, see “.profile” on page 127.

The script customizes the environment before you compile or run a Java application. This gives you the possibility to place the script and sample files in a directory to which several people have access (like /etc), and to run the SQLJ/JDBC samples or tests.

4.4.5 Customizing the SQLJ/JDBC runtime properties file (optional)

Modify the JDBC properties file (default: db2sqljjdbc.properties) to reflect your installation parameters.

Note: It is a good practice not to modify the db2sqljjdbc.properties file directly. Copy it to a file with a meaningful name in the same directory and then make the modification.

The parameters you can set are:

► **DB2SQLJDBRMLIB**

This variable specifies the fully qualified name of the MVS partitioned data set into which DBRMs are placed. DBRMs are generated by the creation of a JDBC profile and the customization step of the SQLJ program preparation process. The default DBRMLIB data set will be your <userid>.DBRMLIB.DATA.

```
DB2SQLJDBRMLIB=COOK4.DBRMLIB.DATA
```

If the DBRM data set does not already exist, create it. This data set requires space to hold all the SQL statements, with additional space for each host variable name and some header information. See 4.4.7, “Customizing the JDBC profile (optional)” on page 135.

Note: DB2 for z/OS V7 provides the DBRMLIB members DSNJDBC1 to DSNJDBC4 in the <HLQ_DB2_V7>.SDSNDBRM data set to set up JDBC and a sample file. Therefore, you do not need to create the members with the db2genJDBC utility, which uses the properties file to determine the location for the members created. But you do need to bind the members to the plan, which is described in “Binding a plan for an SQLJ program” on page 146. That means that, for installing JDBC, you do not need to specify the DB2SQLJDBRMLIB data set in the properties file.

If you want to run an SQLJ application, it is necessary to specify the DB2SQLJDBRMLIB data set. The db2profc customizer creates standard DB2 for z/OS DBRMs and places them into the data set defined in the DB2SQLJDBRMLIB variable.

► DB2SQLJPLANNAME

This variable specifies the name of the plan that is associated with a JDBC or an SQLJ application. The default plan is DSNJDBC, which is created during the bind process. Since we were testing JDBC and SQLJ with different plan names, we needed to point to the appropriate plan name for each application. For the JDBC sample we used:

```
DB2SQLJPLANNAME=DSNJDBC
```

For the SQLJ sample we used:

```
DB2SQLJPLANNAME=SQLBIND
```

Important: You cannot export this variable; you must code it in the properties file and point to it with the DB2SQLJPROPERTIES environmental variable.

If you do not set this variable and try to execute a plan with a different name, you will receive an SQLException (Example 4-2).

Note: For new JDBC applications it is recommended to use the default DB2SQLJPLANNAME. SQLJ applications might need their own plan name, depending on performance requirements.

Example 4-2 SQL Exception for wrong plan name

```
**** SQLException ...
Error msg: java.sql.SQLException: DB2SQLJConnection error in native method:
constructor: RRS "CREATE THREAD" failed using DB2 system:DB7A, Plan:DSNSQLJ ,
RC=0c and REASON=00f30040 SQLSTATE=FFFFF and SQLCODE=-1. SQLSTATE=FFFFF
SQLCODE=-1 java.sql.SQLException: DB2SQLJConnection error in native method:
```

```
constructor: RRS "CREATE THREAD" failed using DB2 system:DB7A, Plan:DSNSQLJ ,
RC=0c and REASON=00f30040 SQLSTATE=FFFFF and SQLCODE=-1 at
COM.ibm.db2os390.sqlj.jdbc.DB2SQLJConnection.setError(DB2SQLJConnection.java:18
13) at
COM.ibm.db2os390.sqlj.jdbc.DB2SQLJConnection.<init>(DB2SQLJConnection.java:423)
at
COM.ibm.db2os390.sqlj.jdbc.DB2SQLJConnection.<init>(DB2SQLJConnection.java:304)
at COM.ibm.db2os390.sqlj.jdbc.DB2SQLJDriver.connect(DB2SQLJDriver.java:1283) at
java.sql.DriverManager.getConnection(DriverManager.java:523) at
java.sql.DriverManager.getConnection(DriverManager.java:205) at
newsample02.main(newsample02.java:159)
```

► **DB2SQLJJDBCPROGRAM**

This variable specifies the name of a connected profile that is used by the DB2 for OS/390 and z/OS SQLJ/JDBC driver. The default connected profile name is DSNJDBC.

DB2SQLJJDBCPROGRAM=DSNJDBC

► **DB2SQLJSSID**

This variable specifies the name of the DB2 subsystem to which a JDBC or an SQLJ application connects. For example, DB7A.

DB2SQLJSSID=DB7A

If the DB2SQLJSSID variable is incorrectly set in the db2sqljjdbc.properties file, you will receive an SQL IDENTIFY error RC=08 REASON=00F30002 when you execute a program.

► **DB2SQLJMULTICONTXT**

This variable specifies whether each connection in an application is independent of other connections in the application, and each connection is a separate unit of work, with its own commit scope. The default is YES.

DB2SQLJMULTICONTXT=YES

► **DB2CURSORHOLD**

This parameter specifies the effect of a commit operation on open DB2 cursors (ResultSets). A value of YES (default) means that cursors are not destroyed when the transaction is committed. A value of NO means that cursors are destroyed when the transaction is committed.

DB2CURSORHOLD=YES

This parameter does not affect cursors in a transaction that is rolled back. All cursors are destroyed when a transaction is rolled back.

- db2.connpool.max.size

This parameter specifies the maximum number of concurrent physical connections (DB2 threads) that the driver maintains in the connection pool. The default is 100. If a logical connection is closed, and the pool is at the maximum size, the driver closes the underlying physical connection.

```
db2.connpool.max.size=200
```

- db2.connpool.idle.timeout

This parameter specifies the minimum number of seconds that an unused physical connection remains in the connection pool before the thread is closed. The default is 600. Specifying a value of zero disables idle connection timeout.

```
db2.connpool.idle.time=300
```

- db2.connpool.connect.create.timeout

This parameter specifies the maximum number of seconds that a DataSource object waits for a connection to a data source. This value is used when the loginTimeout property for the DataSource object has a value of 0. The default is 0, which means connection creation timeout is disabled.

```
db2.connpool.connect.create.timeout=300
```

► DB2SQLJATTACHTYPES

This variable specifies the attachment facility that a JDBC application program uses to connect to DB2. The value can be Call Attach Facility (CAF) or Recoverable Resource Services Attach Facility (RRSAF). You have to define resource recovery services (RRS) to enable SQLJ/JDBC multiple context support.

```
DB2SQLJATTACHTYPE=RRSAF
```

Important: WebSphere for z/OS V4 requires the use of the RRSAF of DB2.

RRSAF requires that RRS be set up. You should have installed and configured RRS when you installed WebSphere for z/OS. For details about RRS setup, refer to *z/OS MVS Programming: Resource Recovery*, SA22-7616. Details about the RRS Attach Facility are in *DB2 Universal Database for OS/390 and z/OS: Application Programming Guide and SQL Guide Version 7*, SC26-9933. Also see 4.8.2, “DB2 attachment types” on page 171.

► **DB2SQLJ_TRACE_FILENAME**

This variable enables the SQLJ/JDBC trace and specifies the names of the files to which the trace is written. The following setting enables the trace to two files named /tmp/jdbctrace and /tmp/jdbctrace.JTRACE (in text format).

```
DB2SQLJ_TRACE_FILENAME=/tmp/jdbctrace
```

► **DB2SQLJ_TRACE_BUFFERSIZE**

This parameter specifies the size of the trace buffer in virtual storage in kilobytes (rounded down to a multiple of 64 KB). The default is 256 KB.

```
DB2SQLJ_TRACE_BUFFERSIZE=1024
```

► **DB2SQLJ_TRACE_WRAP**

This parameter enables (1) or disables (0) wrapping of the SQLJ trace. The default is 1.

```
DB2SQLJ_TRACE_WRAP=0
```

You should set the parameters for diagnostic traces only if it is required or directed by an IBM representative.

Our db2jdbc.properties file is shown in Example 4-3..

Example 4-3 db2jdbc.properties - a sample for a db2sqljjdbc.properties file

```
*****
# DB2SQLJDBRMLIB=COOK4.DBRMLIB.DATA
DB2SQLJSSID=DB7A
DB2SQLJPLANNAME=DSNJDBC
DB2SQLJMULTICONTXT=YES
DB2SQLJATTACHTYPE=RRSAF
# DB2SQLJ_TRACE_FILENAME=/tmp/db2v7.err
# DB2SQLJ_TRACE_BUFFSIZE=256
# DB2SQLJ_TRACE_WRAP=1
DB2CURSORHOLD=YES
*****
```

4.4.6 Customizing the cursor properties file (optional)

JDBC result sets require a valid DB2 cursor. You can customize the file to modify the number of DB2 cursors available for JDBC and to control cursor names. The default file defines 125 cursors with hold and 125 cursors without hold. If there is no attribute specified, the cursor is assigned to nohold. To customize the cursor properties file, edit the file that you specify in the -cursors option of the db2genJDBC utility (see 4.4.7, “Customizing the JDBC profile (optional)” on page 135), and define an entry for each JDBC cursor.


```
cursor=SAMPLECURSORA:ho1d
cursor=SAMPLECURSORB:noho1d
```

We did not modify the cursor file.

4.4.7 Customizing the JDBC profile (optional)

The JDBC profile that DB2 provides is sufficient for most installations. If you need additional resources for JDBC, you can run the db2genJDBC utility to customize JDBC resources. From the /usr/lpp/db2/db2710/classes directory, run the db2genJDBC utility, for example (as one command):

```
db2genJDBC -pgmname=DSNJDBC -statements=150 -cursors=db2jdbc.cursors
-calls=5
```

where:

- pgmname is the JDBC program name (less than eight characters); the default is DSNJDBC.
- statements is the number of sections to reserve in the DBRMs for JDBC statements and prepared statements for non-result set processing; the default is 150.
- cursors is the name of the cursor properties file; the default is db2jdbc.cursors.
- call is the number of sections to reserve in the DBRMs for JDBC callable statements for non-result set processing; the default is 5.

Make sure that you have write access to /usr/lpp/db2/db2710/classes because a JDBC profile called DSNJDBC_JDBCProfile.ser will be created. This is the JDBC profile that captures all resources defined for JDBC and must be made available by way of CLASSPATH at runtime, since it will be loaded and deserialized by the JDBC driver.

For customizing the serialized profiles for SQLJ, see 4.5.2, “SQLJ application preparations” on page 140.

4.4.8 Binding the DBRMs

All DB2 programs require an application plan to allocate DB2 resources and support SQL requests made at runtime. For more information on using BIND PLAN, see Part 5 of *DB2 Universal Database for OS/390 and z/OS: Application Programming Guide and SQL Guide Version 7*, SC26-9933.

The required DB2 JDBC profile and JDBC DBRMs are now shipped with DB2 V7. It is *not* necessary to run the db2genJDBC utility unless you wish to alter the default JDBC resource limits.

In order to create the DB2 JDBC plan you need to bind the four DBRMs into packages, and include those packages in the plan. There is one DBRM for each transaction isolation level. You will find them in the <hlq_db2_v7>.SDSNDBRM data set. A sample bind job is provided in <hlq_db2_v7>.SDSNSAMP in member DSNTJJCL. Update DBRMLIB with the <hlq_db2_v7>.SDSNDBRM file, and update the DSN with your DB2 subsystem ID. The JCL shown in Example 4-4 binds the JDBC DBRMs into a plan named DSNJDBC.

Example 4-4 sample JDBC BIND job

```
//JDBC BIND JOB <JOB CARD>
//JOB LIB DD DISP=SHR,
//          DSN=DB7A7.SDSNLOAD
//BINDJDBC EXEC PGM=IKJEFT01,DYNAMNR=20
//DBRMLIB DD DISP=SHR,
//          DSN=DB7A7.SDSNDBRM
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(DB7A)
BIND PACKAGE (DSNJDBC) MEMBER(DSNJDBC1) ISOLATION(UR)
BIND PACKAGE (DSNJDBC) MEMBER(DSNJDBC2) ISOLATION(CS)
BIND PACKAGE (DSNJDBC) MEMBER(DSNJDBC3) ISOLATION(RS)
BIND PACKAGE (DSNJDBC) MEMBER(DSNJDBC4) ISOLATION(RR)
BIND PLAN(DSNJDBC)
          PKLIST(DSNJDBC.DSNJDBC1, -
                  DSNJDBC.DSNJDBC2, -
                  DSNJDBC.DSNJDBC3, -
                  DSNJDBC.DSNJDBC4)
END
/*
```

For binding SQLJ DBRMs, see 4.5.2, “SQLJ application preparations” on page 140.

4.4.9 Granting privileges

If you use your applications in the WebSphere for z/OS V4 runtime environment, then all J2EE servers must be granted EXECUTE authority on the DSNJDBC plan. If your installation allows public access to the plan, you can issue these commands using SQL Processor Using File Input (SPUFI):

```
GRANT EXECUTE ON PLAN <plan_name> TO PUBLIC;  
GRANT EXECUTE ON PACKAGE <package_name>.* TO PUBLIC;
```

In our JDBC example, <plan_name> meant DSNJDBC for the JDBC plan.

If your installation does not allow public access to the DSNJDBC plan, then you must grant EXECUTE authority to all J2EE servers. If you use DB2 secondary authorization IDs, then you can grant the authority to the groups to which the server IDs belong.

4.5 Preparing Java applications

This section explains the preparation necessary to run Java applications for JDBC, SQLJ and Stored Procedures.

4.5.1 JDBC application preparations

After you successfully completed all the configuration steps, prepare the JDBC application.

JDBC application flow

1. Imports the appropriate Java packages and classes.
2. Loads the appropriate JDBC driver (via DriverManager or DB2DataSource).
3. Connects to a database.
4. Passes SQL statements to the database.
5. Receives the results.
6. Closes the connection.

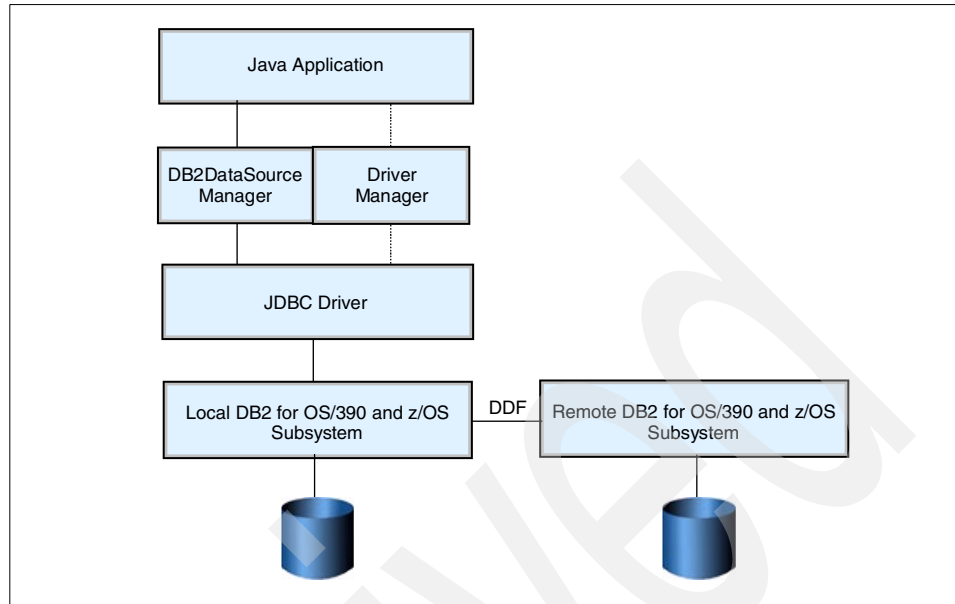


Figure 4-2 Java application flow with JDBC 2.0 for OS/390 and z/OS

Coding your program

When you begin coding your program, consider that there are two ways to create database connections. One method, the `DriverManager`, is available when you use the JDBC 1.2 driver or later. To connect via the `DriverManager` driver, refer to the README file provided with DB2 V7 in your installation directory (default: `/usr/lpp/db2/db2710` and in the subdirectory `/samples`) where you will find the description on how to compile and run the samples.

This method lacks portability because the application must identify a specific JDBC driver class name and driver URL. The driver class name and driver URL are specific to a JDBC vendor and driver implementation.

The other method, the `DB2DataSource`, is available only when you use the JDBC 2.0 driver or later. If your applications need to be portable among data sources, you should use this driver. Your application can reference a data source using a logical name, rather than an explicit driver class name and URL. In addition, you can define or modify the data source attributes without changing the JDBC application program.

The application uses the Java Naming Directory Interface (JNDI) to associate a logical name with a specific data source implementation. This data source object contains all the information that is necessary to determine the correct JDBC driver and return a `java.sql.Connection` object to the specified data source. If you use an application server, such as WebSphere for z/OS, to write your JDBC applications, the application server creates and registers `DataSource` objects for you. For a detailed description of how to develop Java applications accessing DB2 for z/OS and OS/390 data via JDBC, and how to migrate them into the environment of WebSphere for z/OS V4, see *e-business Cookbook*, Volume III: Java Development, SG24-5980.

If you want to use the Enterprise Toolkit for OS/390 with VisualAge for Java to code your program, then you need to consider that this is based on JDK 1.1.8. For a detailed description of how to install and use it with VisualAge for Java, see the previous cookbook.

Compiling your Java program

After transferring the source code into your runtime environment onto z/OS, you only need to compile and debug (if necessary) the code to be able to run the application. If you have updated your environment (the `.profile` file), then you can compile the Java code (from your work directory in OMVS) with the command:

```
javac <application_name>.java
```

To compile an application using the shell script (for `jdbcscrip.sh`, see “Shell script” on page 129), you first need to ensure that all variables have been set according to your specific configuration, especially in `DB2SQLJPROPERTIES` with the right `db2jdbc.properties` file where you set the specific `DB2SQLJSSID` (your DB2 subsystem) and the `DB2SQLJPLANNAME` (default: `DSNJDBC`). Then you can issue the following command:

```
simple.sh c <application_name>.java
```

The “c” stands for compile. After compiling the sample successfully, you should find the class file in your directory: `<application_name>.class`.

Sample application `newsample01.java`

We provided a sample JDBC application using `DB2DataSource` to connect to DB2. The source code is in Appendix D, “Sample files” on page 377.

For test purposes you should download this file into your home directory, for example:

```
/u/cook4
```

This application does the following:

1. Creates the DB2 for OS/390 JDBC DataSource.
2. Creates a connection instance.
3. Creates a statement instance.
4. Executes a query (SELECT * FROM SYSIBM.SYSROUTINES) and generates a ResultSet instance.
5. Prints column 1 (table name) to system.out.
6. Closes the ResultSet.
7. Closes the statement.
8. Closes the connection.

If you have updated your environment (the .profile file) then you can compile the sample file with this command (from your work directory in OMVS):

```
javac newsample01.java
```

In case you decided to use the shell script, make sure you updated it (simple.sh) according to your environment, especially in DB2SQLJPROPERTIES with the db2jdbc.properties file where you set the specific DB2SQLJSSID (in our case: DB7A) and the DB2SQLJPLANNAME (default: DSNJDBC). Then you can issue the following command (from your work directory in OMVS).

```
simple.sh c newsample01.java
```

The sample should be compiled without errors. If there are error messages, you have to check whether your environment is set according to the instructions and to your system settings.

Before you test the sample you must have been granted SELECT authority on SYSIBM.SYSROUTINES on the DB2 V7 subsystem. The sample program selects all rows from SYSIBM.SYSROUTINES. You can use SPUFI and issue the following command to grant the needed authorities:

```
GRANT SELECT ON TABLE SYSIBM.SYSROUTINES TO <userid>
```

4.5.2 SQLJ application preparations

After you completed all the configuration steps, prepare the SQLJ application for execution. Preparing SQLJ programs to z/OS requires more steps than preparing JDBC programs. Do the following:

1. Translate the source code to produce modified Java source code and serialized profiles.
2. Customize the serialized profiles to produce DBRMs.
3. Ensure grant authorities.
4. Bind the DBRMs into a plan.

5. Customize the db2sqljjdbc.properties file.
6. Compile the modified source code to product Java bytecode.

Figure 4-3 shows the steps of the program preparation process.

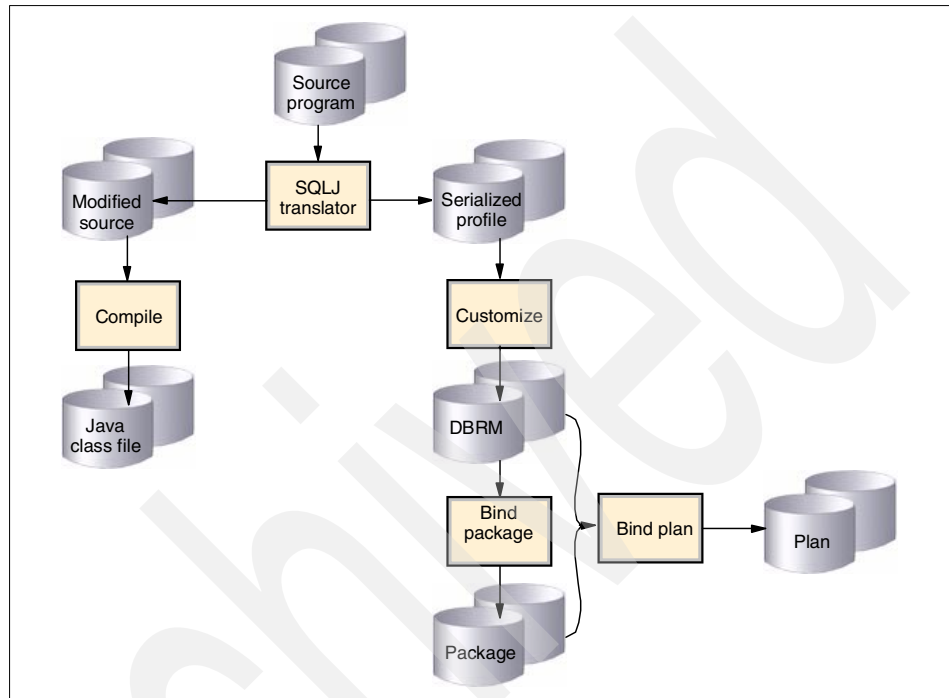


Figure 4-3 The SQLJ program preparation process

SQLJ application flow

1. Import the appropriate Java packages and classes.
2. Load the appropriate SQLJ/JDBC driver (via DriverManager or DB2DataSource).
3. Connect to a database.
4. Create SQLJ Connection Context from the JDBC connection.
5. Execute the SQL statement.
6. The result is returned in an SQL ResultSet iterator object.
7. Close the Connection Context and the connection.

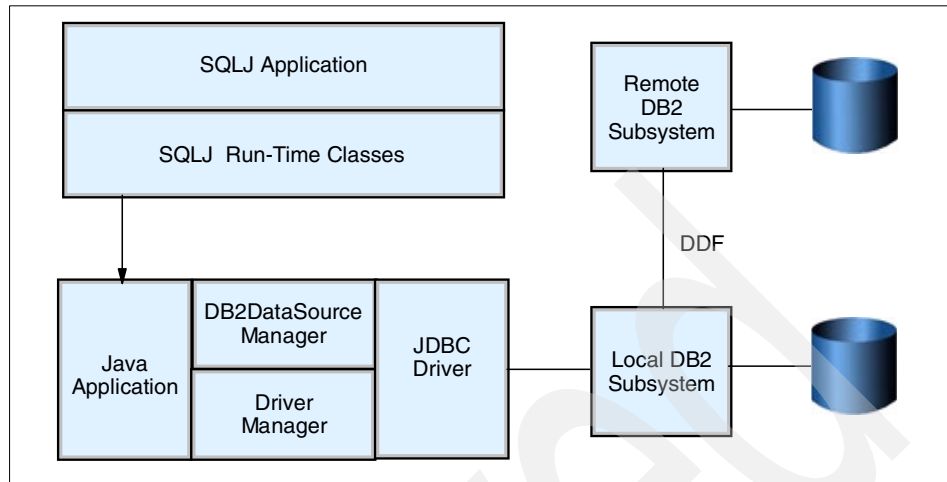


Figure 4-4 Java application flow with SQLJ

Coding your program

For a detailed description of how to develop Java applications accessing DB2 data with SQLJ, and how to migrate them into the environment of WebSphere for z/OS V4, refer to *e-business Cookbook*, Volume III: Java Development, SG24-5980.

If you want to use the Enterprise Toolkit for OS/390 with VisualAge for Java, then you need to consider that this is based on JDK 1.1.8. For a detailed description of how to install and use it with VisualAge for Java, see the previous cookbook.

Translating the SQLJ code

Once you have created the SQLJ application, you must translate the source code to produce modified Java source code and serialized profiles. Executing the `sqlj` command from the z/OS Unix System Services command line invokes the DB2 for OS/290 and z/OS SQLJ translator. The SQLJ translator runs without connecting to DB2.

```
sqlj -help -dir=directory -props=properties-file -compile=true/false
-warn=[options] application_name.sqlj
```

For a detailed description of the parameters refer to *DB2 Universal Database for OS/390 and z/OS: Application Programming Guide and Reference for Java Version 7*, SC26-9932.

For each source file, `application_name.sqlj`, the SQLJ translator produces the following files:

- ▶ The modified source program: `application_name.java`
- ▶ A serialized profile file for each connection declaration clause in the program, and one serialized profile for the default context, if it is used:
`application_name_SJProfilen.ser`, where *n* is 0 for the first serialized profile generated for the application, 1 for the second, and so on.

Ensure grant authorities

With SQLJ you can verify authorization and data types during the preparation process. You have the option to use online checking of data types when customizing serialized profiles. During the bind process, DB2 checks your SQL statements for valid table, view, and column names. Because the bind process occurs as a separate step before program execution, errors are detected and can be corrected before the program is executed.

Therefore, you have to ensure that you and future users of your application have been granted authority to access tables, views and columns according to your application before your next steps. Contact your DB2 systems administrator to get the authorities. If you have the permission to do it yourself, submit JCL jobs which grant the authority or use SPUFI to issue the GRANT command.

Compiling

Before you can customize the serialized profiles and create DBRMs, you need to compile the code. If you have updated your environment (the `.profile` file) then you can compile the Java code (from your work directory in OMVS) with the command:

```
javac <application_name>.java
```

To compile an application using the shell script (for `simple.sh`, see “Shell script” on page 129) you first need to ensure that all variables have been set according to your specific configuration, especially in `DB2SQLJPROPERTIES` with the right `db2sqlj.properties` file where you set the specific `DB2SQLJSSID` (your DB2 subsystem) and the `DB2SQLJPLANNAME` (default: `DSNJDBC`). Then you can issue the following command:

```
simple.sh c <application_name>.java
```

The “c” stands for compile. After compiling the sample successfully you should find the class file in your directory: `<application_name>.class`.

Customizing the serialized profiles for SQLJ

Before you can bind SQLJ DBRMs into a plan you need to create them. Due to performance considerations (database access optimization) it is recommended to create a new package or even a new plan for each SQLJ application. To create standard DB2 for OS/390 and z/OS DBRMs and customized profiles you need to customize each serialized profile you created by translating the SQLJ code. To customize a serialized profile, execute the following command on the z/OS UNIX System Services command line:

```
db2profc -date=ISO|USA|EUR|JIS -time=ISO|USA|EUR|JIS -sql=ALL|DB2  
-online=location-name -schema=authorization-ID -inform=YES|NO  
-validate=CUSTOMIZE|RUN -pgmname=DBRM-name serialized-profile-name
```

For a detailed description of the parameters refer to *DB2 Universal Database for OS/390 and z/OS: Application Programming Guide and Reference for Java Version 7*, SC26-9932, and *DB2 Universal Database for OS/390 and z/OS, SQL Reference, Version 7*, SC26-9944-01.

The highlighted parameters and options are:

► **-online=location-name**

Specifies that the SQLJ customizer connects to DB2 to do online checking of data types in the SQLJ program. `location-name` is the location name that corresponds to a DB2 subsystem to which the SQLJ customizer connects to do online checking. The name of the DB2 subsystem is specified in the DB2SQLJSSID keyword in the SQLJ runtime properties file.

Before you can do online checking, your SQLJ/JDBC environment must include a JDBC profile. See Chapter 6, “JDBC and SQLJ administration”, in DB2’s *Application Programming Guide and Reference for Java, Version 7*, SC26-9932-01.

Online checking is optional. However, to get the best mapping of Java data types to DB2 data types, it is recommended that you request online checking.

► **-schema=authorization-ID**

Specifies the authorization ID that the SQLJ customizer uses to qualify unqualified DB2 object names in the SQLJ program during online checking.

► **-pgmname=DBRM-name**

Specifies the common part of the names for the four DBRMs that the SQLJ customizer generates. `DBRM-name` must be seven or fewer characters in length and must conform to the rules for naming members of MVS partitioned data sets. See “Binding a plan for an SQLJ program” on page 146 for information on how to bind each of the DBRMs.

► **serialized-profile-name**

Specifies the name of the serialized profile that is to be customized. Serialized profiles are generated by the SQLJ translator and have names of the form

program-name_SJProfile*n*.ser

where program-name is the name of the SQLJ source program, without the extension .sqlj; *n* is an integer between 0 and *m*-1, where *m* is the number of serialized profiles that the SQLJ translator generated from the SQLJ source program.

For a detailed description of the parameters refer to *DB2 Universal Database for OS/390 and z/OS: Application Programming Guide and Reference for Java Version 7*, SC26-9932, and *DB2 Universal Database for OS/390 and z/OS, SQL Reference, Version 7*, SC26-9944-01.

When the SQLJ customizer runs, it modifies the contents of the serialized profile and creates DBRMs. If the customizer runs successfully, it generates an output message similar to the one shown in Example 4-5.

Example 4-5 SQLJ customizer output

```
-----
-> DB2 7.1: Begin Customization of SQLJ Profile
-----
-> SQLJ profile name is: SQLJSampleOne_SJProfile0
... (information messages) ...
-----
-> Profile <<SQLJSampleOne_SJProfile0.ser>> has been customized for DB2 for
OS/390
-> This profile must be present at runtime
-----
... (information messages) ...
-----
-> DB2 7.1: End Customization of SQLJ Profile
-----
```

Important: Customize all serialized profiles that were generated from all source files that constitute an SQLJ program.

The SQLJ customizer produces four DBRMs, one for each DB2 isolation level with which the application can run (Table 4-1 on page 146).

Table 4-1 SQLJ DBRMs and their isolation levels

DBRM name	Bind with isolation level
DBRM-name2	Cursor stability (CS)
DBRM-name3	Read stability (RS)
DBRM-name4	Repeatable read (RR)

Binding a plan for an SQLJ program

The bind process establishes a relationship between an application program and its relational data. This step is necessary before you can execute your program. Currently, DB2 allows you two basic ways of binding a program: to a package, or directly to an application plan.

When determining the maximum size of a plan, consider several physical limitations, including:

- ▶ The time required to bind the plan
- ▶ The size of the EDM pool
- ▶ Fragmentation

There are no restrictions to the number of DBRMs that can be included in a plan. However, packages provide a more flexible method for handling large numbers of DBRMs within a plan. As a general rule, it is suggested that the EDM pool be at least 10 times the size of the largest DBD or plan, whichever is greater.

The BIND PACKAGE subcommand allows you to bind DBRMs individually. It gives you the ability to test different versions of an application without extensive rebinding. Package binding is also the only method for binding applications at remote sites. Even when they are bound into packages, all programs must be designated in an application plan.

Plans can specify explicitly named DBRMs, packages, collections of packages, or a combination of these elements. The plan contains information about the designated DBRMs or packages and about the data the application program intends to use. It is stored in the DB2 catalog.

If your application uses DRDA access to distribute data, then you must use packages. During the precompilation process, the DB2 precompiler produces both modified source code and a database request module (DBRM) for each application program.

In addition to building packages and plans, the bind process does the following:

- ▶ Validates the SQL statements using the DB2 catalog. During the bind process, DB2 checks your SQL statements for valid table, view, and column

names. Because the bind process occurs as a separate step before program execution, errors are detected and can be corrected before the program is executed.

- ▶ Verifies that the process "binding the program" is authorized to perform the data accessing operations requested by your program's SQL statements. When you issue BIND, you can specify an authorization ID as the owner of the plan or package. The owner can be any of the authorization IDs of the process performing the bind. The bind process determines whether the owner of the plan or package is authorized to access the data the program requests.
- ▶ Selects the access paths needed to access the DB2 data your program wants to process. In selecting an access path, DB2 considers indexes, table sizes, and other factors. DB2 considers all indexes available to access the data and decides which ones (if any) to use when selecting a path to the data.

BIND PLAN and BIND PACKAGE can be accomplished using DB2I panels (see Figure 4-5), the DSNH CLIST, or the DSN subcommands BIND PLAN and BIND PACKAGE (see Example 4-6 on page 148).

```

. . . . .
                                BIND/REBIND/FREE                                SSID: DB7A
COMMAND ==>
Select one of the following and press ENTER:

1 BIND PLAN          (Add or replace an application plan)
2 REBIND PLAN        (Rebind existing application plan or plans)
3 FREE PLAN          (Erase application plan or plans)
4 BIND PACKAGE        (Add or replace a package)
5 REBIND PACKAGE      (Rebind existing package or packages)
6 REBIND TRIGGER PACKAGE (Rebind existing trigger package or packages)
7 FREE PACKAGE        (Erase a package or packages)

PRESS:  ENTER to process   END to exit           HELP for more information

F1=HELP   F2=SPLIT   F3=END   F4=RETURN   F5=RFIND   F6=RCHANGE
F7=UP     F8=DOWN    F9=SWAP   F10=LEFT  F11=RIGHT  F12=RETRIEVE

```

Figure 4-5 DB2 BIND Panel

With the JCL in Example 4-6 on page 148 you can bind the DBRMs which you created with the SQLJ Customizer (SQLJBIND1 to SQLJBIND4) into packages and then the packages into a plan (SQLJBIND). As you can see in the script, we used the JDBC sample bind job DSNTJJCL as a starting point and added a job card and some additional statements.

The user ID under which you bind the plan must be the same as the user ID under which you customized the profiles, because the newly allocated partitioned data set belonging to that user ID is used in the script.

Example 4-6 Sample JCL for binding a plan for an SQLJ program

```
//SQLJBIND JOB <JOB CARD>
//JOB LIB DD DISP=SHR,
//          DSN=<HLQ_DB2> .SDSNLOAD
//BINDJDBC EXEC PGM=IKJEFT01,DYNAMNBR=20
//DBRMLIB DD DISP=SHR,
//          DSN=<user-ID> .DBRMLIB.DATA
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(<DB2_subsystem>)
BIND PACKAGE (SQLJBIND) MEMBER(SQLJBIND1) ISOLATION(UR)
BIND PACKAGE (SQLJBIND) MEMBER(SQLJBIND2) ISOLATION(CS)
BIND PACKAGE (SQLJBIND) MEMBER(SQLJBIND3) ISOLATION(RS)
BIND PACKAGE (SQLJBIND) MEMBER(SQLJBIND4) ISOLATION(RR)
BIND PLAN(SQLJBIND) DYNAMICRULES(BIND) -
    PKLIST(DSNJDBC.DSNJDBC1, -
            DSNJDBC.DSNJDBC2, -
            DSNJDBC.DSNJDBC3, -
            DSNJDBC.DSNJDBC4, -
            SQLJBIND.SQLJBIND1, -
            SQLJBIND.SQLJBIND2, -
            SQLJBIND.SQLJBIND3, -
            SQLJBIND.SQLJBIND4)
END
/*
```

DB2 executes SQLJ-positioned update and delete operations dynamically and executes all other SQLJ statements statically. In addition, you might write an SQLJ program that includes JDBC methods. It is therefore recommended that you bind your SQLJ plans with option DYNAMICRULES(BIND). This option causes DB2 to use uniform authorization and object qualification rules for dynamic and static SQL statements.

With interoperability established, a Java application can contain both static and dynamic SQL. The application can execute SQLJ clauses and invoke JDBC methods. When you bind a plan for SQLJ, include JDBC packages in the PKLIST of that SQLJ plan. The default names of the JDBC packages are:

```
DSNJDBC.DSNJDBC1
DSNJDBC.DSNJDBC2
DSNJDBC.DSNJDBC3
```

DSNJDBC.DSNJDBC4

If your SQLJ application is only based on static SQL, then it is sufficient to bind the SQLJ DBRMs in your plan. Otherwise, if you use dynamic SQL in your application, include the JDBC DBRMs in your SQLJ plan to allow your SQLJ applications to interoperate with JDBC. Make sure that the JDBC profile is accessible in a directory specified in the CLASSPATH environment variable.

A complete description of the bind process can be found in *DB2 Universal Database for OS/390 and z/OS: Application Programming Guide and SQL Guide Version 7*, SC26-9933, and *DB2 Universal Database for OS/390 and z/OS: Command Reference Version 7*, SC26-9934-01.

When using SQLJ programs running under WebSphere for z/OS V4, you can define a db2jdbcsqlj.properties file (in which you specify the plan name) in three different levels:

- ▶ Sysplex
- ▶ Server region
- ▶ Server instance

If you set the DB2SQLJPROPERTIES variable in the server instance level, then it takes precedence over the setting for the same variable in the server region, and the server region setting takes precedence over the sysplex level. As a consequence, you have to bind all parts of your Java application into one or more DB2 packages and bind all the packages into one DB2 plan per server instance or region to ensure consistency and access to the right properties file.

Granting privileges

Grant EXECUTE privilege for the PACKAGES and PLANs to PUBLIC using SPUFI and the following command:

```
GRANT EXECUTE ON PLAN <plan_name> TO PUBLIC
```

In our SQLJ example <plan_name> meant SQLJBIND for the SQLJ plan. You can also include the following JCL in your BIND job, unless you have run the BIND job already:

```
//SYSIN DD *  
GRANT EXECUTE ON PLAN SQLJBIND TO PUBLIC;  
//
```

SQLJ sample application newsample02.sqlj

We provided a sample SQLJ application using DB2DataSource to connect to DB2. You find the source code in Appendix D, “Sample files” on page 377.

For test purposes you should download this file into your home directory, for example:

```
/u/cook4
```

This application does the following:

1. Creates the DB2 for OS/390 JDBC DataSource.
2. Creates a connection instance.
3. Creates an SQL Connection Context.
4. Executes a query (SELECT * FROM SYSIBM.SYSROUTINES).
5. Returns the results in an SQLJ ResultSet iterator object.
6. Closes the Connection Context.
7. Closes the connection.

To generate an executable form of the sample file you need to translate the SQLJ code to produce modified Java source code and serialized profiles:

```
sqlj newsample02.sqlj
```

The following files are created:

```
newsample02.java
```

```
newsample02.sqlj
```

```
newsample02_SJProfile0.ser
```

Before you precede you must have been granted SELECT authority on SYSIBM.SYSROUTINES on the DB2 V7 subsystem. The sample program selects all rows from SYSIBM.SYSROUTINES. You can use SPUFI and issue the following command to grant the needed authorities:

```
GRANT SELECT ON TABLE SYSIBM.SYSROUTINES TO <userid>
```

The next step lets you compile the Java source code and generate some extra files. If you have updated your environment (the .profile file) then you can compile the sample file with the command (from your work directory in OMVS):

```
javac newsample02.java
```

If you decided to use the shell script, make sure you updated it (simple.sh) according to your environment, especially in DB2SQLJPROPERTIES with the db2jdbc.properties file where you set the specific DB2SQLJSSID (in our case: DB7A) and the DB2SQLJPLANNAME (in our case: SQLBIND). Then you can issue the following commands (from your work directory in OMVS).

```
simple.sh c newsample02.java
```

When you check the content of your work directory you will find four more class files:


```

newsample02.class
newsample02Iter.class
newsample02_SJProfileKeys.class
newsample02ctx.class

```

At this point you need to customize the serialized profiles to produce DBRMs. You will produce DBRMs by issuing the following command (the command is in /usr/lpp/db2/db2710/bin, which was added to the PATH statement in a previous step):

```
db2profrc -online=DB7A -pgmname=SQLBIND newsample02_SJProfile0.ser
```

Note that the pgmname can be a maximum of 7 characters in length (we chose -pgmname=SQLBIND). The data set where the DBRMs are created is the same data set you specified in your db2sqlj.properties file, which you point to in your profile file or in the shell script simple.sh, in our case: COOK4.DBRMLIB.DATA.

The output should look like what is shown in Example 4-7.

Example 4-7 Output after customizing the profile for newsample02

```

COOK4@SC59:/u/cook4: db2profrc -online=DB7A -pgmname=SQLBIND
newsample02_SJProfile0.ser
-----
-> DB2 7.1: Begin Customization of SQLJ Profile
-----
-> SQLJ profile name is: newsample02_SJProfile0
-> holdability variable not found in class newsample02Iter. Defaulting to
holdability = false.
-> Number of sections is:      2
-> Number of HOLD   cursors is: 0
-> Number of NOHOLD cursors is: 1
-----
-> Profile <<newsample02_SJProfile0.ser>> has been customized for DB2 for
OS/390
-> This profile must be present at runtime
-----
->INFORMATIVE<-
->INFORMATIVE<- ***** IMPORTANT *****
->INFORMATIVE<- -> The following DBRMs must be bound as specified into the
->INFORMATIVE<- -> packages with the associated Transaction Isolation:
->INFORMATIVE<- -> Bind DBRM in //COOK4.DBRMLIB.DATA(SQLBIND1)
->INFORMATIVE<- -> into Package SQLBIND1 with Transaction Isolation UR
->INFORMATIVE<- -> Bind DBRM in //COOK4.DBRMLIB.DATA(SQLBIND2)
->INFORMATIVE<- -> into Package SQLBIND2 with Transaction Isolation CS
->INFORMATIVE<- -> Bind DBRM in //COOK4.DBRMLIB.DATA(SQLBIND3)
->INFORMATIVE<- -> into Package SQLBIND3 with Transaction Isolation RS
->INFORMATIVE<- -> Bind DBRM in //COOK4.DBRMLIB.DATA(SQLBIND4)

```

```

->INFORMATIVE<- -> into Package SQLBIND4 with Transaction Isolation RR
->INFORMATIVE<-
->INFORMATIVE<- -> These packages must then be bound into the plan to be used
for SQLJ applications.
->INFORMATIVE<- -> This plan must be specified at run time via the properties
file
->INFORMATIVE<- ***** IMPORTANT *****
-----
-> DB2 7.1: End Customization of SQLJ Profile
-----
COOK4@SC59:/u/cook4: >

```

The next step is to bind the DBRMs created. The sample JCL can be found in Appendix D, “Sample files” on page 377 or you may adapt the sample JCL in the DSNTJJCL member in the <HLQ_DB2_V7>.SDSNSAMP data set. We used the JCL from Example 4-6 on page 148 and adapted the following:

- ▶ The JOBLIB statement:

```
//JOBLIB DD DISP=SHR,DSN=DB7A7.SDSNLOAD
```
- ▶ The DBRMLIB statement:

```
//DBRMLIB DD DISP=SHR,DSN=COOK4.DBRMLIB.DATA
```
- ▶ The DSN SYSIN definition:

```
//SYSIN DD *
      DSN SYSTEM(DB7A)
```

We then granted EXECUTE privilege to PUBLIC for plan SQLBIND using SPUFI:

```
GRANT EXECUTE ON PLAN SQLBIND TO PUBLIC;
```

4.6 Executing Java applications

Before you run this sample, make sure both the DB2 subsystem and RRS are up and running. If you have updated your environment (the .profile file), then you can execute the Java application (from your work directory in OMVS) with the command:

```
java <java_class_file> <db2_subsystem_name>
```

To run an application using the shell script (for simple.sh see “Shell script” on page 129) you first need to ensure that all variables have been set according to your specific configuration, especially in DB2SQLJPROPERTIES with the right db2sqlj.properties file, where you set the specific DB2SQLJSSID (your DB2 subsystem) and the DB2SQLJPLANNAME (default: DSNJDBC). Then you can issue the following command:

```
simple.sh r <java_class_file> <db2_subsystem_name>
```

The "r" stands for run. If <db2_subsystem_name> is not the local site, <db2_subsystem_name> must be defined in the SYSIBM.LOCATIONS catalog table. If <db2_subsystem_name> is the local site, <db2_subsystem_name> must have been specified in field DB2 LOCATION NAME of the DISTRIBUTED DATA FACILITY (DDF) panel during installation. This value can also be seen in the output of the DB2xMSTR address space during DDF initialization.

For the examples we provided issue the command:

```
simple.sh r newsample01 db7a    for the JDBC application
or
simple.sh r newsample02 db7a    for the SQLJ application
```

You ran the samples successfully if you see output similar to Figure 4-6 on page 154 for the newsample01 application and Figure 4-7 on page 155 for the newsample02 application.

Figure 4-7 Output from `newsample02` (SQLJ)

4.7 Special considerations for WebSphere for z/OS V4

Several issues need to be considered when using JDBC and SQLJ to connect Java applications in WebSphere for z/OS with DB2 data. *WebSphere Application Server V4.01 for z/OS and OS/390 Assembling Java 2 Platform, Enterprise Edition (J2EE) Applications*, SA22-7836, and *WebSphere Application Server V4.01 for z/OS and OS/390 System Management User Interface (SMUI)*, SA22-7838 give lots of advice and comments that were very useful when we tried our samples.

We did, however, stumble over two issues that are not described in the those books and also not in *WebSphere Application Server V4.01 for z/OS and OS/390 Installation and Customization*, GA22-7834. One deals with high-level qualifiers for existing DB2 tables, the other with accessing data in previous versions of DB2, like Version 5 and Version 6.

The following sections describe the problems and their solutions in detail.

4.7.1 HLQ for DB2 tables

While deploying Java applications that access existing tables in DB2 into a J2EE Server of WebSphere for z/OS V4, you need to consider that unqualified table names in the application receive the qualification by the J2EE server they are deployed to. That means tables receive the Server ID as high-level qualifier. An attempt to access the tables will fail if the existing tables do not have the same HLQ. Example 4-8 shows the DB2SQLException thrown when the application (here a CMP EJB) bound to the BBFRNKS server attempts to access the BBFRNKS.WINES table. The existing table, which should be accessed, has another HLQ, in our case BBJRGES.

Example 4-8 Example for DB2SQLException if HLQ for DB2 table is wrong

```
Creating home instance...
com.ibm.db2.jcc.DB2SQLException: DB2JDBCClient Received Error
in Method prepare: SQLCODE==> -204 SQLSTATE ==> 42704 Error Tokens ==>
<<DB2 7.1 ANSI SQLJ-0/JDBC 1.0>> BBFRNKS.WINES
at COM.ibm.db2os390.sqlj.jdbc.DB2SQLJDBCClient.setError
(DB2SQLJDBCClient.java:1032)
```

To avoid this problem, either name your J2EE server like your DB2 HLQ or assign an alias (HLQ) to the existing table, which is named after your server ID. To do so, issue the following command in SPUFI:

```
CREATE ALIAS <server_id>.<table_name> FOR <real_table_hlq>.<table_name>
```

For our example, with BBJRGES.WINES as table name and BBFRNKS as server ID, the command is:

```
CREATE ALIAS BBFRNKS.WINES FOR BBJRGES.WINES
```

4.7.2 Accessing data with WebSphere in previous versions of DB2

WebSphere for z/OS V4 requires DB2 for OS/390 and z/OS Version 7.1 during the installation and runtime processes. Because DB2 Version 7.1 is quite recent, DB2 production applications and data often reside in Version 6 or 5. Therefore, Java applications running in the environment provided by WebSphere for z/OS V4 have access to data in Version 7 but not to the production data in the older DB2 versions. This is especially interesting if DB2 runs in data sharing mode in a Parallel Sysplex. There are several approaches to this dilemma:

- Option 1. You can migrate all members of a DB2 data sharing group to DB2 V7 prior to installing WebSphere for z/OS V4, as shown in Figure 4-8 on page 158. Because this is resource intense and requires a bigger time frame

than a WebSphere for z/OS installation project usually allows, it is only a solution for customers without time constraints.

- Option 2. You could migrate one member of the data sharing group to DB2 V7, use it as local instance for WebSphere for z/OS V4, and run a mixed level group with an older DB2 version using the same catalog in the Coupling Facility (see Figure 4-8 on page 158). This is possible since DB2 V7 can share data with an earlier DB2 version.

Note: Note that only two levels of DB2 for OS/390 can be in the same data sharing group. If data sharing, you must install DB2 for OS/390 compatibility APARs. This solution still requires a reasonable time frame for the migration process.

If you decide not to migrate to DB V7 before installing WebSphere for z/OS V4, you might face the issue that Web applications (new and migrated ones) need to access production data that resides in DB2 V6 or V5. To work around the migration to DB2 V7, you have two options (see Figure 4-8 on page 158):

- Option 3. DB2 V7 can coexist with an earlier DB2 on the same image with unique data. That means you can install DB2 V7 and WebSphere for z/OS V4 in the same image that contains DB2 V6. To access data on an earlier version of DB2, you can do a distributed call from DB2 V7.
- Option 4. Another possibility is to install DB2 V7 separate from the DB2 V6 data sharing group (Parallel Sysplex) and use the DB2 Distributed Data Facility (DDF) to access production data.

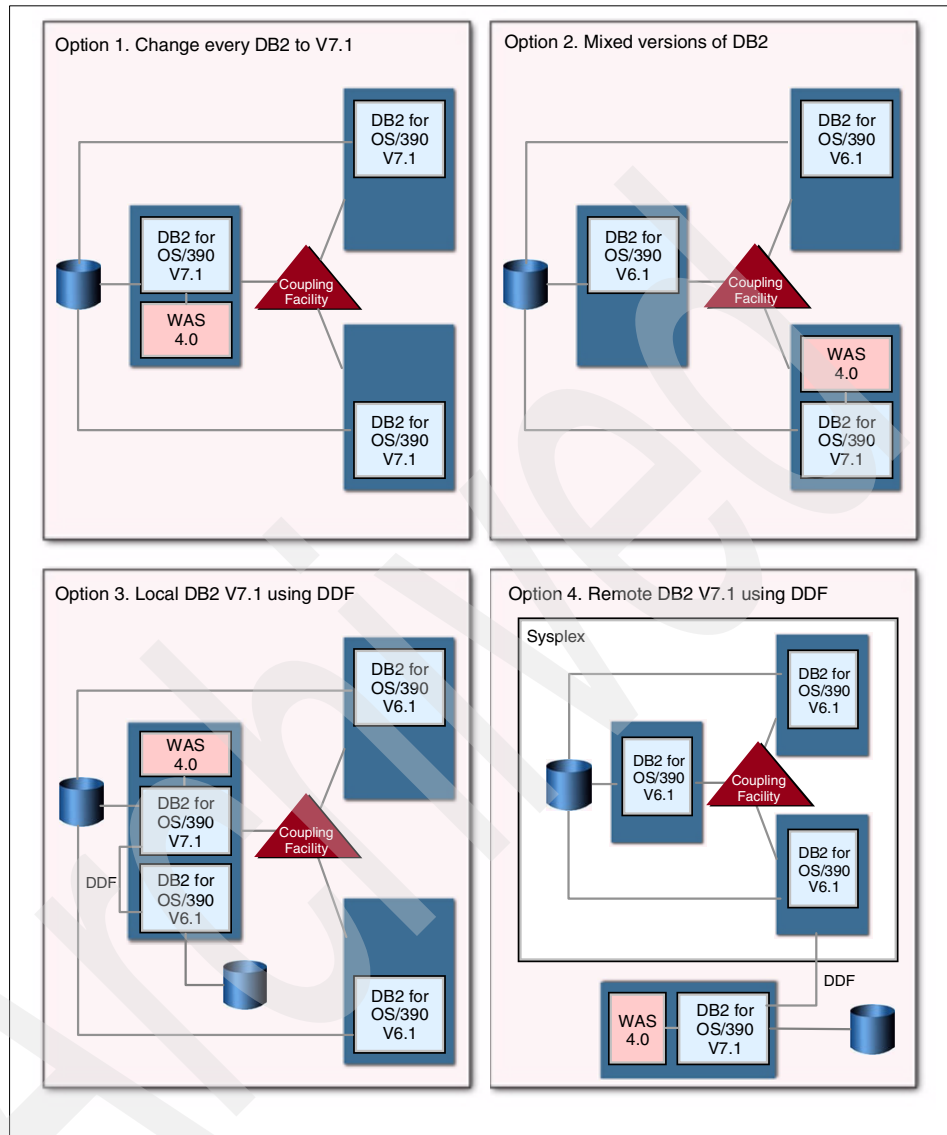


Figure 4-8 Possible configurations for accessing data in DB2 for OS/390 V6.1

Tip: Keep data sharing between multiple releases of DB2 for OS/390 to a limited time frame.

Because it is most unlikely to accomplish a quick migration to DB2 V7 if production data is involved, we decided to test the third option as the most likely one for customer situations. The Washington Systems Center group delivered documentation on what they have experienced and helped with support for our test case.

Note: The solution described is still applicable if you do not use shared HFS, data sharing, or Parallel Sysplex in general. When following our description, adjust the changes to your specific environment to be able to access DB2 V6 data via DB2 V7.

Environment description

Assume the following situation:

The ITSO runs a Parallel Sysplex named POKPLEX containing a 3-way DB2 data sharing environment (see Table 4-2) with the DB2 group name DBRAFI.

Table 4-2 Parallel Sysplex - POKPLEX - configuration

Image	DB2 V6 subsystem	DB2 V7 subsystem
POK1	DBS1	none
POK2	DBS2	none
POK3	DBS3	DB7A

POK3 was chosen as the target OS/390 image for the WebSphere for z/OS V4 environment, and therefore had to have a DB2 V7 system installed. We wanted BMP (bean managed persistent) and CMP (container managed persistent) EJBs residing in the WebSphere for z/OS V4 environment (POK3) to be backed by data in the production DB2 V6 system (DBS1, DBS2, DBS3). Additionally, we wanted to continue to allow JDBC access to the production DB2 V6 subsystems from programs running on POK1, POK2, and POK3.

TCP/IP information

The POKPLEX is part of the poughkeepsie.ibm.com domain. The TCP/IP configuration was set up to use the DNS/WLM sysplex support. The DB2 data sharing configuration is TCP/IP- and SNA-enabled for remote access. However, for the purposes of this example, we show the TCP/IP-specific information.

The DNS on POK1 is the authoritative DNS for all applications running in the POKPLEX sysplex. In particular it is responsible for resolving the names that are dynamically registered by subsystems (or application servers) such as DB2, FTP, Telnet, WebSphere for z/OS V4, etc. This requires the “real” poughkeepsie.ibm.com DNS to have CNAME entries, which causes name resolution requests to be forwarded to the DNS on POK1 where they can be properly resolved.

This configuration allows a great deal of dynamics. Members of the sysplex may be absent, yet FTPing to POKFTP will succeed if at least one member is up. Similarly, we can handle a system or subsystem failure gracefully because the DNS on POK1 (or its backup on POK2) is aware of which members are active and where. It is not important for this discussion how the generic IP name for the DB2 V6 data sharing group gets resolved, so long as it intelligently routes requests to the correct system.

Shared HFS

Because the POKPLEX is a Parallel Sysplex, shared HFS has been implemented as well. To overcome the limitation of mounting a single file system at different mount points (caused by different \$VERSION settings) in this environment, all program products which would have had file systems mounted in /usr/lpp/... are actually mounted off a single mount point in the root file system. Thus, /usr/lpp/db2/db2610 is actually a symlink to /shared/db2/db2610 and the HFS containing the DB2 V6.1.0 code (DSN610.DSNHFS) is mounted at /shared/db2/db2610. This allows us to run multiple levels of any program product throughout the sysplex with only one instance of the file system.

DB2 V6 information

To retrieve the information for each member in the data sharing group we restarted Distributed Data Facility (DDF) and received the output in the system log shown in Example 4-9).

Example 4-9 DDF configuration for DBS3

```
=DBS3 STO DDF
DSNL021I =DBS3 STOP DDF COMMAND ACCEPTED
DSNL005I =DBS3 DDF IS STOPPING
DSNL006I =DBS3 DDF STOP COMPLETE
=DBS3 STA DDF
DSNL021I =DBS3 START DDF COMMAND ACCEPTED
DSNL003I =DBS3 DDF IS STARTING
DSNL519I =DBS3 DSNLILNR TCP/IP SERVICES AVAILABLE 235
          FOR DOMAIN dbrafj.pokplex.poughkeepsie.ibm.com AND PORT 33380
DSNL519I =DBS3 DSNLIRSY TCP/IP SERVICES AVAILABLE 236
          FOR DOMAIN dbrafj.pokplex.poughkeepsie.ibm.com AND PORT 33381
DSNL004I =DBS3 DDF START COMPLETE 237
          LOCATION      DBRAFJ
```

LU	USIBMSC.DBS3
GENERICLU	USIBMSC.DBRAFJ
DOMAIN	dbrafj.pokplex.poughkeepsie.ibm.com
TCPPORT	33380
RESPORT	33383

For the other two DB2 V6 subsystems we retrieved the following information:

DBS1

LOCATION	DBRAFJ
LU	USIBMSC.DBS1
GENERICLU	USIBMSC.DBRAFJ
DOMAIN	dbrafj.pokplex.poughkeepsie.ibm.com
TCPPORT	33380
RESPORT	33381

DBS2

LOCATION	DBRAFJ
LU	USIBMSC.DBS2
GENERICLU	USIBMSC.DBRAFJ
DOMAIN	dbrafj.pokplex.poughkeepsie.ibm.com
TCPPORT	33380
RESPORT	33382

Each member must listen on the same port, in this case 33380.

The POKPLEX DNS “knows” about all the active DB2 V6 subsystems accepting incoming DDF requests by both the specific and generic names.

The essence of DB2 data sharing is that there is one copy of the data and DB2 Catalog and Directory. All DB2 subsystems which are members of the group share the data. In our environment, requests can be routed to any member that is active: DBS1, DBS2, DBS3.

To verify that you can route a request to any member, stop all other members, and run a test. To execute a sample using DBS1, for example, we stopped DBS2 and DBS3. You can permanently exclude one of the members in the group as a server by changing a value in the member's DSNZPARM. To exclude a member from DDF server processing, set the MAX REMOTE ACTIVE option of installation panel DSNTIPE to zero for that member (or job SDSNSAMP(DSNTIJUZ) macro keyword MAXDBAT). Submit the resulting DSNTIJUZ job. The effects of setting this parameter to zero are:

- DB2 does not register the member's LU name during DDF startup. Connections using the generic LU name are directed to the other members with a value greater zero.

- ▶ DB2 does not register the member to WLM for member routing.
- ▶ DB2 does not listen on the DRDA SQL port, which means that SQL TCP/IP requests are accepted only by members with values (MAXDBAT) greater than zero.

DB2 V7 information

The DB2 configuration/implementation of this subsystem is quite minimal. It exists to support the WebSphere for z/OS V4 configuration data, the stateful session bean data, the DB2 tables for the Installation Verification Program EJBs, and to be a conduit to the DB2 V6 system. Notice that the DB2 subsystem does not listen on port 33380, because the DBS3 member of the DB2 V6 data sharing group DBRAFJ uses it.

DB7A

LOCATION	DB7A
LU	USIBMSC.DB7A
GENERICLU	-NONE
DOMAIN	pok3.pokplex.poughkeepsie.ibm.com
TCPPORT	33390
RESPORT	33391

Configuring the workaround to access data in DB2 V6

The following steps describe how the system's infrastructure (POK3) needs to be modified to accomplish the JDBC access to data in the DB2 V6 system via the DB2 V7 system.

1) Set up a communication database in the DB2 V7 subsystem

Since our DB2 V6 product libraries are in the linklist, we must STEPLIB to our DB2 V7 libraries on POK3. Modify STEPLIB to include the SDSNEXIT, SDSNLOAD, and SDSNLOAD2 data sets (unless the data sets are linklisted):

```
export
STEPLIB=<hlq_db2_v7>..SDSNEXIT:<hlq_db2_v7>..SDSNLOAD:<hlq_db2_v7>..SDSNLOAD2
```

Create a JOB (Example 4-10) which inserts the server configuration in the appropriate SYSIBM tables and checks its success.

Example 4-10 Sample job DB2INSERT

```
//DB2INSERT JOB <jobcard>
//JOB LIB DD DSN=<hlq_db2_v7>..SDSNEXIT,DISP=SHR
// DD DSN=<hlq_db2_v7>..SDSNLOAD,DISP=SHR
// DD DSN=<hlq_db2_v7>..SDSNLOAD2,DISP=SHR
//STEP1 EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTPRR DD SYSOUT=*
//SYSTIN DD *
DSN SYSTEM(DB7A)
```

```

        RUN PROGRAM(DSNTEP2) PLAN(DSNTEP71) -
          LIB('<hlq_db2_v7>.RUNLIB.LOAD')
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
INSERT INTO SYSIBM.LOCATIONS (LOCATION, LINKNAME, PORT)
VALUES ('DBRAFJ','DBRAFJ','33380');
INSERT INTO SYSIBM.IPNAMES (LINKNAME, SECURITY_OUT, USERNAMES, IPADDR)
VALUES ('DBRAFJ','R',' ','dbrafj.pokplex.poughkeepsie.ibm.com');
SELECT * FROM SYSIBM.LOCATIONS;
SELECT * FROM SYSIBM.IPNAMES;
/*

```

The SYSIBM.LOCATIONS table contains a row for every accessible remote server. The row associates a LOCATION name with the TCP/IP or SNA network attributes for the remote server. The LINKNAME value is the generic LUNAME. In our configuration, it has the same name as the DB2 location name.

The SYSIBM.IPNAMES table defines the remote DRDA servers DB2 can access using TCP/IP. The SECURITY_OUT column defines the DRDA security option that is used when local DB2 SQL applications connect to any remote server associated with this TCP/IP host. There are three options. The option 'A' means "already verified". Outbound connection requests contain an authorization ID and no password. The authorization ID used for an outbound request is either the DB2 user's authorization ID (USERNAMES=' ') or a translated ID (USERNAMES=0).

The option 'R' means "RACF PassTicket". Outbound connection requests contain a userid and a RACF PassTicket. The generic LU is used to generate the application name for the RACF PassTicket. Since PassTickets are enabled in our environment, we specified 'R'. The third option, 'P', means "password". Outbound connection requests contain an authorization ID and a password. The password is obtained from the SYSIBM.USERNAMES table. The IPADDR contains the IP address or domain name of a remote TCP/IP host.

After adapting the job, submit it and check the output to see if it ran successfully (Example 4-11).

Example 4-11 Job output

```

SELECT * FROM SYSIBM.LOCATIONS;
LOCATION LINKNAME IBMREQD PORT TPN
-----
DBRAFJ DBRAFJ N 33380
DSNE610I NUMBER OF ROWS DISPLAYED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100

SELECT * FROM SYSIBM.IPNAMES;

```

```

LINKNAME SECURITY_OUT USERNAMES IBMREQD IPADDR
-----
DBRAFJ R N
dbrafj.pokplex.poughkeepsie.ibm.com
DSNE610I NUMBER OF ROWS DISPLAYED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100

```

2) Set up RACF profiles and DB2 DSNZPARM to enforce security

To set up RACF profiles to generate PassTickets, define and activate a RACF PassTicket data class (PTKTDATA). This class must contain a RACF profile for each remote DB2 subsystem to which you send requests.

- a. Activate the PTKTDATA class by issuing the following RACF commands:

```

SETROPTS CLASSACT(PTKTDATA)
SETROPTS RACLIST(PTKTDATA)

```

- b. Define profiles for the remote systems by entering the name of each remote system as it appears in the LINKNAME column of the SYSIBM.LOCATIONS table in the RACF PTKTDATA class:

```

RDEFINE PTKTDATA dbrafj SSIGNON(KEYMASKED(U111R28A39F25J28)
RALTER DBRAFJ APPLDATA('NO REPLAY PROTECTION')

```

- c. Refresh the RACF PTKTDATA definition:

```

SETROPTS RACLIST(PTKTDATA) REFRESH

```

The reason for specifying APPLDATA('NO REPLAY PROTECTION') is to deal with the fact that requests being sent to the DBRAFJ system are occurring faster than one per second. This causes non-unique passtickets to be generated by RACF on the sending system, which are rejected on the target system. The keymask data is 16 hexadecimal characters of your choice.

We set the DB2 DSNZPARM value TCPALVER=NO on each of the members of the DB2 V6 data sharing group (each DB2 has its own DSNZPARM control block, but this value should be the same for all). This was done to meet our security standards. The DB2 Installation Panel is DSNTIP5 and the parameter is TCP/IP ALREADY VERIFIED. It means that the DB2 V6 servers receive both user ID and PassTicket from any request which comes through TCP/IP. This is the default.

3) Bind JDBC-relevant DBRMs from DB2 V7 into DB2 V6 subsystem

During the installation of the DB2 V6 and DB2 V7 subsystems, JDBC was enabled for local access (see 4.4, "Installation and configuration" on page 125). In each instance, the authority to execute the DSNJDBC plan and DSNJDBC.* packages was given to PUBLIC (4.4.9, "Granting privileges" on page 136). Once the CDB is established, the DBRMs have to be bound to the DB2 V6 subsystem remotely from the DB2 V7 subsystem with the job shown in Example 4-12 on page 165.

Example 4-12 DB2 BIND job DBV6BIND for remote DBRMs

```
//DBV6BIND JOB <JOB CARD>
//JOB LIB DD DISP=SHR,
//          DSN=<hlq_db2_v7>.SDSNLOAD
//BINDJDBC EXEC PGM=IKJEFT01,DYNAMNBR=20
//DBRMLIB DD DISP=SHR,
//          DSN=<hlq_db2_v7>.SDSNDBRM
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(<db2_subsystem_id>)
BIND PACKAGE (DBRAFJ.DSNJDBC) MEMBER(DSNJDBC1) ISOLATION(UR)
BIND PACKAGE (DBRAFJ.DSNJDBC) MEMBER(DSNJDBC2) ISOLATION(CS)
BIND PACKAGE (DBRAFJ.DSNJDBC) MEMBER(DSNJDBC3) ISOLATION(RS)
BIND PACKAGE (DBRAFJ.DSNJDBC) MEMBER(DSNJDBC4) ISOLATION(RR)
BIND PLAN(DSNJDBC) ACTION(REPLACE) RETAIN -
      PKLIST(DSNJDBC.DSNJDBC1, -
              DSNJDBC.DSNJDBC2, -
              DSNJDBC.DSNJDBC3, -
              DSNJDBC.DSNJDBC4, -
              DBRAFJ.DSNJDBC.DSNJDBC1, -
              DBRAFJ.DSNJDBC.DSNJDBC2, -
              DBRAFJ.DSNJDBC.DSNJDBC3, -
              DBRAFJ.DSNJDBC.DSNJDBC4)
END
/*
```

This job looks like the one that was originally a part of the DB2 install job DSNTJJCL in <hlq_db2_v7>.SDSNSAMP, but it binds four more packages, using the same DBRMs as the original job (see also binding the SQLJ DBRMs to JDBC in Example 4-5 on page 145). They are prefixed with DBRAFJ.DSNJDBC, the location and collection names respectively for our remote DB2 V6 data sharing group. These are the packages that will be executed at the remote system DBRAFJ, because we invoke the sample from <db2_subsystem_id>. We can submit the BIND job from the local <db2_subsystem_id> and via the location name; a DB2 member of the DBRAFJ group will actually perform the bind. It can be any of the three members which are available.

The second part of the BIND job binds the DSNJDBC plan on the DB2 subsystem <db2_subsystem_id>. This enables the access to the remote packages. The four packages added to the package list again are the same ones as the original packages, but have the DB2 location name as the first qualifier.

We want to replace the original DSNJDBC plan, which we bound during the JDBC installation process. We do not want to invalidate the permissions which we granted to this plan, as the ACTION(REPLACE) will do on its own. We specified the RETAIN option, which is not the default for BIND PLAN. It allows all previous authorities to be retained.

Because the DB2 V7 DBRMs have "version" support (a timestamp in the VERSION column of SYSIBM.SYSPACKAGE), they will not overlay the DB2 V6 (or V5) packages of the same name, which are invoked by local DB2 clients. When the BIND job is successful, it also demonstrates that the CDB is configured correctly and communication is actually possible between the DB2 V7 system and one of the DB2 V6 systems.

To ensure SQLJ as well, submit the BIND job:

Example 4-13 DB2 BIND job DBV6SQLJ for remote DBRMs

```
//DBV6SQLJ JOB <JOB CARD>
//JOB LIB DD DISP=SHR,
//          DSN=<hlq_db2_v7>.SDSNLOAD
//BINDJDBC EXEC PGM=IKJEFT01,DYNAMNBR=20
//DBRMLIB DD DISP=SHR,
//          DSN=<user_id>.DBRMLIB.DATA
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(<db2_subsystem_id>)
BIND PACKAGE (DBRAFJ.DSNJDBC) MEMBER(DSNJDBC1) ISOLATION(UR)
BIND PACKAGE (DBRAFJ.DSNJDBC) MEMBER(DSNJDBC2) ISOLATION(CS)
BIND PACKAGE (DBRAFJ.DSNJDBC) MEMBER(DSNJDBC3) ISOLATION(RS)
BIND PACKAGE (DBRAFJ.DSNJDBC) MEMBER(DSNJDBC4) ISOLATION(RR)
BIND PLAN(SQLBIND) ACTION(REPLACE) RETAIN -
      PKLIST(DSNJDBC.DSNJDBC1, -
              DSNJDBC.DSNJDBC2, -
              DSNJDBC.DSNJDBC3, -
              DSNJDBC.DSNJDBC4, -
              SQLBIND.SQLBIND1, -
              SQLBIND.SQLBIND2, -
              SQLBIND.SQLBIND3, -
              SQLBIND.SQLBIND4, -
              DBRAFJ.DSNJDBC.DSNJDBC1, -
              DBRAFJ.DSNJDBC.DSNJDBC2, -
              DBRAFJ.DSNJDBC.DSNJDBC3, -
              DBRAFJ.DSNJDBC.DSNJDBC4)
END
/*
```

4) Test JDBC access to DB2 V6 via DB2 V7

The shell script `simple.sh` (“Shell script” on page 129) and the sample programs we created to test JDBC and SQLJ (“Sample application `newsample01.java`” on page 139 and “SQLJ sample application `newsample02.sqlj`” on page 149) are perfectly suitable to test the data access to DB2 V6.

When we tested JDBC we ran the script with `db7a` specified as the name of the DB2 subsystem we wanted to access:

```
simple.sh r newsample01 db7a
```

Now we just needed to change the parameter for the DB2 subsystem to `dbrafj` to access data in DB2 V6 with the same application:

```
simple.sh r newsample01 dbrafj
```

This will only work if the environment is set accordingly. You have to specify the right `db2sqljjdbc` properties file in the shell script (in our case `db2jdbc.properties` for JDBC applications or `db2sqlj.properties` for SQLJ applications like `newsample02`). This properties file should contain the right `DB2SQLJSSID` (the DB2 subsystem ID we used) and the right `DB2SQLJPLANNAME` (default name: `DSNJDBC`, or `SQLBIND` for SQLJ applications like `newsample02`). Also make sure that the users you use to run the samples are granted `SELECT` authorities on the `SYSIBM.SYSROUTINES`.

```
GRANT SELECT ON TABLE SYSIBM.SYSROUTINES TO <userid>
```

Test the SQLJ connection via JDBC to access data on DB2 V6 with the following command:

```
simple.sh r newsample02 dbrafj
```

To compare outputs you can issue an `SPUFI` command to select the names from the `SYSIBM.SYSROUTINES` table within DB2 V6. The results should be the same.

So far you have configured and tested the immediate environment for running those sample applications. To ensure access to DB2 V6 data from within WebSphere for z/OS V4, you should follow the next steps as well.

5) Configure data sources in WebSphere for z/OS V4 to use DB2 V6

Remember the Policy IVP application for WebSphere for z/OS V4. The deployment process involved defining a data source and a data source instance for each of the DB2 systems to which you wanted to connect. Then you had to resolve the resource reference to a DB2 data source.

Now we want the CMP bean to reference data residing in the DB2 V6 system (dbrafj) and the BMP bean to reference data in the DB2 V7 system (db7a). Figure 4-9 illustrates the configuration of the data source for db7team which represents db7a and its data source instance with a corresponding entry for dbrafj.

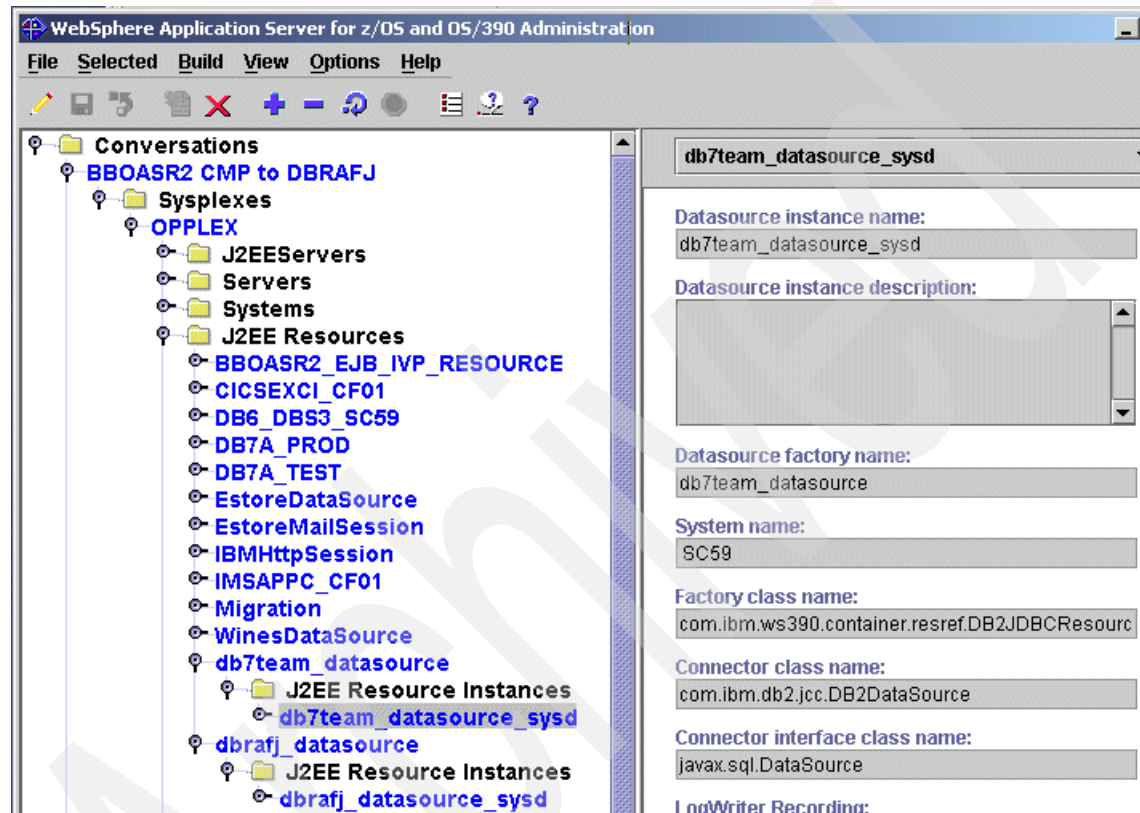


Figure 4-9 Definition of DB2 data source and its instance (here db7team instead of db7a)

These data sources and their instances were created once and can be used by any or all EJBs that need data from one or the other DB2 system. Figure 4-10 on page 169 shows how the CMP data source reference was associated with the dbrafj_datasource, which we defined previously.

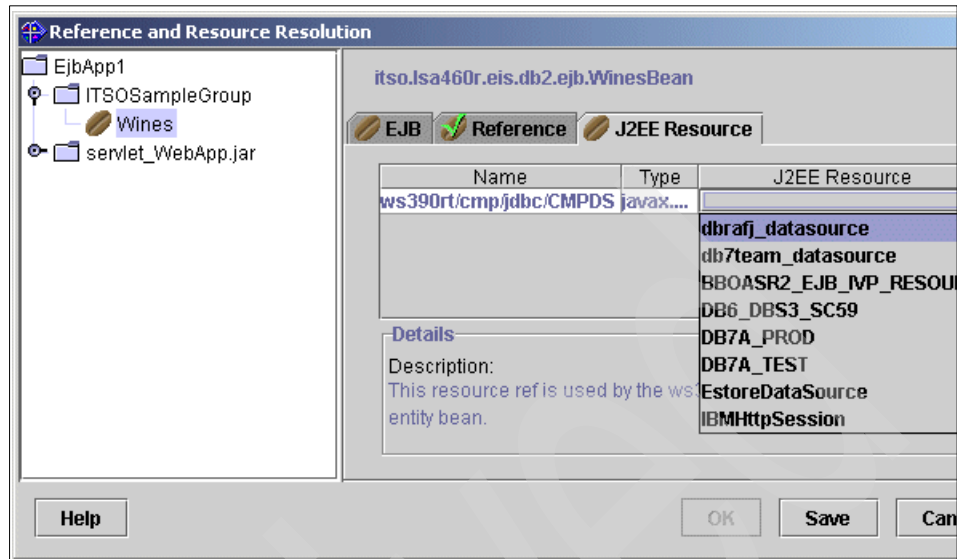


Figure 4-10 Assign data source to a bean (here dbrafj_datasource to the CMP WinesBean)

Once the resource reference is resolved and the deployment process is complete, you can see this bean is indeed backed by data in the DB2 V6 data sharing group.

Now it is a simple matter of creating the table (i.e., BBO.POLICYDO) referenced in both dbrafj and db7a locations, grant permission for the userid associated with the server to Select, Update, Insert, and Delete from the BBO.POLICYDO table and run the IVP script. The BMP EJB test will update the table in db7a, and the CMP EJB test will update the table in dbrafj.

If you run the script from the shell or use the browser, then you should get the output shown in Example 4-14.

Example 4-14 Parts of the output after running the IVP script

```
***** bmp bean will be run!
...
bmp IVP has completed successfully
***** cmp bean will be run!
...
cmp IVP has completed successfully
```

To verify that this actually worked, log on to TSO on your system with a userid having the authority to select from BBO.POLICYDO and run an SPUFI script to select everything in the table at each location; you will see the output in Example 4-15 on page 170.

Example 4-15 Select statements and their output to verify the configuration

```
-----+-----+-----+-----+-----+-----+-----+
select * from db7a.bbo.policydo
-----+-----+-----+-----+-----+-----+-----+
                PAMOUNT    PNUMBER                PPREMIUM
-----+-----+-----+-----+-----+-----+-----+
+0.1100000000000000E+02    1122    +0.1700000000000000E+03
+0.3300000000000000E+02    3344    +0.3800000000000000E+03
DSNE610I NUMBER OF ROWS DISPLAYED IS 2
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
-----+-----+-----+-----+-----+-----+-----+
select * from dbrafj.bbo.policydo
-----+-----+-----+-----+-----+-----+-----+
                PAMOUNT    PNUMBER                PPREMIUM
-----+-----+-----+-----+-----+-----+-----+
+0.2200000000000000E+02    2468    +0.1500000000000000E+03
+0.1980000000000000E+03    2244    +0.3000000000000000E+03
DSNE610I NUMBER OF ROWS DISPLAYED IS 2
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
```

If these values correspond to the ones you received when you ran the IVP, then you can be sure that your configurations were successful. That means you can access DB2 V6 data with your Java application running in WebSphere for z/OS V4 via the DB2 V7 subsystem.

4.8 JDBC and SQLJ security issues

This section (based on *DB2 Universal Database for OS/390 and z/OS: Application Programming Guide and Reference for Java Version 7*, SC26-9932) describes JDBC and SQLJ security issues to be considered. It explains how authorization IDs are determined and the use of attachment facilities.

4.8.1 How authorization IDs are established

With the JDBC 2.0 driver, the method that DB2 uses to determine the SQL Authorization ID to use for a connection depends on whether you provide user ID and password values for the connection. If you provide a user ID and password, the JDBC driver passes these values to DB2 for validation and uses these values for the connection. If you do not provide a user ID and password, the JDBC driver uses the external security environment that is associated with the thread to establish the DB2 authorization ID.

4.8.2 DB2 attachment types

The security environment (the RACF ACEE) that DB2 uses to establish the DB2 authorization IDs is dependent on which DB2 attachment type you use. JDBC and SQLJ use a DB2 attachment facility to communicate with DB2. They use the RRS attachment facility (RRSAF) or the CICS attachment facility (CAF). All attachment types support multithreading, that is, multiple, concurrent threads (TCBs) that execute within a single process (address space). In a multithreading environment, each process and thread can have its own unique security environment. The DB2 attachment facility that you select determines which security environment DB2 uses to verify the DB2 authorization IDs.

Attention: See DB2SQLJATTACHTYPES in 4.4.5, “Customizing the SQLJ/JDBC runtime properties file (optional)” on page 130 for further consideration and modification.

See “Appendix B. Special considerations for CICS applications” on page 113 in *DB2 Universal Database for OS/390 and z/OS: Application Programming Guide and Reference for Java Version 7*, SC26-9932 for information on using the CICS attachment facility.

The DB2 RRSAF supports multithreading, and applications can run under multiple authorization IDs. If you use the RRSAF, DB2 uses a task-level security environment, if present, to establish the authorization IDs.

4.9 SQLJ and JDBC performance considerations

Based on the redbook *DB2 for OS/390 and z/OS Powering the World's e-business Solutions*, SG24-6257 we make a series of recommendations. They can have a dramatic effect on the performance of your SQLJ applications. For performance issues related to the Java code, refer to recent articles at the developerworks Internet pages:

www.developerworks.ibm.com

Recent performance studies have highlighted the fact that these recommendations should not be regarded as implementation options but as actions you must take in order to get good SQLJ application performance. A good source of general considerations on Java performance is *IBM Systems Journal Vol.39, No.1, 2000, Java Performance*, G321-0137. The points we make are not listed in order of importance but should be implemented as a whole.

Use the correct level of the JDK

Make sure that your JDK level is at least 1.3. With JDK 1.3 you can exploit the hardware support for floating point calculations provided by the G5, G6, and newer processors.

BIND options

Specify DYNAMICRULES(BIND) for dynamic SQL to make sure that the table access privileges of the binder are implicitly used during execution of the package. Otherwise, you will have to grant authorization over the underlying DB2 objects to the authorization ID executing the plan. Dynamic SQL includes JDBC and cursor-controlled updates and deletes in SQLJ. Use the QUALIFIER keyword of the BIND command to provide qualification for unqualified table or view names referenced in JDBC and SQLJ.

Online checking with db2prof

For string data types in Java there is no concept of length. The associated SQL column data types are CHAR and VARCHAR. In order to have the predicates used for index matching, the definition in the DBRM for SQLJ (static SQL) must match the definition in the DB2 catalog in terms of data type and length. To achieve this, you must customize the SQLJ serialized profile using **db2prof** and specify the online checker option `-online = <DB2 location name>`:

```
db2prof ... -online=<db2_location_name> ...
```

Use the correct serialized profile

To achieve true static SQL execution for SQLJ applications, customize the SQLJ serialized profile using **db2prof**, and the output customized profile must be available and accessible in the runtime PATH.

Activate dynamic SQL statement caching

For relatively simple SQL requests, the processing cost of preparing dynamic SQL statements can be very significant. To dramatically reduce the processing overhead and get reasonably close to the performance of static SQL, DB2 provides for dynamic SQL statement caching. Dynamic statement caching avoids the full cost of preparing an SQL request. Dynamic statement caching is enabled by specifying CACHEDYN=YES on the DSN7SPRM macro in DSNZPARM. When the prepared statement is found in the prepared statement cache, it is possible to realize significant savings. A typical hit ratio is between 70 percent and 90 percent. For optimal performance, we strongly recommend that you use SQLJ for database access. When using JDBC or controlled updates or deletes in SQLJ, we recommend that you turn on dynamic statement caching.

WebSphere MQ

In this chapter, we discuss the MQSeries classes for Java as the prime connector for Java-based Web applications. We also briefly introduce you to some of the MQSeries features to further connect to backend systems on z/OS. In most cases, the purpose of your new Java application will be to run a CICS or IMS transaction or do something else with the messages you send from the Java application. The features are:

- ▶ MQSeries-CICS bridge
- ▶ MQSeries-IMS bridge
- ▶ MQSeries messages to other applications

We will also discuss the MQ classes for Java Messaging Service (JMS). JMS is a specification from Sun Microsystems that provides a standard way for Java programs to interface with enterprise messaging systems such as MQSeries.

MQSeries was renamed to WebSphere MQ not long ago. We will use MQSeries when referring to WebSphere MQ in this chapter.

5.1 MQSeries for z/OS

MQSeries for z/OS runs on a System/390 processor that is capable of running z/OS, to provide messaging and queuing support for the following types of applications:

- ▶ CICS
- ▶ IMS
- ▶ TSO and batch

MQSeries products enable applications to use message queuing to participate in message-driven processing. With message-driven processing, applications can communicate with each other on the same or different platforms, by using the appropriate message queuing software products. For example, z/OS and OS/400 applications can communicate through MQSeries for z/OS and MQSeries for iSeries, respectively. With MQSeries products, all applications use the same kind of messages and communication protocols are hidden from the applications. A prime requirement for a message delivery system is that it must be reliable.

Many functions are built into MQSeries for z/OS to ensure that:

- ▶ Messages are not lost despite system failures
- ▶ Messages are not delivered more than once
- ▶ Messages are not accessed by, or delivered to, unauthorized persons or applications

The features provided by MQSeries for z/OS to support this, and which you must plan for, include:

- ▶ Logging (including options for dual logging and archiving)
- ▶ Automatic recovery from transaction, system, and storage media failures (for which suitable backups are needed)
- ▶ Restart from backup files
- ▶ Access to security management facilities

Also, plan for the administration of MQSeries for z/OS in your operation, make decisions regarding the performance aspects of the product, and consider, if necessary, possible migration of both your system and applications from other products.

5.1.1 Preparing your applications for the use of MQSeries

MQSeries for z/OS brings the Message Queue Interface (MQI) to your applications. This interface allows you to modify existing applications and to write new applications. The MQI removes much of the need to understand any network or communication systems that you use. Thus, you can expect to complete applications more speedily than before. However, you must plan to take advantage of the MQI by planning its use in your applications, and by understanding the ways in which it assists you.

You can find more information on how to use MQSeries for z/OS in your applications in *MQSeries Application Programming Guide*, SC33-0807.

5.1.2 Interfacing with CICS, IMS, or batch

With MQSeries for z/OS you can create applications in these environments:

- ▶ A CICS transaction environment
- ▶ An IMS transaction environment
- ▶ A z/OS Batch and Time Sharing Option (TSO) environment, optionally using RRS applications (or transactions) connected to MQSeries by means of an adapter

Adapters for each of these environments are included in MQSeries for z/OS.

The MQSeries-CICS bridge

The MQSeries-CICS bridge enables an application not running in a CICS environment to run a CICS program or transaction on CICS and receive a response back. This non-CICS application can be run from any environment that has access to an MQSeries network that encompasses MQSeries for z/OS.

This CICS program can be invoked using the EXEC CICS LINK command. It must conform to the DPL subset of the CICS API, that is, it must not use CICS terminal or syncpoint facilities.

The CICS transaction is designed to run on a 3270 terminal. This transaction can use BMS or TC commands. It can be conversational or part of a pseudo conversation. It is permitted to issue syncpoints.

The MQSeries-IMS bridge

The MQSeries-IMS bridge is a component of MQSeries for z/OS that allows direct access from MQSeries applications to applications on your IMS system. It enables implicit MQSeries API support. This means that you can re-engineer legacy applications that were controlled by 3270-connected terminals to be controlled by MQSeries messages, without having to rewrite, recompile, or relink them. The bridge is an IMS Open Transaction Manager Access (OTMA) client.

5.1.3 Planning to use MQSeries in a network

MQSeries for z/OS uses the Distributed Queue Management (DQM) facility to exchange messages between MQSeries platforms.

With MQSeries for z/OS, access to remote queues can be with:

- ▶ SNA LU 6.2
- ▶ TCP
- ▶ CICS (using ISC)

Which of these alternative methods of connection is best for your installation depends on a number of factors, including:

- ▶ Whether you require CICS for use by your applications
- ▶ What other MQSeries products you wish to connect to, and the methods of connection that they support
- ▶ If alternatives are possible, you need to decide which of these protocols you are going to use before you install MQSeries for z/OS. Having decided on which form of connection to use, you need to define the channels that are to be used in the exchange of messages. You can simplify your definitions for distributed queuing by using MQSeries clusters (not available if you are using CICS ISC for distributed queuing).

For further information about distributed queuing, refer to *MQSeries Intercommunication*, SC33-1872 and *MQSeries Queue Manager Clusters*, SC33-5349. Refer to *CICS Transaction Server for OS/390*, SC33-1695, for details on managing remote CICS links.

5.2 MQSeries classes

In this section we discuss the MQSeries classes for Java, which is the prime connector for any Java program (applets, servlets, applications, beans) that needs to exchange messages with any other program using MQSeries. We also discuss the MQSeries classes for Java Messaging Service (JMS). These classes allow applications to send MQSeries messages to either existing MQSeries or new JMS applications.

5.2.1 MQSeries classes for Java

The MQSeries classes for Java enable you to write MQSeries applications using the Java programming language. These applications communicate directly with MQSeries queue managers to provide a high-productivity development option.

MQSeries provides an excellent infrastructure for access to enterprise applications. A service request can be queued and processed when possible, thus allowing a timely response to be sent to the end-user regardless of system loading. By placing this queue “close” to the user in network terms, the timeliness of the response is not impacted by network loading.

The MQSeries classes for Java also enable application developers to exploit the power of the Java programming language to create applications which can run on any platform that supports both the Java runtime environment and the MQSeries classes for Java. These factors combine to dramatically reduce the development time for multi-platform MQSeries applications.

The MQSeries classes for Java on z/OS are compatible with the Java Virtual Machine available in both the UNIX System Services and CICS Transaction Server environments. In addition, the MQSeries classes for Java on z/OS are compatible with VisualAge for Java, Enterprise Edition for z/OS.

The MQSeries classes for Java on z/OS provide the same programming interface as the MQSeries classes for Java supplied with MQSeries Version 5 platforms, with the exception that they do not support connecting to a queue manager as a remote client. Rather, they use Java native methods to directly access a local queue manager running under z/OS. This restriction implies that the MQSeries classes for Java on z/OS cannot be downloaded as part of an applet and cannot be run inside an applet viewer or Web browser. However, as the MQSeries classes for Java share a common programming interface, application code can be written to use either the MQSeries classes for Java on z/OS or the MQSeries classes for Java supplied with MQSeries Version 5 platforms, which do support client connections and may be downloaded as part of an applet.

5.2.2 MQSeries classes for Java Message Service (JMS)

MQSeries classes for Java Message Service is a set of Java classes that implement Sun Microsystem's Java Message Service specification. A JMS application can use the classes to send MQSeries messages to either existing MQSeries or new JMS applications. An application can be configured to connect as an MQSeries client using TCP/IP, or directly using the Java Native Interface (JNI). If the client-style connection is used, no additional MQSeries code is required on the client machine.

The JMS classes provide some additional features not present in the MQSeries classes for Java. These extra features include:

- ▶ Explicit support for publish and subscribe
- ▶ Asynchronous message delivery
- ▶ Message selectors
- ▶ Structured message classes

5.2.3 Software prerequisites

Depending on your environment and what you want to do, there are two available SupportPacs.

The *MA1G* (MQSeries classes for Java on z/OS) SupportPac requires:

- ▶ OS/390, Version 2 Release 6 or above (including UNIX System Services)
- ▶ Java for OS/390 at the JDK 1.1.8 level or above
- ▶ MQSeries for MVS/ESA, Version 1 Release 2 or above
- ▶ CICS Transaction Server for OS/390, Version 1.3 or above
Optional; only in case the classes are to be used in JCICS transactions.
- ▶ VisualAge for Java Enterprise Edition for OS/390, Version 2 or above
Optional, only if the Java applications using the MQSeries classes for Java are to be compiled using the HPCJ compiler.

MA88 includes the MQSeries classes for Java and the MQSeries classes for JMS.

MQSeries classes for Java require:

- ▶ An MQSeries server
 - For JNI connections, this must be MQSeries V5.1 or V5.2.
 - For the OS/390, z/OS environments, MQSeries for MVS/ESA V1.2, MQSeries for OS/390 V2.1, and MQSeries for OS/390 V5.2 are supported.

- ▶ OS/390, Version 2 Release 9 or above (including UNIX System Services)
- ▶ Java for OS/390 at the JDK 1.3 level or above

MQSeries classes for JMS require:

- ▶ MQSeries classes for Java
- ▶ An MQSeries server
 - MQSeries V5.2 is required to use the JMS JTA/XA interface.
 - MQSeries for OS/390 V5.2 is required to use JMS in the OS/390, z/OS environment.
- ▶ If you wish to use Pub/Sub applications, you also need one of the following:
 - SupportPac MA0C: MQSeries Publish/Subscribe
 - MQSeries Integrator V2

The following restrictions apply:

- ▶ TCP/IP client connectivity is not supported from OS/390 or z/OS.
- ▶ The OS/390, z/OS version does not support CICS or High Performance Java (HPJ). Users requiring this support should consider SupportPac MA1G: MQSeries for MVS/ESA - MQSeries classes for Java.

From now on we use MA88: MQSeries classes for Java and the MQSeries classes for JMS, and only refer to this SupportPac.

5.2.4 Installation and configuration

The MQSeries classes for Java are available as downloads from the following Web site:

<http://www.ibm.com/software/ts/mqseries/txppacs/ma88.html>

For additional documentation, the preceding Web site has a PDF file for *MQSeries Using Java*, SC34-5456.

Installation overview

The following are the steps for making the environment available on z/OS:

- ▶ Download the tar file to your workstation.
- ▶ Upload the tar file to UNIX System Services.
- ▶ Uncompress and untar the file.
- ▶ Update the environment variables.

Downloading the ma88.tar.Z file

Download the tar file from the Web site:

<http://www.ibm.com/software/ts/mqseries/txppacs/ma88.html>

to your local hard drive.

Uploading the tar file to UNIX System Services

Using ftp in binary mode, upload and install the ma88_zos.tar.Z file to a UNIX System Services directory (we chose /usr/lpp/mqm):

```
ftp wtsc61oe.itso.ibm.com
binary
cd /tmp
put ma88_zos.tar.Z
```

Uncompress and untar the file with the following commands:

```
cd /usr/lpp
mkdir mqm
tar -xpozf /tmp/ma88_zos.tar.Z
```

Updating environment variables

In order to compile and run Java programs using the MQSeries classes for Java on z/OS and the MQSeries classes for JMS, you need to have access to the Java API classes and the native drivers. This is true for both the UNIX System Services environment and WebSphere Application Server (for running servlets and EJBs). Make the following additions:

- Add the following Java classes to CLASSPATH as shown in Example 5-1.

Example 5-1 CLASSPATH

```
export CLASSPATH=/usr/lpp/mqm/java/lib/com.ibm.mq.jar:
/usr/lpp/mqm/java/lib/connector.jar:
/usr/lpp/mqm/java/lib:
/usr/lpp/mqm/java/lib/com.ibm.mqjms.jar:
/usr/lpp/mqm/java/lib/jms.jar:
/usr/lpp/mqm/java/lib/jta.jar:
/usr/lpp/mqm/java/lib/providerutil.jar:
/usr/lpp/ldap/lib/ibmjndi.jar:
/usr/lpp/ldap/lib/jndi.jar:
/usr/lpp/mqm/java/samples/base:$CLASSPATH
```

If there are existing applications with a dependency on the deprecated bind package com.ibm.mqbind, you must also add the file com.ibm.mqbind.jar to your CLASSPATH.

- Add the native drivers contained in /usr/lpp/java/lib to LIBPATH, as in the following example:

```
export LIBPATH=$LIBPATH:/usr/lpp/mqm/java/lib
```

- Some of the JMS scripts require that MQ_JAVA_INSTALL_PATH is defined, and that it points to the directory in which MQ JMS is installed. If you intend to use these scripts, then set this variable. It is not required for the IVPs.

```
export MQ_JAVA_INSTALL_PATH=/usr/lpp/mqm/
```

We assume that you have already installed the Java Development Kit (JDK), which is a prerequisite for the MQSeries classes for Java.

Download documentation

The publication *MQSeries Using Java*, SC34-5456 can be downloaded from

```
http://www.ibm.com/software/ts/mqseries/txppacs/ma88.html
```

and also from

```
http://www.ibm.com/software/mqseries/
```

This manual explains in detail how to use the MQSeries classes for Java and the MQSeries classes for JMS.

5.2.5 Installation verification process

MQ classes for Java

An installation verification sample program called MQIVP is provided in /usr/lpp/mqm/java/samples/base. It tests the connection modes of MQ base Java. Once the environment variables have been set, you can execute the IVP program:

```
java MQIVP
```

You will be prompted for some information. Use the following values:

- Type of Connection: MQSeries (default)
- IP address of Server: leave blank
- Queue manager name: MQSA (on our system) and in uppercase.

After you press Enter, the program contacts the MQ queue manager and then places and retrieves a message from the default system queue. The expected output is shown in Figure 5-1 on page 182.

```

COOK2 @ SC63:/pp/mqm/java/samples/base>java MQIVP
MQSeries for Java Installation Verification Program
5639-B43 (C) Copyright IBM Corp. 1997, 1998. All Rights Reserved.
=====

Please enter the type of connection (MQSeries or VisiBroker) : (MQSeries)
Please enter the IP address of the MQSeries server           :
Please enter the queue manager name                         :MQSA
Success: Connected to queue manager.
Success: Opened SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Put a message to SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Got a message from SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Closed SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Disconnected from queue manager

Tests complete -
SUCCESS: This MQSeries Transport is functioning correctly.
Press Enter to continue ...

```

Figure 5-1 Output from the MQIVP program

MQ classes for Java Messaging Service

An installation verification test (IVT) program called PTSSample01.java is provided in /usr/lpp/mqm/java/samples/jms. The IVT attempts to verify the installation by connecting to the specified queue manager on the local machine, using the MQ JMS in bind mode. It then sends a message to the SYSTEM.DEFAULT.LOCAL.QUEUE queue and reads it back again.

Because a JNDI-based repository is relatively complex to set up, we are running the IVT. The administered objects are created at runtime.

Once the environment variables have been set, compile the IVT:

```
javac PTPSample01.java
```

and run it:

```
java PTPSample01 -nojndi -m QMgr
```

The expected output is shown in Figure 5-2 on page 183.

5648-C60 (c) Copyright IBM Corp. 1999. All Rights Reserved.
MQSeries Classes for Java(tm) Message Service - Point to Point Sample 1

```
Creating a QueueConnectionFactory
Creating a Connection
Starting the Connection
Creating a Session
Creating a QueueSender
Creating a QueueReceiver
Creating a TextMessage
Adding Text
Sending the message to queue:///SYSTEM.DEFAULT.LOCAL.QUEUE
Reading the message back again

Got message:
JMS Message class: jms_text
  JMSType:          null
  JMSDeliveryMode:  2
  JMSExpiration:    0
  JMSPriority:       4
  JMSMessageID:     ID:c3e2d840d4d8e2c140404040404040b6b53bbc2e702a61
  JMSTimestamp:     1005243320380
  JMSCorrelationID: null
  JMSDestination:   queue:///SYSTEM.DEFAULT.LOCAL.QUEUE
  JMSReplyTo:       null
  JMSRedelivered:   false
  JMSXAppID:        COOK2
  JMS_IBM_Format:   MQSTR
  JMS_IBM_PutApplType:2
  JMS_IBM_MsgType:  8
  JMSXUserID:       COOK2
  JMSXDeliveryCount:1
A simple text message from PTPSample01
Reply string equals original string
Closing QueueReceiver
Closing QueueSender
Closing Session
Closing Connection
finished
```

Figure 5-2 Output from PTPSample01

For more information about Java programming for MQSeries, refer to *Java Programming Guide for OS/390*, SG24-5619, and *e-business Cookbook for z/OS Volume III: Java Development*, SG24-5980.

Java connectors for CICS

This chapter gives a broad overview of the Java connectors available for use with CICS. During the writing of this redbook the connectors were radically changed. The Common Connector Framework became obsolete. The WebSphere for z/OS CICS EXCI connector (known as “beta” connector) succeeded it as a Java2 Connector Architecture (JCA)-compliant connector. Finally, during the writing of the book, the official JCA-compliant connector has become supported in the new 4.0.2 version of the CICS Transaction Gateway. We show how to install the newest ECI resource adapters for WebSphere for z/OS in “Enabling CICS JCA connector support” on page 190. For a complete description of the use of this new resource adapter, see *From code to deployment: Connecting to CICS from WebSphere V4.01 for z/OS*, REDP0206.

Restriction: For the z/OS environment, only ECI adapters are supported and WebSphere for z/OS should reside on the same z/OS system as the CICS to be connected. This restriction is known as “local” mode only.

We provide explanations of the following:

- ▶ CICS Transaction Gateway
- ▶ CTG Application Programming Interfaces
- ▶ The CICS CCF connector
- ▶ The CICS connector for CICS TS
- ▶ J2EE CICS resource adapters

6.1 CICS Transaction Gateway

The CICS Transaction Gateway (CTG) has a long heritage as a Java connector for CICS, originally being provided as the CICS Gateway for Java, which was available as a free download for use with the CICS Client. Since then the CTG has advanced along with the Java world and now provides three principal interfaces for communication with CICS:

- ▶ Base classes
- ▶ Common Connector Framework API
- ▶ J2EE Common Client Interface

For further details on the features and facilities provided by the CTG, see *Java Connectors for CICS*, SG24-6401.

6.2 CTG APIs

The CTG provides its own set of base classes and additionally a set of EPI Support classes and EPI beans.

CTG base classes

The CTG provides a set of base classes that provide a simple but low-level interface to CICS. They are relatively easy to use but require a reasonable understanding of CICS to implement. Three request classes are provided:

ECIRequest	Provides a Java interface to the ECI and is used for calling COMMAREA-based CICS applications.
EPIRequest	Provides a Java interface to the EPI and is used for invoking 3270-based transactions. Due to its low-level nature, developing EPI applications requires substantial knowledge of CICS and 3270 data streams, and for this reason we do not provide any examples in this book and advise you to use the ECI support classes instead.
ESIRquest	Provides a Java interface to the ESI, which invokes the Password Expiration Management (PEM) functions in CICS. This provides the ability to verify and change passwords.

Restriction: Note that the CCF and J2EE connectors do not provide an alternative interface to the ESIRquest class, as they do for the EPI and ECI interfaces.

EPI support classes

The EPI support classes provide high-level constructs for handling 3270 data streams. You do not need detailed knowledge of 3270 data streams to use these classes and they are considerably easier to use than the `EPIRequest` class.

A wide range of classes is provided including `AID`, `FieldData`, `Screen`, `Terminal`, `Map` and `MapData`. These are used to represent the interface to a CICS 3270 terminal and the resulting 3270 response. For details on how we used these support classes to develop test applications, see *Java Connectors for CICS*, SG24-6401.

6.2.1 EPI beans

The EPI beans, are based on the EPI support classes and JavaBean development environment. They allow you to create EPI applications in a visual development environment, using one of the visual application builder tools, such as VisualAge for Java.

For further information on using the EPI beans refer to the IBM Redbook, *CICS Transaction Gateway and More CICS Clients Unmasked*, SG24-5277.

6.3 CICS CCF connector

The IBM Common Connector Framework (CCF) is an architecture that provides Java developers with a standardized set of interfaces to access Enterprise Information Systems (EIS). The CCF connector classes implement the CCF interfaces and programming model. The CCF is comprised of three core components:

- ▶ **CCF classes**

The principal components are `CICSConnectionSpec`, `ECIInteractionSpec`, and `EPIInteractionSpec`, which are used to control the interaction with the CICS EIS.

- ▶ **Java Record Framework**

The Java Record Framework is used to build *Record* objects to wrap data structures such as a COBOL COMMAREA or a BMS map. It provides a powerful set of marshalling options for encoding of data and retrieving fields from the COMMAREA or BMS data streams.

- ▶ **Enterprise Access Builder (EAB)**

The EAB is a function of VisualAge for Java. Through a set of SmartGuides, it allows *Command beans* to be developed which encapsulate the CCF classes and the Java Record Framework *Records*.

The CICS Transaction Gateway provides the `CICSConnectionSpec`, `ECIInteractionSpec`, and `EPIInteractionSpec` classes that implement the CCF connector for CICS. These classes are provided in the CTG class library `ctgclient.jar`, and are also provided by VisualAge for Java and by CICS Transaction Server V2. For further information on the CCF, see *CCF Connectors and Databases Connections using WebSphere Advanced Edition*, SG24-5514.

6.4 CICS Connector for CICS TS

In CICS TS V2.1 and V2.2, a Java program or enterprise bean running within CICS can use the new *CICS Connector for CICS TS* to link to any CICS program with a COMMAREA interface. This connector runs within the CICS region and provides the same CCF connector interface as provided by the CTG. However, when using the CICS Connector for CICS TS, the CTG is not required, because the connector runs within the CICS TS V2 environment.

Note that since the CICS connector for CICS TS is based on the technology in the CTG class library `ctgclient.jar`, it is also possible to develop and deploy Java applications into CICS TS V2 that use the CTG `JavaGateway` and `ECIRequest` objects, instead of using the CCF `CICSConnectionSpec` and `ECIInteractionSpec` objects.

Note: As an alternative to the CICS connector for CICS TS you can use the `link()` method provided in the `JCICS ibm.cics.server.Program` class. This is a lower level interface; if used in a session bean, it will prevent the session bean being deployed in an environment other than CICS.

For more information on the CICS connector for CICS TS, see *Enterprise JavaBeans for z/OS and OS/390 CICS Transaction Server V2.1*, SG24-6284.

6.5 JCA connectors

The J2EE Connector Architecture (JCA) defines a standard architecture for connecting the Java 2 Platform Enterprise Edition (J2EE) platform to heterogeneous EISs such as CICS. Java applications interact with resource adapters using the Common Client Interface (CCI), which is largely based on the CCF, but is a standard open to the entire Java community. For further details on the J2EE connector architecture, see *Java Connectors for CICS*, SG24-6401.

The connector architecture enables an EIS vendor to provide a standard *resource adapter* for its EIS. A resource adapter is a middle tier between a Java application and an EIS, which permits the Java application to connect to the EIS.

Currently there are two resource adapters available for use with CICS: The CICS ECI resource adapter and the EPI resource adapter. CTG V4.0.1 provides an ECI and an EPI resource adapter, which together provide the ability to call COMMAREA-based CICS programs, and start 3270-based CICS transactions from a J2EE environment. They can be used in any Java application as a nonmanaged environment or from a session bean in a managed environment in WebSphere Application Server Advanced Edition V4. The new CTG V4.0.2 is generally available and provides the JCA-compliant ECI adapter for WebSphere for z/OS and OS/390.

6.6 Enabling CICS JCA connector support

This section describes how to install and configure the CICS ECI resource adapter into WebSphere for z/OS. The steps required for configuration are fully described in the “Configuring the WebSphere for z/OS-supported connectors” section of Chapter 5 in *WebSphere Application Server V4.01 for z/OS and OS/390 Installation and Customization*, GA22-7834. Make sure you obtain at least the Fifth Edition of this manual, which was published on April 2, 2002. The instructions in the section summarize the instructions given in this manual.

6.6.1 Preparing the CICS ECI resource adapter

This section describes the steps necessary to prepare for the use of the CICS ECI resource adapter with WebSphere Application Server. It consists of the following steps:

- ▶ CICS Transaction Gateway considerations
- ▶ CICS Transaction Server considerations
- ▶ Installation of the CICS ECI resource adapters

CICS Transaction Gateway considerations

We assume you have CICS Transaction Gateway V4.0.2 installed.

To configure the CICS ECI resource adapter you need to determine whether you will use a generic or a specific EXCI pipe to communicate with CICS. The default is to use a generic pipe. If you wish to use a specific pipe, you must define the DFHJVPIPE environment variable in the J2EE server that will be used for the CICS connector support. This value must match the NETNAME parameter of an installed CONNECTION definition in CICS.

CICS Transaction Server considerations

We assume you have a CICS Transaction Server V1.3 region, or higher, configured and running.

You need to perform the following tasks:

1. Set the RRMS SIT parameter.
2. Configure EXCI in CICS.
3. Define an EXCI library to your J2EE server.
4. Define security profiles.

Setting the RRMS SIT parameter

For each CICS region that you wish to access through the CICS ECI resource adapter, set the following System Initialization Table parameter:

RRMS=YES

Configuring EXCI in CICS

Ensure that EXCI is configured in each CICS region you wish to access using the CICS ECI resource adapter. This involves installing a SESSIONS and CONNECTION resource definition. Samples are provided in the CICS-supplied group DFH\$EXCI. Also ensure IRC is set to Open.

Defining an EXCI library to your J2EE server

The J2EE server that will use the CICS ECI resource adapter needs to access and load the CICS module DFHXCSTB from the SDFHEXCI library supplied with CICS. This library is typically CICSTS13.CICS.SDFHEXCI or CICSTS22.CICS.SDFHEXCI, depending on the CICS release and naming conventions you use. Complete one of the following steps:

- ▶ Add module DFHXCSTB to the LPA.
- ▶ Add the SDFHEXCI library to either the system LINKLIST concatenation, or to the STEPLIB concatenation of the J2EE server.

Defining security profiles

If your CICS region uses SURROGAT checking for CICS regions (as defined in the DFHXCPT table), you must authorize the userid associated with the J2EE server region to act as a surrogate for the clients that invoke J2EE applications running in the J2EE server. Complete the following:

1. For all potential clients of J2EE application components that will use the CICS ECI resource adapter, define a RACF CICS DFHEXCI SURROGAT profile definition. You may define one profile for each user, one profile for each group of users, or a single surrogate profile for all users.

RDEFINE SURROGAT *.DFHEXCI UACC(NONE) OWNER(profile-owner-userid)

2. Authorize the userid of the J2EE server region to be the CICS DFHEXCI surrogate for the specific userid or set of userids that you just identified through the profile in the RACF SURROGAT class.

PERMIT *.DFHEXCI CLASS(SURROGAT) ID(server-userid) ACCESS(READ)

Installation of the CICS ECI resource adapter

Resource adapters are packaged in Resource Adapter Archive (RAR) files. The CICS ECI resource adapter for z/OS is provided in the file cicseciRRS.rar in the deployable directory of the CICS Transaction Gateway. Note the following:

- ▶ This cicseciRRS.rar resource adapter archive is designed exclusively for z/OS. It makes use of RRS to provide global transaction support. All non-z/OS platforms use the cicseci.rar file instead.
- ▶ A RAR file is much like an EAR file. It is an archive made up of JAR files and a deployment descriptor.
- ▶ Unlike many application servers, WebSphere Application Server for z/OS cannot work with RAR files directly, so you must extract the contents of the RAR file into a directory, and point the WebSphere classpath to each JAR file in this directory.

To install the CICS ECI resource adapter, perform the following:

1. Create a new directory in UNIX System Services. Extract the contents of the RAR file to this directory. For example:

- /usr/lpp/connectors

Make sure your userid has *write* access to this directory, and grant *read* and *execute* permissions to everyone else. We recommend that you also mount an HFS against this directory.

2. Locate the cicseciRRS.rar file and copy it to the connectors directory you just created. By default, cicseciRRS.rar will be in the following directory:

- /usr/lpp/ctg/deployable

3. In the connectors directory, extract the contents of cicseciRRS.rar. To do this:

- a. Add the bin directory of your Java installation to the PATH environment variable if it is not already defined. For example:

```
export PATH=/usr/lpp/java/IBM/J1.3/bin:$PATH
```

- b. Enter the following command from the connectors directory:

```
jar -xvf cicseciRRS.rar
```

This will extract the following directory and files:

- META-INF
- ccf2.jar
- cicseciRRS.jar
- cicsframe.jar
- ctgclient.jar
- ctgserver.jar
- libCTGJNI.so
- libCTGJNI_g.so

4. Use the following commands to change the permissions of the libCTGJNI.so and libCTGJNI_g.so files:
 - `chmod ugo+x libCTGJNI.so`
 - `chmod ugo+x libCTGJNI_g.so`

6.6.2 Defining connection information

You are now ready to define connection information to the J2EE server. Perform the following:

1. From the Systems Management User Interface, create a new conversation. This conversation will be used to add the CICS connector support.
2. Turn on connection management in the sysplex. To do this, right-click the sysplex name in the conversation and select **Modify**. In the Configuration Extensions section check the box labeled Connection Management. Click **Selected -> Save** to save your changes.
3. Expand the list of J2EE servers and locate the J2EE server that you wish to use with the CICS ECI resource adapter. Right-click the J2EE server and select **Modify**. Make the following changes in the *environment variable list*:
 - a. Add the following classes from the connector directory you created earlier to the CLASSPATH variable:
 - `cicseciRRS.jar`
 - `cicsframe.jar`
 - `ctgserver.jar`
 - `ctgclient.jar`

Note: Separate each entry with a colon (:)

- b. Add the connectors directory to the LIBPATH. For example:
`/usr/lpp/connectors`
 - c. If you are using a specific EXCI pipe, you must set the DFHJVPIPE environment variable here.
 - d. We recommend that you initially turn on JNDI tracing so you can see the connections being made to CICS. Set the CTG_JNI_TRACE environment variable to a file, for example:
 - `CTG_JNI_TRACE = /tmp/jniTrace.txt`

Press **Selected -> Save** to save your changes.

4. To define a CICS ECICConnectionFactory as a new J2EE resource, perform the following:

- a. Locate the J2EE Resources folder in your sysplex, right-click it and select **Add**.
- b. In the properties window, enter the following:
 - i. Set the *J2EE resource name*. We used **CICS_ECI**.
 - ii. Set the *J2EE resource type* to **CICS_ECICConnectionFactory**.

Note: If you are not presented with the CICS_ECICConnectionFactory option in the J2EE resource type list, then APAR PQ55873 may not have been applied. If the APAR has been applied and you still do not have a CICS_ECICConnectionFactory resource type, move to the directory:

`/usr/lpp/WebSphere/samples/`

Look for the following two files:

`CICS_ECICConnectionFactory.properties`

`CICS_ECICConnectionFactory.xml`

If these files exist, then copy them to the following directory (replacing *SYSPLEX* with the sysplex name you are using):

`/WebSphere390/CB390/SYSPLEX/resources/templates`

This is the directory the SM EUI uses to locate J2EE resource types by default. Restart the SM EUI to pick up the change.

- iii. Press **Selected -> Save** to save your changes. Look for the following message in the status bar to confirm the operation completed successfully:
`BBON0515I J2EE Resources name was added.`
5. You can now create a J2EE resource instance that defines the connection information. To do this, perform the following:
 - a. Expand the J2EE resource you have just created. This displays a J2EE resource instances folder. Right-click it and select **Add**.
 - b. In the properties window, enter the following:
 - i. Set the *CICS_ECICConnectionFactory instance name*. We used **CIC2**.
 - ii. Set the *System name* with which this J2EE resource instance is to be associated.
 - iii. Set the *Server name* to the APPLID of the CICS server you wish to call. We used **SC58CIC2**.

- c. Click **Selected** -> **Save** to save your changes. Look for the following message in the status bar to confirm that the operation completed successfully:

BB0N0515I J2EE resource Instance [name] was added.

The J2EE server region is now configured to use the CICSECI resource adapter.

Archived

Java-based access to IMS

This chapter introduces some basic concepts and the connectivity options that are provided by IMS.

We included concepts based on APPC and OTMA. There are other possibilities for connecting to IMS, but we do not consider them here because they are not Java-based or require a client to access IMS.

Beside the APPC-based concept like the Procedural Application Adapter, and OTMA-based concepts like the OTMA/Callable Interface, and the MQSeries-IMS bridge, we describe the tasks of defining and tailoring IMS Connect and guidelines for using IMS Connector for Java.

While we were writing this redbook, the connectors were radically changed. The Common Connector Framework became obsolete and the WebSphere for z/OS IMS connector (known as “beta” connector) succeeded it as a Java2 Connector Architecture (JCA, sometimes called J2C) compliant connector. Since then, the official JCA-compliant connector has become supported in the new 1.2 version of IMS Connect and version 1.2.2 of IMS Connector for Java. Refer to the IBM IMS pages on the Internet for the latest details on the connectors.

7.1 Introduction to IMS connector options

There are two protocols that IMS can use to communicate externally (although a simulation of a 3270 device would also be a possibility): Open transaction Manager Access (OTMA) and Advanced Program-to-Program Communication (APPC). The access to IMS based on APPC is described in 7.2, “APPC-based access to IMS” on page 198.

After the introduction of APPC support in the IMS Transaction Manager, IMS development decided IMS/TM should not be extended to include support for new protocols. One unique protocol called the Open Transaction Manager Access (OTMA) connects IMS to protocol gateways used for new protocols instead. OTMA is a very fast protocol based on the Extended Connection Facility (XCF). XCF does not necessarily require a Coupling Facility (CF). There are two options for connecting via OTMA:

- ▶ Through the OTMA Callable Interface (OTMA/CI) described in 7.3, “OTMA Callable Interface” on page 217.
- ▶ Through a protocol gateway. The gateway indicates that a transformation of a protocol is taking place on the backend side. In both cases the gateway looks like an OTMA client.

There are currently two gateways available for IMS:

- The MQSeries-IMS Bridge, which is delivered with the Message Queue Interface (MQI) product for the frontend, which is described in 7.4, “MQSeries-IMS bridge” on page 225.
- The IMS Connect gateway product (formerly called ITOC). TCP/IP or a Program Call API can be used for the frontend access, which is described in 7.5, “IMS Connect” on page 234.

The IMS Connector for Java is described in 7.6, “IMS Connector for Java based on IMS Connect” on page 258.

7.2 APPC-based access to IMS

Advanced Program to Program Communication (APPC) is a fast and reliable protocol that can be used in a Wide Area Network or locally on the same system. When used locally, storage-based transport is used, omitting all VTAM address space support. For this reason, using APPC in a local environment is a highly recommended option in terms of performance. Although APPC might be

considered obsolete, several of its qualities are unmatched even by newer protocols. But to exploit all the possibilities that APPC offers, like support for unsolicited messages and time-out facilities, it has to be programmed in C and wrapped into Java via JNI support.

APPC conversation is the logical connection between transaction programs and is carried out over a session. Conversations serially reuse a session to exchange end-user information. When a conversation ends, another conversation can be allocated and use the same session. This allows you to avoid overhead of session setup every time a client request is made without additional program development.

There are several implementations of the LU 6.2 protocol (programming interface) in OS/390, but the one that is easiest to use and portable across multiple platforms is Common Programming Interface for Communications (CPIC), so we recommend using this API. An application designed to access IMS via APPC will have the flow illustrated in Figure 7-1, as follows:

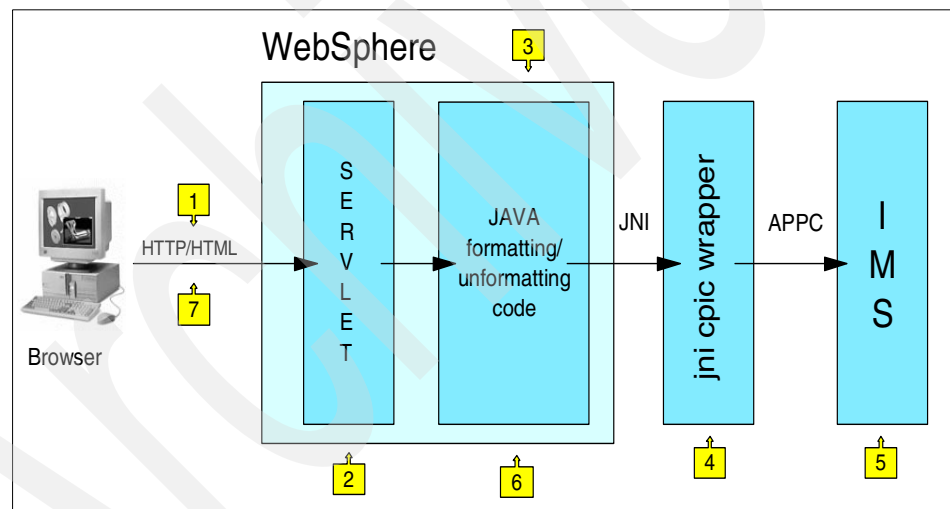


Figure 7-1 IMS access from a servlet via APPC

1. A Web browser requests the execution of a servlet.
2. The servlet receives all the information that the end user entered into the HTML form. The data entered by the user is passed to formatting services to be prepared for transaction execution.
3. The Java formatting program formats the message to be sent to IMS. This Java program makes a call to JNI, specifying all the parameters needed to establish an LU 6.2 conversation with IMS. These parameters are:

- Symbolic Destination name (CPIC profile)
 - Transaction Program name (logical network name)
 - LU name
 - Data (formatted message)
 - A time-out value
4. A native C program establishes a conversation with IMS, sends data, and receives the response from IMS. The response data or the return code is sent back to the Java servlet.
 5. IMS processes the transaction.
 6. The Java servlet formats the response from IMS and converts it to an HTML page to be sent to the client.
 7. The HTML page is returned to the browser.

APPC requires the implementation of a Communication Manager and acts as a Communication Resource Manager (CRM).

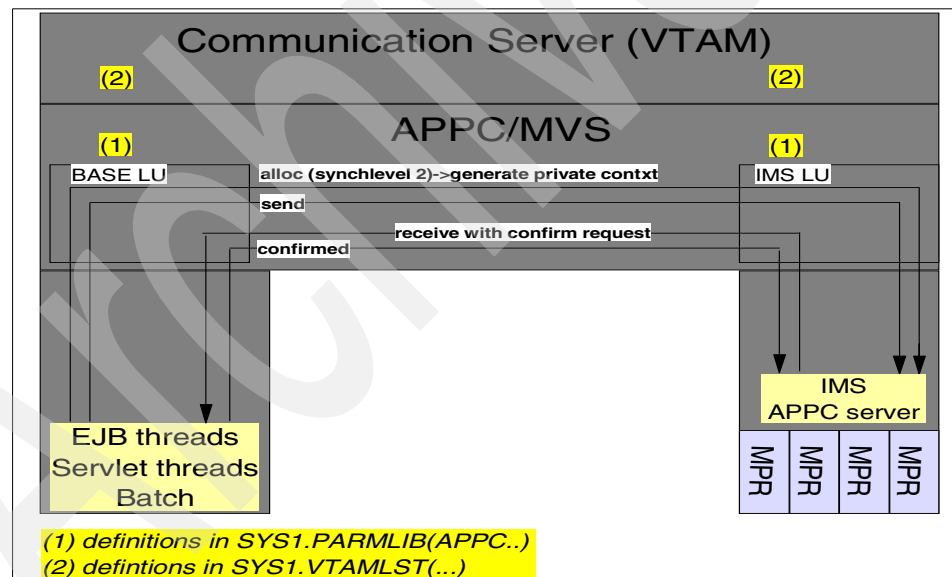


Figure 7-2 APPC flow with IMS

IBM provides an APPC connector called WS/390 IMSAPPC Connector, based on the J2EE Connector Architecture. It is made available in "beta state" together with the general announcement of WebSphere Application Server for z/OS and OS/390 Version 4. The WebSphere IMSAPPC Connector implements the Sun

J2EE Connector Architecture. It allows EJBs running under a WebSphere server region to connect to an IMS host system for the purpose of issuing IMS transactions. It uses APPC to establish a connection to IMS, enabling IMS to reside on a different system from the one the WebSphere server region is located on. The connector supports 2-phase commit of global transactions coordinated across multiple resource managers using the z/OS and OS/390 Resource Recovery Services (RRS).

Note: Currently this implementation cannot handle multisegment output or conversational transactions. Before making any decisions based on this statement, though, refer to the most current version. Changes might have occurred.

The WebSphere IMSAPPC Connector is fully documented in the *WebSphere/390 Connector Installation and Usage Guide*. The most current version of this publication is contained in WS390Connectors_yymmdd.zip at this Web site:

http://www.ibm.com/software/webserver/appserv/download_v4z.html

Based on our experiences and publications such as Chapter 5, “Performing advanced tasks” in *WebSphere Application Server V4.01 for z/OS and OS/390 Installation and Customization*, GA22-7834 and *WebSphere/390 Connector Installation and Usage Guide*, the following sections describe how to set up the environment for the WebSphere IMSAPPC connector. For more details on the connector and its usage in applications, refer to *e-business Cookbook for z/OS Volume III: Java Development*, SG24-5980.

7.2.1 Elements of the WebSphere IMSAPPC Connector

The WebSphere IMSAPPC Connector provides the following classes, which implement the J2EE Common Client Interface (CCI) as defined by the J2EE Connection Architecture (JCA, sometimes called J2C):

- **IMSAPPCConnectionFactory**

Factory objects are stored in the java:comp/env/name space. A JNDI lookup is still required to access objects stored in this space. This space, however, resides in the server address space and not in LDAP, so a JNDI lookup is going to be relatively fast. A connection factory is an architecture dictated object responsible for manufacturing and managing connections. A connection factory allows a service provider to provide caching for instance, so that connections can be shared in a more effective and controlled manner.

- ▶ **IMSAPPCConnection**

A connection object represents a virtual software wire between a client, in this case, the EJB, and the server, in our case IMS. In this context, the connection provides an abstraction on top of the underlying communication protocol APPC.

- ▶ **IMSAPPCConnectionSpec**

Objects of this class represent connection specifications. They serve to specialize and provide details on a particular connection.

- ▶ **IMSAPPCInteractionSpec**

Provides the details for driving the interaction with IMS using APPC.

- ▶ **IMSAPPCInteraction**

Objects of this class are created by the `createInteraction()` method of the connection instance. An interaction instance is used for executing transaction requests against the target IMS associated with the connection that was obtained.

- ▶ **IMSAPPCConnectionMetaData**

Provides information about an APPCConnection.

- ▶ **IMSAPPCResourceAdapterMetaData**

Provides information about APPC Resource Adapter.

7.2.2 Preparations for setting up the IMS - APPC Procedural Application Adapter for WebSphere

This section provides installation and usage information for the WebSphere V4 IMSAPPC Connector, which is available as beta-only support. This connector support is contained in a zip file, `WS390Connectors_yymmdd.zip`, at this Web site:

http://www.ibm.com/software/webservers/appserv/download_v4z.html.

It is assumed that the reader has already obtained the zip file from the Web site and performed the instructions in the Readme file.

Other requirements are:

- ▶ The target IMS must be at IMS release level 6.1 or higher.
- ▶ The fix for APPC APAR OW47988 must be installed. This APAR fix provides surrogate support that allows the userid associated with an address space to be identified to APPC as a surrogate issuer of requests on behalf of other

userid. This APAR is fixed on release 8 and 9 of OS/390 by PTF UW79242 and on release 10 by PTF UW79243.

The IMS-APPC Procedural Application Adapter allows WebSphere to communicate with IMS on a remote or local system through APPC/MVS. APPC/MVS provides a programming interface (LU 6.2 architecture) that WebSphere exploits to communicate on a peer-to-peer basis with application programs. Through settings in WebSphere, you can determine whether an APPC conversation is protected. Three possibilities exist:

- ▶ You can require protected conversations by specifying `syncpt` on the logical resource manager instance. APPC/MVS becomes a communication resource manager and has expressed interest in the outcome of a WebSphere transaction, driving the IMS transaction running on another system under the same transactional scope as WebSphere. All of the processing done on behalf of a distributed application is treated as a single operation. In other words, APPC/MVS, WebSphere, and IMS coordinate their processing so that all application updates are either made (committed) or not made (rolled back). This coordination is most beneficial for applications that have a critical dependency on data integrity.

This transaction management happens automatically when you create a server with the IMS-APPC Procedural Application Adapter and protected conversations. The conversations have syncpoint capabilities; that is, data in your application is synchronized with data in the IMS database.

- ▶ You can allow unprotected conversations by specifying `none` on the logical resource manager instance. You create a server with the IMS-APPC Procedural Application Adapter without syncpoint capabilities. There is no guarantee that data in the IMS database is synchronized with data in your client application. Your client application becomes responsible for rechecking the data in IMS if it makes an update. Although there is no synchronization of data, there are benefits:
 - Your application no longer pays the performance cost associated with distributed transactions.
 - Fewer IMS message processing regions (MPR) become busy. In a transaction, a simple read/write operation requires two MPRs to remain busy until a transaction commitment occurs. If you use no syncpoint capabilities, one MPR can serve a data request and then immediately become available for another request.
- ▶ You can allow WebSphere to determine whether an APPC conversation is protected when the conversation is allocated by specifying `autotran` on the logical resource manager instance. WebSphere makes the determination based on the container transaction policy and the type of transaction the current execution thread is running under.

Container Transaction policies control the type of transaction under which an execution thread runs. Container can require global transactions (TX_REQUIRED) or allow variations of local transactions started by your application (HYBRID_GLOBAL). When WebSphere is about to allocate an APPC conversation and if autotran is the setting, then WebSphere will:

- Allocate a protected conversation if the execution thread is running under a global transaction (the container policy requires a global transaction).
- Allocate an unprotected conversation if the execution thread is running under a local transaction (the container policy allows a local transaction).

Before you set up the APPC connection, you must determine the transactional characteristics of your application and know the appropriate container transaction policy. Details on transaction policies are in *WebSphere Application Server V4.01 for z/OS and OS/390 Assembling Java 2 Platform, Enterprise Edition (J2EE) Applications*, SA22-7836. You need to determine whether you must define the APPC connection with syncpoint capabilities. If you plan to use the syncpt or autotran settings on the logical resource manager instance, you should define syncpoint capabilities for the APPC connection. If you plan to use the none setting on the logical resource manager instance, you do not need to define syncpoint capabilities for the APPC connection.

To set up a server with the IMS-APPC Procedural Application Adapter, you must coordinate the configuration on both sides of the communication path, then define the connection to your WebSphere server through the Administration application (SMUI administration tool):

- ▶ On the WebSphere side (local system), you must coordinate the configuration for VTAM and APPC.
- ▶ On the IMS side (partner system), you must coordinate the configuration for VTAM, APPC, and IMS.
- ▶ Finally, you must define the connection for your WebSphere J2EE Server through the System Management User Interface (SMUI).

For more information on configuring APPC/MVS, see *z/OS MVS Planning: APPC/MVS Management*, SA22-7599.

7.2.3 Steps for setting up the WebSphere (local) side

Before you begin, make sure you have VTAM and APPC installed on your WebSphere system. Also, decide whether your application requires syncpoint capabilities. The userid associated with the WebSphere server regions (used to run EJBs that use the IMSAPPC Connector support) must be authorized to be an APPC surrogate for the users (i.e., clients) that may invoke the EJBs.

First, for all the users/clients that may need to invoke the EJBs that use the IMSAPPC connector, define a RACF APPC surrogate profile for each userid or group of userids. This is done under RACF by issuing the command:

```
RDEFINE SURROGATE ATBALLC.xxxxx
```

where xxxxx is the specified userid, the RACF group, or an * to indicate all userids. For example, in the case where an installation wants to define a single surrogate profile for all users, the RACF APPC SURROGATE profile definition should be:

```
RDEFINE SURROGATE ATBALLC.* UACC(NONE) OWNER(profile_owner_userid)
```

Next, issue a RACF PERMIT to authorize the userid of the WebSphere server region to be the APPC surrogate for the specific userid or set of userids that were identified by a profile in the RACF SURROGATE class. For example, in the case where an installation wants to authorize the userid associated with a WebSphere server region (e.g. BBJRGES) to be a surrogate for all userids, issue the command:

```
PERMIT ATBALLC.* CLASS(SURROGAT) ID(BBJRGES) ACCESS(READ)
```

Finally, make sure that the SURROGAT class is defined in RACLIST, as follows:

```
SETOPTS RACLIST(SURROGAT) REFRESH
```

Note: The SURROGAT class must be RACLISTed. In order to provide the new APPC SURROGAT support, APPC used internal security services that require SURROGAT to be RACLISTed in RACF.

Then perform the following steps:

1. Define a logical unit (an LU specifically for WebSphere) to VTAM in its APPL definitions.
 - To enable syncpoint capabilities for the LU, code the VTAM APPL definition with SYNCLVL=SYNCPT (standard) and ATNLOSS=ALL. Also, you must configure RRS and make it active. This provides the greatest flexibility in handling your application EJBs, which use IMSAPPC Connector support. This type of LU definition handles EJBs that issue requests to IMS while running under a global transaction and thus need to do syncpoint processing (i.e., sync level 2 processing). It will also handle EJBs that are not running under a transaction and wish to have their requests to IMS immediately committed by IMS (i.e., sync level 0 processing). The IMSAPPC Connector processing automatically determines which way processing is to be performed based on whether or not the EJB is running under a global transaction.
 - To run without syncpoint capabilities (not recommended), you do not need to specify either the SYNCLVL or ATNLOSS keywords. In this case, the

IMSAPPC ConnectionFactory J2EE resource that is using this LU must then only be configured to EJBs that run without a transaction. If an EJB that runs with global transaction support is configured to use the IMSAPPC ConnectionFactory J2EE resource, all requests issued to the target IMS will fail as syncpoint (level 2) requests will be issued to the LU, which does not have the syncpoint capabilities enabled.

Tip: Create an LU specifically for WebSphere because you can manage the LU more easily. You only need to define one LU through which all WebSphere initiated conversations will pass. For sample LU definitions, see the sample in SYS1.SAMPLIB(ATBAPPL) .

Our WebSphere LU was defined in member MF1APPLS of the SYS1.VTAMLST data set as shown in Example 7-1. Make sure you define the LU in a member that is set in VTAM's start list; in our case, in SYS1.VTAMLST(ATTCON59). If you want to use a new member, add it to the SYS1.VTAMLST(ATTCONxx) that you use for starting VTAM.

Example 7-1 SYS1.VTAMLST(MF1APPLS)

MF1AP001	APPL	ACBNAME=MF1AP001, APPC=YES, AUTOSSES=0, DDRAINL=NALLOW, PARSESS=YES, MODETAB=LOGMODES, DLOGMOD=TRANPAR, DMINWNL=5, DMINWNR=5, DRESPL=NALLOW, DSESLIM=10, LMDENT=19, SECACPT=CONV, ATNLOSS=ALL, SYNCLVL=SYNCPT, SRBEXIT=YES	
			X <== protected resources
			X <== protected resources

2. Create at least one APPC TP Profile data set (sample in SYS1.SAMPLIB(ATBTPVSM)). We defined it as shown in Example 7-2:

Example 7-2 TPPROF1

//TPPROF1	JOB	<JOB CARD>
//TPSAMPLE	EXEC	PGM=IDCAMS
//VOLUMENM	DD	DISP=OLD,UNIT=3390,VOL=SER=TARSY1
//SYSPRINT	DD	SYSOUT=*
//SYSABEN	DD	SYSOUT=*
//AMSDUMP	DD	SYSOUT=*

```
//SYSIN      DD      *
DEFINE CLUSTER (NAME(SYS1.OMVS.APPCTP) -
  VOLUMES(TARSY1) -
  INDEXED REUSE -
  SHAREOPTIONS(3 3) -
  RECORDSIZE(3824 7024) -
  KEYS(112 0) -
  RECORDS(300 150)) -
DATA -
  (NAME(SYS1.OMVS.APPCTP.DATA)) -
INDEX -
  (NAME(SYS1.OMVS.APPCTP.INDEX))
```

3. Define an APPC LU that matches the LU you defined for VTAM. On the TPDATA keyword, specify the APPC TP Profile data set you created in step 2. If the target IMS LU is located in a different VTAM network, then the WebSphere LU needs to be a fully qualified LU name (i.e., *networkID.networkLUname*) to allow communication with IMS, and the WebSphere LU must have the NQN keyword specified in the APPC LUADD statement.

Tip: Since this LU will likely support outbound conversations only, you can avoid starting up a transaction scheduler and increasing resource overhead by specifying NOSCHED on the LU.

The LU names are defined in the APPCPMxx member in SYS1.PARMLIB. For a sample member, see SYS1.SAMPLIB(APPCPMxx), or our sample:

```
LUADD ACBNAME(MF1AP001) SCHED(ASCH) BASE
      TPDATA(SYS1.OMVS.APPCTP) TPLEVEL(SYSTEM)
```

4. To implement syncpoint capabilities, define the ATBAPPC.LU.LOGNAMES log stream to the system logger.

Note: If WebSphere and IMS are on different systems in the same sysplex, you must use the coupling facility for the log stream. APPC/MVS only supports a DASD-only log stream in a single system environment.

Example 7-3 shows our definition in a sample job called DEFLAPPC.

Example 7-3 DEFLAPPC

```
//DEFLAPPC JOB <JOB CARD>
//STEP1 EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//SYSIN DD *
```

```
DATA TYPE(LOGR) REPORT(YES)
DEFINE LOGSTREAM
NNAME(ATBAPPC.LU.LOGNAMES)
DASDONLY(YES)
HLQ(LOGR)
LS_SIZE(1024)
STG_SIZE(3000)
LOWOFFLOAD(20)
HIGHLOFFLOAD(80)
```

5. Make sure that you have VTAM connectivity to the IMS system. You can use VTAM Subarea, VTAM APPN, or SNA over TCP/IP network configurations.
6. Enable the VTAM APPL into the VTAM configuration. Inactivate VTAM APPL by issuing:

```
/V net,inact,ID=MFAPPLS,I
```

7. Deactivate VTAM APPL by issuing:

```
/V net,act,ID=MFAPPLS
```

8. Start APPC with the new WebSphere LU defined or dynamically activate the new WebSphere LU into the APPC configuration. Issue the following command to verify that the local LU is active. If you want syncpoint capabilities, check that SYNCPT=YES:

```
/D APPC,LU,ALL
```

You know you are done when you see the local LU active and the syncpoint capabilities as wanted.

7.2.4 Steps for setting up the IMS (partner) side

Before you begin, you must have VTAM and APPC installed on your IMS system. Also, decide whether your application requires syncpoint capabilities. Perform the following steps:

1. Define a logical unit (LU) to VTAM that is associated with IMS. This is the LU with which WebSphere will allocate a conversation to establish communications with IMS.
 - To enable syncpoint capabilities for the LU, code the VTAM APPL definition with SYNCLVL=SYNCPT and ATNLOSS=ALL. Configure RRS and make it active. For sample LU definitions, see SYS1.SAMPLIB(ATBAPPL).
 - To run without syncpoint capabilities, you do not need to specify either the SYNCLVL or ATNLOSS keywords.

Example 7-4 represents a sample IMS LU definition for APPC/MVS that is located in the SYS1.VTAMLST member APIMS7A, which is defined in VTAM's start list, in SYS1.VTAMLST(ATTCON59). The IMS acts as an APPC scheduler for its logical unit, which is located in the APPC/MVS Address Space and defined in a SYS1.PARMLIB(APPCPMS1) member. Although the standard OS/390 scheduler ASCH is *not* used, it has to be started.

Example 7-4 SYS1.VTAMLST(APIMS7A)

APIMS59A APPL	AUTH(ACQ),	
	ACBNAME=APIMS59A,	
	EAS=100,	
	APPC=YES,	
	PARSESS=YES,	
	MODETAB=LOGMODES,	
	DLOGMOD=APPCHOST,	
	DSESLIM=10,	
	SECACPT=ALREADYV,	
	ATNLOSS=ALL,	X <== protected resources
	SYNCLVL=SYNCPT,	X <== protected resources

Important: This partner LU must be able to accept a user ID without a password when communication is initiated because WebSphere already verifies the password. You can set this up through the VTAM APPL definition, in which you specify the parameter SECACPT=ALREADYV. An alternative is to set up a RACF APPCLU profile, in which you specify CONVSEC(ALREADYV). Details on APPC security are in the chapter on security in *z/OS MVS Planning: APPC/MVS Management*, SA22-7599.

2. Create at least one APPC TP Profile data set. See SYS1.SAMPLIB(ATBTPVSM) for a sample job. We used the job shown in Example 7-5.

Example 7-5 TPPROF2

```
//TPPROF2 JOB <JOB CARD>
//TPSAMPLE EXEC PGM=IDCAMS
//VOLUMEN DD DISP=OLD,UNIT=3390,VOL=SER=TARSY1
//SYSPRINT DD SYSOUT=*
//SYSABEN DD SYSOUT=*
//AMSDUMP DD SYSOUT=*
//SYSIN DD *
      DEFINE CLUSTER (NAME(SYS1.IMS.APPCTP) -
        VOLUMES(TARSY1) -
        INDEXED REUSE -
        SHAREOPTIONS(3 3) -
        RECORDSIZE(3824 7024) -
        KEYS(112 0) -
```

```

RECORDS(300 150)) -
DATA -
(NAME(SYS1.IMS.APPCTP.DATA)) -
INDEX -
(NAME(SYS1.IMS.APPCTP.INDEX))

```

3. Define an APPC LU that matches the partner LU you defined in VTAM. On the TPDATA keyword, specify the APPC TP Profile data set you created in Step 2. Specify the SCHED keyword with the value of the IMS System Identifier on the LU definition. The LU names are defined in the APPCPMxx member in SYS1.PARMLIB. (For a sample member, see SYS1.SAMPLIB(APPCPMxx). You can add the statement to the member created in Step 3 in 7.2.3, “Steps for setting up the WebSphere (local) side” on page 204.)

```

LUADD ACBNAME(APIMS59A) SCHED(IMS) BASE
      TPDATA(SYS1.IMS.APPCTP) TPLEVEL(SYSTEM)

```

4. To implement syncpoint capabilities, make sure you have a log stream defined for APPC on the IMS side.
 - If IMS is running on the same system as WebSphere, APPC needs a DASD-only or coupling facility log stream called ATBAPPC.LU.LOGNAMES.
 - If IMS is running on a different system in the sysplex than WebSphere, APPC needs a log stream called ATBAPPC.LU.LOGNAMES to be defined to use the coupling facility. That is because APPC/MVS supports a DASD-only log stream in a single-system environment only.
 - If IMS is running on a remote system (not on the same system or sysplex as WebSphere), it needs a log stream called ATBAPPC.LU.LOGNAMES on the remote system. The log stream can use either a DASD-only or coupling facility configuration.

APPC/MVS requires the names of the local and remote partner LU logs, and the negotiated syncpoint capabilities for each LU to provide resource recovery for protected conversations. This information has to be available for resynchronization after a failure. To store this information, APPC/MVS also uses the system logger. More information about this subject can be found in *OS/390 MVS Planning: APPC/MVS Management*, GC28-1807. APPC/MVS must be authorized to use the system logger resources. The APPC/MVS logstream definition has to be present in the LOGR policy. Example 7-6 shows the "DASDONLY logstream" definitions for APPC/MVS, done through the IXCMIAPU utility, assuming that the LOGR definition is correct.

Example 7-6 DEFLAPPC

```

//DEFLAPPC JOB <JOB CARD>
//STEP1 EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=*

```

```
//SYSABEND DD SYSOUT=*
//SYSIN DD *
DATA TYPE(LOGR) REPORT(YES)
DEFINE LOGSTREAM NAME(ATBAPPC.LU.LOGNAMES)
DASDONLY(YES)
LS_SIZE(1024)
STG_SIZE(3000)
LOWOFFLOAD(20)
HIGHLOFFLOAD(80)
```

The name of the APPC/MVS logstream has to be specified as indicated in Example 7-6. If the APPC/MVS logstream is DASDONLY, then cross-domain protected conversations cannot be set up. Cross-domain protection requires the APPC log to be in the Coupling Facility. Particular information about defining logstreams and setting up coupling data sets is available in *OS/390 MVS Setting up a Sysplex*, GC28-1779.

5. Set the IMS parallel scheduling limit to 0, which means that any number of transactions can be scheduled.
6. Size the number of message processing regions (MPR) IMS must have. The sizing depends on whether you use syncpoint capabilities.

- Using syncpoint capabilities

A transaction in a WebSphere application may result in several transactions in IMS. For instance, within a transactional scope in a WebSphere application, a program may perform a `findByPrimaryKey`, three `setters`, and three `getters`, resulting in three separate IMS transactions. This multiplying effect on transactions affects the number of MPRs in the DFSMPR job to equal the number of transactions that could result from a WebSphere transaction. Here you should set the number of MPRs to 3.

- Not using syncpoint capabilities

Specify the number of MPRs according to the number of simultaneous operations you expect IMS to process.

If additional WebSphere applications generate additional IMS transactions on the same database, set the number of MPRs according to the maximum number of transactions that could be generated from all applications.

7. Enable the VTAM APPL into the VTAM configuration.

Activate the VTAM definitions by adding them to the ATTCONxx (if you have not done it already).

8. Start APPC with the new IMS LU defined or activate the new IMS LU dynamically into the APPC configuration.

We issued the following commands:

```
/V net,inact,ID=APIMS7A,I
```

to inactive VTAM APPL first and then:

```
/V net,act,ID=APIMS7A
```

to deactivate VTAM APPL.

To enable the APPC-IMS LU, issue the following IMS command from the MVS or IMS console:

```
/START APPC
```

9. Issue the following command to verify that the local LU is active:

```
/D APPC,LU,ALL
```

You know you are done when APPC starts successfully. If you want syncpoint capabilities, verify that SYNCPT=YES.

After activating the IMSAPPL APPC physical unit, messages similar to the following should be received:

```
ATB227I LOCAL LU USIBMSC.APIMS59A is COLD STARTING AS A RESOURCE 560 MANAGER  
WITH RRS/MVS.
```

```
LOCAL LOG: xxx
```

```
ATB201I LOGICAL UNIT APIMS59A FOR TRANSACTION SCHEDULER IVC1 NOW 561 ACCEPTS  
PROTECTED CONVERSATIONS.
```

A display with the D LOGGER,CONN console command should show the presence of the new logstream for APPC, as shown in Example 7-7.

Example 7-7

```
D LOGGER,CONN  
IXG601I 06.04.27 LOGGER DISPLAY 852  
CONNECTION INFORMATION BY LOGSTREAM FOR SYSTEM SC59  
LOGSTREAM          STRUCTURE      #CONN      STATUS  
-----  
ATR.OPPLEX.RM.DATA  *DASDONLY*  000001    IN USE  
ATR.OPPLEX.MAIN.UR  *DASDONLY*  000001    IN USE  
ATR.OPPLEX.DELAYED.UR *DASDONLY*  000001    IN USE  
ATR.OPPLEX.RESTART  *DASDONLY*  000001    IN USE  
ATR.OPPLEX.ARCHIVE  *DASDONLY*  000001    IN USE  
ATBAPPC.LU.LOGNAMES *DASDONLY* 000001 IN USE  
WAS.ERROR.LOG       *DASDONLY*  000010    IN USE
```

7.2.5 Using IMS Connector for Java based on APPC with WebSphere

At the time of writing there were still uncertainties about the general availability of the IMS Connector for Java classes. We used the beta version of the IMS Connector for Java based on APPC because this was the only one available for the J2EE Connector Architecture. Refer to 7.6, “IMS Connector for Java based on IMS Connect” on page 258 for more details on IMS Connector for Java.

Before you begin you must have WebSphere and its administration tool (SMUI) installed. In most cases the application deployer will come and ask you for the connection data to define the access to IMS with the SMUI. This is what the application deployer needs to consider and what data you need to provide.

Choose IMS_APPC_PAA as the logical resource manager (LRM) subsystem type and identify the following for the LRM instance connection data in the SMUI:

ManagedConnectionFactory class name

This represents the fully qualified class name of the ManagedConnectionFactory to be used to create the particular type of Connection Factory resource that is required. For an IMSAPPCConnectionFactory resource definition, the class name is `com.ibm.connector2.ws390.imsappc.IMSAPPCManagedConnectionFactory` and cannot be changed.

LogWriter Recording

If "enabled", a LogWriter is created for use by connector processing to create record data. Whether or not the connector makes use of the LogWriter is dependent on the logging support provided by the connector.

WebSphere/390 local LU name

Fill in the LU name associated with WebSphere for z/OS. This local LU name is defined in an LUADD statement in the APPCPMxx parmlib member for the system on which WebSphere for z/OS runs.

Look for the LUADD statement for the LU associated with WebSphere for z/OS. Use the value specified on the ACBNAME parameter as the local LU name.

Note: Use only the value specified on the ACBNAME parameter, which is the network LU name. If you specify a network-qualified (or fully qualified) name for the local LU, you will receive the error message BB0U0106E, which indicates that the local LU name is not valid.

Our example: MFAP001

Partner LU name

Fill in the name of the LU that will be used to initiate an APPC conversation for the WebSphere J2EE Server with a target IMS subsystem. This partner LU is defined in an LUADD statement in the APPCPMxx parmlib member for the system on which IMS runs. The IMS subsystem may be, but does not have to be, on a system other than the one on which the WebSphere J2EE Server runs.

Look for the LUADD statement for the LU associated with IMS (an LU associated with IMS has the IMS subsystem name specified for the SCHED parameter on the LUADD statement). Use the value specified on the ACBNAME parameter as the partner LU name.

Tip: When you specify the partner LU name, you may use one of the following forms:

- ▶ Only the value specified on the ACBNAME parameter (the network LU name).
- ▶ A network-qualified name in the form *networkID.networkLUname*.
networkID is the value specified for the VTAM start option NETID and *networkLUname* is the value specified on the ACBNAME parameter.
- ▶ A VTAM generic resource name, if your installation is configured to use generic resources.

Our example: APIMS59A

VTAM logmode name

Fill in the name of the VTAM logmode that designates the network properties to be associated with any APPC conversations between this local LU and its partner LU. Logmode names appear in the VTAM logon mode table, which resides in your installation's VTAMLIB data set.

Our example: TRANSPAR.

You can check the setting by issuing:

```
/d net,e,id=MF1AP001
```

APPC conversation time-out value

Specify the length of time, in minutes, for the WebSphere J2EE Server to wait for a response to the Allocate call and any subsequent calls the server issues during its conversation with IMS. Valid time-out values range from 0 through 1440, which is 24 hours.

If you specify a value that is less than the value set for the `OTS_DEFAULT_TIMEOUT` environment variable, the APPC conversation time-out value will have no effect. Look for the `OTS_DEFAULT_TIMEOUT` environment variable setting that you use for the application server's control and server regions.

APPC sync level

This value controls the type of APPC/MVS conversation the WebSphere J2EE server uses to communicate with IMS. Base your choice on the transaction policies you select for containers in this server configuration, and the characteristics of the applications to be deployed in this server. Use a sync level value that corresponds with the transactional context of the request that the server is currently processing.

- **Syncpt**

With Syncpt, the server allocates a protected conversation, which preserves the global transactional context for the interaction between the server and the IMS subsystem, and allows the system to recover any resources if conversation errors or failures occur.

Recommendation: Use Syncpt if this LRM is connected to one or more containers that use the transaction policy *Required*, or if you cannot guarantee that your server application will always run on the same z/OS or OS/390 system on which the IMS subsystem runs.

- **None**

With None, APPC/MVS, WebSphere, and IMS do not coordinate any processing done on behalf of a distributed application; without the overhead of coordination, your application's performance improves.

Recommendation: Use None judiciously. In this case, resources that the application uses might be in inconsistent states if conversation errors or failures occur.

- **AutoTran**

The easiest way to match the sync level and context is to select Autotran, so the system can determine which conversation type, Syncpt or None, is appropriate for the transactional context associated with the current thread of execution: If the current thread has a local transactional context, the server uses a sync level of None; for a global transactional context, the server uses Syncpt.

Recommendation: Use AutoTran if the LRM is connected to one or more containers that use a transaction policy other than Required.

Our example: AUTOTRAN

If you need additional information about the transactional policies for containers, see *WebSphere Application Server V4.01 for z/OS and OS/390 Assembling Java 2 Platform, Enterprise Edition (J2EE) Applications*, SA22-7836.

You know you are done when you save the logical resource manager and receive the message that the system added the LRM.

7.2.6 Guideline for recovery of IMS transactions

Consider automatic restart management (ARM) and what would happen if a system failed and WebSphere restored on a second system in the sysplex. As long as RRS is not running on the failed system, transactions could complete on the restored system if you had an LU with the same name and attributes as that on the failed system. However, you cannot set up two LUs with the same name and attributes in the same sysplex in anticipation of a failure, because VTAM will not allow it. Rather, you could manually reactivate the WebSphere LU on the restored system after a failure on the first system.

7.2.7 When to choose APPC

You may want to use APPC for connecting WebSphere to IMS for the following reasons:

- ▶ If the IMS application is conversational, the program-to-program communication is connection-oriented. In this case, the network and both programs must be available.
- ▶ When you need good performance.
- ▶ If you need to handle unsolicited messages sent from IMS.
- ▶ If you want to use many clients at a time, as multiple connection support is implicit to the API.

For more information and samples of APPC connectors to IMS, refer to *e-business Application Solutions on OS/390 Using Java: Volume 1*, SG24-5342, and *e-business Cookbook for z/OS Volume III: Java Development*, SG24-5980.

7.3 OTMA Callable Interface

IMS originally provided the OTMA interface to give OS/390 applications a high-performance mechanism for executing transactions. Subsequently, IMS added the complementary OTMA Callable Interface to give some high-level languages (C and C++) access to OTMA. Finally, with the addition of the Java-OTMA package, OTMA access to IMS transactions is now available to z/OS and OS/390 applications and Java applications running in WebSphere Application Server.

IMS OTMA Callable Interface (OTMA/CI) was introduced in IMS Version 6. The OTMA/CI consists of API calls that can be used to join the IMS/OTMA XCF group:

- ▶ Connect to IMS
- ▶ Allocate communication sessions
- ▶ Send IMS transactions/commands
- ▶ Receive output from IMS
- ▶ Close communication sessions
- ▶ Leave an XCF group

A benefit of OTMA/CI is that it is easy to use. It has been possible for the user to implement his own OTMA client, but this has not necessarily been a simple task because an understanding of the technical protocols of MVS XCF and IMS OTMA is required. OTMA Callable Interface extracts the details of OTMA and XCF. OTMA/CI supports the execution of IMS transactions and commands, and it enables programs running from other z/OS subsystems to connect to multiple IMSs. OTMA/CI API calls can be made from an authorized or unauthorized library, and OTMA/CI can connect to all IMS OTMA releases.

The OTMA/CI offers a relatively simple, problem-state C and C++ programming interface which hides the complexity of the protocol that uses the XCF communication layer. The Java OTMA package completes the bridge between OS/390 Java applications and high performance facilities of OTMA and IMS. The Java OTMA package consists of a minimal layer of Java code and JNI code above the OTMA/CI that represents the concepts and services of the OTMA/CI in terms of Java objects and methods. Figure 7-3 on page 218 shows the flow of a Java application to IMS using OTMA/CI.

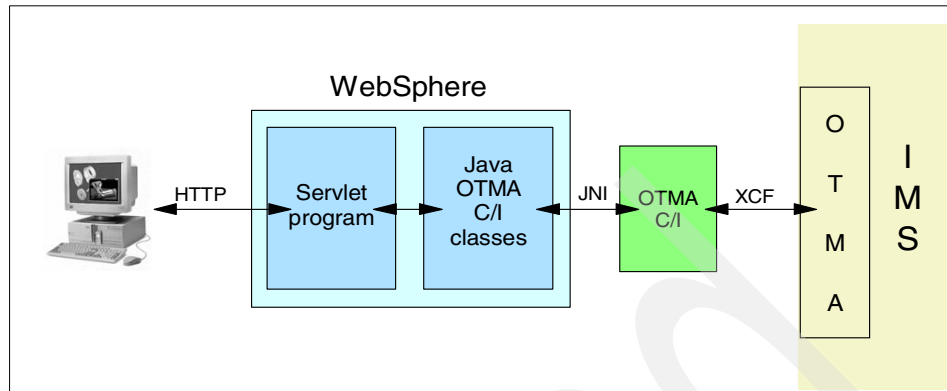


Figure 7-3 Java application flow to IMS using OTMA/CI

The Java OTMA package is a no-charge download package available from the IBM S/390 e-business Web page. The package contains documentation and code samples. You can find those samples in a package called "jotma", which can be downloaded from:

<http://www.s390.ibm.nc/sntc>

The jar file has to be added to the CLASSPATH and the native drivers have to be added to the LIBPATH of the OS/390 runtime environment. The real OTMA/CI is executed through Java Native Interface (JNI) code, which requires an additional DLL routine, libjotmaNative.so, in the LIBPATH. Because many aspects of the OTMA Callable Interface—such as flows, parameter definitions, and return codes—apply to the Java OTMA package, application programmers need to reference the OTMA/CI documentation that can be found at:

<http://www.ibm.com/software/data/ims/otmaci.html>

IMS transactions called via the OTMA/CI interface could eventually participate in a Global Unit of Work under the condition that caller and IMS are located in the same operating system. More information about OTMA can be found in *Open Transaction Manager Access Guide and Reference*, SC26-8743. The following sections on OTMA/CI are cited from the redbook *IMS e-business Connectors: A Guide to IMS Connectivity*, SG24-6514 where you will find more details.

7.3.1 Software requirements for OTMA C/I installation

For IMS Version 6 only: OTMA/CI with basic functionality requires installing APAR PQ17203.

To add asynchronous messaging and extended support to the base OTMA/CI product also requires APAR PQ32398.

In IMS Version 7, OTMA/CI, with extended functionality, is already included.

7.3.2 OTMA/CI initialization

OTMA/CI provides a stand-alone program, DFSYSVIO, that must be run after the MVS IPL to initialize the OTMA/CI. DFSYSVIO invokes DFSYSVC0, one of the OTMA/CI modules.

DFSYSVC0 loads and registers the SVC services by an authorized address space running on the same z/OS image as the application programs accessing it.

You must add an entry in the MVS program properties table (PPT) for the OTMA Callable Interface initialization program. The steps for doing this are:

1. Edit the SCHEDxx member of the SYS1.PARMLIB data set.
2. Add the following entry to the SCHEDxx member:

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

Re-IPL the MVS system or issue the MVS SET SCH= command to make the SCHEDxx changes effective.

For additional reading about updating the program properties table, see *MVS/ESA Initialization and Tuning Reference*.

A sample JCL proc for running DFSYSVIO is shown in Example 7-8:

Example 7-8 Sample JCL for initializing OTMA Callable Interface

```
PROC RGN=3000K,SOUT=A,
//PARM1=
//*
//IEFPROC EXEC PGM=DFSYSVIO,
//REGION=&RGN
//*
//STEPLIB DD DISP=SHR,UNIT=SYSDA,
//DSN=IMSVS.SHWSRESL
//SYSPRINT DD SYSOUT=&SOUT
//SYSUDUMP DD SYSOUT=&SOUT
```

Cross LPAR OTMA/CI communication is also possible over XCF.

The OTMA-related parameters that you must specify in IMS PROCLIB member DFSPBxxx are:

GRNAME=

Specifies the XCF group IMS is to join. The group name is one to eight uppercase alphanumeric characters or other valid characters (#, \$, @). IMS joins the XCF group either during IMS initialization (if OTMA=Y is specified) or as a result of an IMS /START OTMA command. If GRNAME= is not specified and OTMA=N is specified, IMS cannot join the XCF group.

All OTMA clients must know the XCF group name, because any client can create a group. If you specify GRNAME= and OTMA is started, you can use the /DISPLAY OTMA command to display the XCF status. You are not required to define any XCF information.

The XCF member name that IMS uses for the group comes from the name specified for the USERVAR parameter. This USERVAR name is specified in the IMS procedure, in the DFSPBxxx member. If it is not specified anywhere, IMS uses the IMS APPLID1 as the member name.

Important: If you use RACF for security, the IMSXCF.group.member (client member name) must be defined in the RACF FACILITY class.

OTMA=

Specifies whether OTMA is to be enabled. Values are Y, YES, N, NO. The default is NO. If you specify a valid group name for GRNAME=, you can use the /START OTMA command to enable OTMA later, even if you specify OTMA=N during system definition.

7.3.3 OTMA/CI security

A new RACF facility class, IMSXCF.OTMACI, has to be defined for the OTMA/CI to protect XCF groups from any non-authorized caller. When the RACF resource is defined, RACF RACHECK is invoked before OTMA/CI performs an XCF JOIN. This method protects the access to XCF, the XCF group, and the member. This RACF checking is performed only when a non-authorized caller is using OTMA/CI.

Additional security characteristics remain consistent with OTMA in IMS 6.1.

7.3.4 OTMA/CI restrictions

OTMA/CI must be initialized and installed in IMS 6.1 and above, but it can connect to all IMS OTMA releases.

Application program languages other than C and C++ are not currently supported by OTMA/CI. Other languages, such as Cobol, Assembler, and PL/I, are planned to be supported in the future.

All OTMA calls must be made in the same state (PSW key, supervisor, or problem state, authorized or non-authorized) as the `otma_open` call. For example: If you were authorized when you did the `otma_open` call, you must be authorized for all subsequent calls.

The commit-then-send option of IMS OTMA is not supported by OTMA/CI. If IMS generates a commit-then-send output and sends it to an OTMA/CI client, OTMA/CI ignores the output and does not deliver it to the OTMA/CI client.

The resynchronization feature of IMS OTMA is not supported. The IMS command `/SECURE OTMA PROFILE` is currently not supported.

7.3.5 Compiling and link-editing requirements for OTMA/CI

The header file included in the API calling program declares each API invocation and variables used for the invocation. For a C/C++ program using OTMA/CI, the C/C++ header file, `DFSYC0.H`, needs to be included in the C/C++ program.

The object stub, `DFSYCRET`, receives all the API invocations and issues an SVC call to perform the requested function. The object stub needs to be available during the link-editing of the API invoking program. `DFSYCRET` can be found in `SDFSRESL` or `ADFSLOAD` data sets.

7.3.6 Call functions implemented by OTMA/CI

OTMA/CI implements 10 different functions by the following API calls:

- ▶ `otma_create`

Creates storage structures to support communications, but does not establish a connection with IMS.

- ▶ `otma_open`

Establishes a connection with IMS. Issues an `otma_create` prior to establishing an `otma_open` call.

- ▶ `otma_openx`
Provides the same function as `otma_open` API, with an added parameter to specify OTMA Destination Resolution User (DRU) exit name routine and special options.
- ▶ `otma_alloc`
Creates an independent transaction session.
- ▶ `otma_send_receive`
Sends to IMS and passes parameters for receive functions.
- ▶ `otma_send_receivex`
Provides the same function as `otma_send_receive` API, with added parameters to pass OTMA user data.
- ▶ `otma_send_async`
Sends input (transaction or IMS command) only to IMS.
- ▶ `otma_receive_async`
Receives unsolicited or queued output from IMS.
- ▶ `otma_close`
Releases the independent transaction session.
- ▶ `otma_free`
Ends the connection with IMS.

7.3.7 Guidelines for using OTMA/CI with WebSphere Application Server on z/OS

The Callable Interface uses the Open Transaction Manager Access (OTMA) protocol for IMS. As such, there are guidelines and requirements to observe:

- ▶ IMS, Java for z/OS or OS/390, and WebSphere for z/OS must be on the same system in the sysplex. This limitation exists because the OTMA interface allows RRS to coordinate transactions, which requires the client (WebSphere for z/OS) and the IMS server to reside on the same system.
- ▶ Include IMS in the same restart group as WebSphere for z/OS and DB2.
- ▶ A WebSphere for z/OS application server instance acts as an IMS-OTMA client. It must be in the same XCF group to communicate with the IMS-OTMA.

The IMS-OTMA XCF group is one of the parameters required when you define an IMS-OTMA PAA Logical Resource Mapping (LRM) through the SMUI (administration application). The other is the XCF partner name that identifies the specific IMS with which the server communicates. The XCF

partner name is the name specified by the OTMANM parameter in the IMS DFSPBxxx PROCLIB member used for initialization. If no OTMANM parameter is defined, then the name specified by the APPLID1 parameter in the IMS DFSPBxxx member will be used as the default XCF partner name.

- ▶ You must give the control region user ID in the application server instance READ authority to the IMSXCF.OTMACI resource in the RACF FACILITY class.
- ▶ Set the IMS parallel scheduling limit to 0, which means any number of transactions can be scheduled.
- ▶ A transaction in a WebSphere for z/OS application may result in several transactions in IMS (due to several setters and getters within a program). This multiplying effect on transactions affects the number of message processing regions in the DFSMPR job to equal the number of transactions that could result from a WebSphere for z/OS transaction.

If additional WebSphere for z/OS transactions generate additional IMS transactions on the same database, set the number of message processing regions according to the maximum number of transactions that could be generated from all applications.

- ▶ You may only use SendReceive requests when communicating with a target transaction program in IMS. Requests to do Send-only or Receive-only processing with an IMS transaction program are not supported.
- ▶ Details about coding WebSphere for z/OS applications that use IMS, including the setup of the server, are in *WebSphere Application Server V4.01 for z/OS and OS/390 Assembling Java 2 Platform, Enterprise Edition (J2EE) Applications*, SA22-7836. If you want to use the IMS-OTMA Procedural Application Adapter you need to consider the following while setting up the application server.

When defining the LRM for a server, you must choose IMS_OTMA_PAA as the logical resource manager subsystem type and identify the following for the LRM instance connection data:

XCF group name

Fill in the name specified on the GRNAME parameter in the DFSPBxx PROCLIB member used for IMS initialization.

XCF partner name

Fill in the name specified on the OTMANM parameter in the DFSPBxx PROCLIB member used for IMS initialization. Otherwise, use the name specified by the APPLID1 parameter in the DFSPBxx member, which is the default XCF partner name if no OTMANM parameter is defined.

Number of sessions

Specify 1.

TPIPE prefix

Specify a prefix, which must be four characters or less, for the system to use for all transaction pipes required for this LRM. When creating a transaction pipe for this LRM, the system generates a unique transaction pipe name by using this prefix and appending four characters of session-related information.

Note: You cannot have more than one logical resource manager instance with the same XCF group name configured to a given server instance.

For a given server instance, WebSphere for z/OS only connects once to a single IMS member within an IMS XCF group specified by a logical resource manager instance. If you have configured the server instance with another logical resource manager instance that has the same XCF group name, but a different IMS member name, TPIPE name, or number of sessions, initialization of that logical resource manager instance will fail when it attempts to connect to the same IMS XCF group. This is because the server instance will already be a member of the IMS group as a result of the first connection.

Make sure you specify enough members in the XCF data set definitions. You must specify an XCF data set member for each server using the IMS-OTMA Procedural Application Adapter.

7.3.8 When to choose OTMA/CI

The OTMA/CI is a lean, effective and good-performing solution for connecting WebSphere Java applications to IMS. It is a good choice if you need to implement synchronous communication and require good performance.

More information about OTMA/CI can be found at this Web site:

<http://www.software.ibm.com/data/ims/otmaci.html>

and in Appendix D, “OTMA Callable Interface” in *IMS Version 7 OTMA Guide and Reference*, SC26-9434. This document also contains sample code for using the OTMA/CI API for the C language. The sample program contains code for XCF open processing, session allocate processing, executing a command or transaction per invocation parm, session freeing and closing processing.

For more complete documentation on OTMA and its Callable Interface, refer to *IMS e-business Connectors: A Guide to IMS Connectivity*, SG24-6514.

7.4 MQSeries-IMS bridge

This section introduces the IMS bridge. It describes how to customize and control the bridge, as well as which security options are available.

7.4.1 Introduction to the IMS bridge

WebSphere for z/OS provides access to IMS resources through procedural application adapters. Several of these use Open Transaction Manager Access (OTMA) to communicate with IMS, like the MQSeries-IMS bridge. The MQSeries-IMS bridge is an IMS Open Transaction Manager Access (OTMA) client residing in the Queue Manager address space. It is a component of MQSeries for z/OS and OS/390 that allows direct access from MQSeries applications to applications on your IMS system.

An OTMA client (and consequently MQSeries applications using the bridge) can send all types of IMS transactions and some IMS commands to IMS using implicit MQSeries API support. This means that you can re-engineer legacy applications that were controlled by 3270-connected terminals to be controlled by MQSeries messages, without having to rewrite, recompile, or relink them.

In bridge applications there are no MQSeries calls within the IMS application. The application gets its input using a GET UNIQUE (GU) to the IOPCB and sends its output using an INSERT (ISRT) to the IOPCB. MQSeries applications use the IMS header (the MQIIH structure) in the message data to ensure that the applications can execute as they did when driven by nonprogrammable terminals. If you are using an IMS application that processes multi-segment messages, note that all segments should be contained within one MQSeries message. A queue manager can connect to one or more IMS systems, and more than one queue manager can connect to one IMS system. The only restriction is that they must all belong to the same XCF group and must all be in the same sysplex.

The MQSeries classes for Java (used in WebSphere Application Server) enable you to write MQSeries applications including IMS transactions (via the bridge) using the Java programming language. For information on how to enable MQSeries classes for Java for WebSphere Application Server, refer to Chapter 5, “WebSphere MQ” on page 173; about Java programming for MQSeries, refer to *e-business Cookbook for z/OS Volume III: Java Development*, SG24-5980.

7.4.2 Submitting IMS transactions from MQSeries

To submit an IMS transaction that uses the bridge, applications put messages on an MQSeries queue as usual. The messages contain IMS transaction data; they can have an IMS header (the MQIIH structure) or allow the MQSeries-IMS bridge to make assumptions about the data in the message. See the *MQSeries Application Programming Guide* for more information.

MQSeries then puts the message to an IMS queue (it is queued in MQSeries first to enable the use of syncpoints to assure data integrity). The storage class of the MQSeries queue specifies whether the queue is an OTMA queue, and the particular IMS partner to which the message data is sent.

Remote queue managers can also start IMS transactions by writing to these OTMA queues on MQSeries for OS/390.

Data returned from the IMS system is written directly to the MQSeries reply-to queue specified in the message descriptor structure (MQMD).

The OTMA is not "MQ aware," but MQSeries client/requester applications that run IMS transactions via the bridge have to be "IMS aware", that is, knowing the MQIIH header and the IMS application input data structure.

7.4.3 Submitting IMS transactions from a browser

A Tpipe name (an OTMA term) is constructed by the OTMA client and used to tell IMS where to put output from IMS applications. A Tpipe corresponds to an IMS LTERM. IMS commands for OTMA (/DISPLAY and others) refer to Tpipe names. The OTMA client has to provide an OTMA prefix with the message. The full OTMA prefix is described in *IMS/ESA Open Transaction Manager Access Guide and Reference*, SC26-8743.

The MQSeries-IMS bridge builds this prefix based on values in the MQ header of the message, the specific IMS subheader, and some fixed values. The flow of an incoming MQSeries message using the MQSeries-IMS bridge is illustrated in Figure 7-4 on page 227.

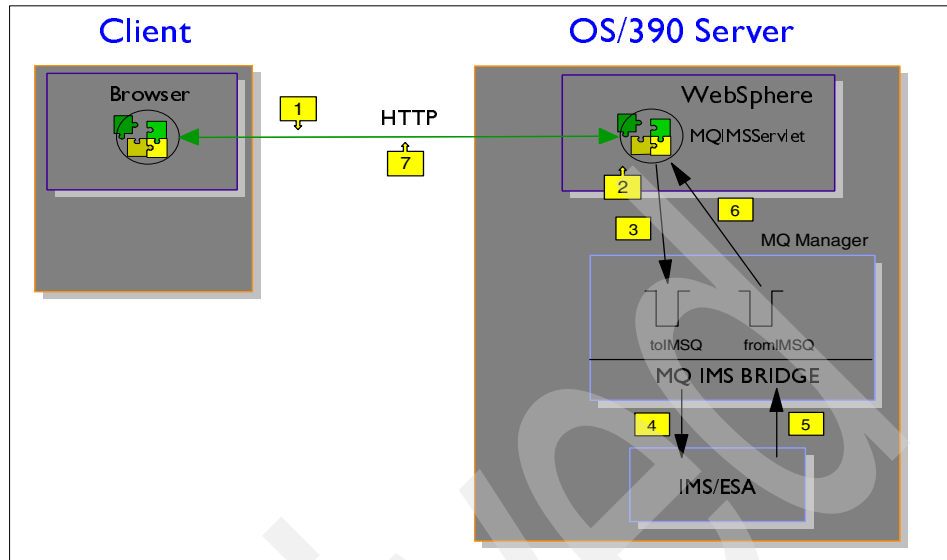


Figure 7-4 Message flow using the MQSeries-IMS bridge

1. The browser requests the servlet to be executed.
2. MQIMSServlet must create a connection to the MQ Manager. Two queues are opened:
 - toIMSQ, which is the queue in which to put the messages to be sent to IMS
 - fromIMSQ, which is the queue to receive the messages back from IMS
3. The message is put into the queue to IMS.
4. The transaction is scheduled by IMS, and a traditional IMS GU call is used to retrieve the message.
5. The IMS application sends a response back to the originating MQSeries application using an IMS ISRT IOPCB call, which routes the message back through OTMA to the destination set as the Reply-to-Queue in the header of the input message.
6. The message is received in the servlet.
7. An HTTP response is given back to the client request.

7.4.4 Customizing the IMS bridge

This section describes what you have to do to customize the MQSeries-IMS bridge, and start sending messages across it to IMS.

1. Define the XCF and OTMA parameters for MQSeries.

This step defines the XCF group and member names for your MQSeries system, and other OTMA parameters. MQSeries and IMS must belong to the same XCF group. Use the OTMACON keyword of the CSQ6SYSP macro to tailor these parameters in the system parameter load module:

<group>	Must be the XCF group name defined by IMS
<member>	Is the XCF member name, by which MQSeries connects to this XCF group. You may code what you like here; the use of the queue manager name is recommended.
<druexit>	Is the name of a program that is to act as the IMS destination resolution exit. This exit has to resolve destinations in case the IMS application issues CHANGE PCB calls. The bridge forwards this name to OTMA.
<age>	Is a time interval that controls security timeout and is forwarded to OTMA, too.
<tpipeprefix>	Is used by the bridge to name tpipes that it uses to send messages to OTMA. It is good practice to code the queue manager name here; it is then easy to know which tpipes are related to which queue manager.

2. Define the XCF and OTMA parameters to IMS.

This step defines the XCF group and member names for the IMS system. IMS and MQSeries must belong to the same XCF group. Add the following parameters to your IMS parameter list, either in your JCL or in member DFSPBxxx in the IMS PROCLIB:

OTMA=Y	This starts OTMA automatically when IMS is started. This is optional. If you specify OTMA=N you can also start OTMA by issuing the IMS command /START OTMA.)
GRNAME=	This gives the XCF group name. This is the same as the group name specified in the storage class definition (see step 3), and in the GROUP parameter of the OTMACON keyword of the CSQ6SYSP macro.
USERVAR=	This gives the XCF member name of the IMS system. This is the same as the member name specified in the storage class definition (see step 3). If you do not specify a name for USERVAR, the value of APPLID1 is used.

3. Tell MQSeries the XCF group and member name of the IMS system. This is specified by the storage class of a queue. If you want to send messages across the MQSeries-IMS bridge, you need to specify this when you define the storage class for the queue. In the storage class, you need to define the XCF group and the member name of the target IMS system. To do this, either

use the MQSeries operations and control panels, or use the MQSC commands as described in *MQSeries Command Reference*, SC33-1369.

4. Set up the security that you require. See 7.4.6, “Security considerations for using MQSeries with IMS” on page 231.

7.4.5 Controlling the IMS bridge and IMS connections

There are no MQSeries commands to control the MQSeries-IMS bridge.

Start the MQSeries bridge by starting OTMA. Either use the IMS command /START OTMA, or start it automatically by specifying OTMA=YES in the IMS system parameters. If OTMA is already started, the bridge starts automatically when MQSeries startup has completed. An MQSeries event message is produced when OTMA is started.

Use the IMS command /STOP OTMA to stop OTMA communication. When this command is issued, an MQSeries event message is produced.

For controlling IMS connections, refer to Appendix E, “Command reference” on page 391.

For more detailed information about these commands, see *IMS/ESA Operator's Reference* for the level of IMS that you are using.

IMS command responses are sent to the terminal from which the command was issued. Authorization to issue IMS commands is based on IMS security.

Resynchronizing the IMS bridge

The IMS bridge is automatically restarted whenever MQSeries, IMS, or OTMA are restarted.

The first task undertaken by the IMS bridge is to resynchronize with IMS. This involves MQSeries and IMS checking sequence numbers on every synchronized Tpipe. A synchronized Tpipe is used when persistent messages are sent to IMS from an MQSeries-IMS bridge queue using commit mode 0 (commit-then-send).

If the bridge is unable to resynchronize with IMS at this time, the IMS sense code is returned in message CSQ2023E and the connection to OTMA is stopped. If the bridge is unable to resynchronize with an individual IMS Tpipe at this time, the IMS sense code is returned in message CSQ2025E and the Tpipe is stopped. If a Tpipe has been cold started, the recoverable sequence numbers are automatically reset to 1.

If the bridge discovers mismatched sequence numbers when resynchronizing with a Tpipe, message CSQ2020E is issued. Use the MQSeries command RESET TPIPE to initiate resynchronization with the IMS Tpipe. You need to provide the XCF group and member name, and the name of the Tpipe; this information is provided by the message.

You can also specify:

- ▶ A new recoverable sequence number to be set in the Tpipe for messages sent by MQSeries, and to be set as the partners receive sequence number. If you do not specify this, the partners receive sequence number is set to the current MQSeries send sequence number.
- ▶ A new recoverable sequence number to be set in the Tpipe for messages received by MQSeries, and to be set as the partners send sequence number. If you do not specify this, the partners send sequence number is set to the current MQSeries receive sequence number.

If there is an unresolved unit of recovery associated with the Tpipe, this is also notified in the message. Use the RESET TPIPE MQSeries command to specify whether to commit it or back it out. If you commit the unit of recovery, the batch of messages has already been sent to IMS, and is deleted from the bridge queue. If you back the unit of recovery out, the messages are returned to the bridge queue, to be subsequently sent to IMS.

Commit mode 1 (send-then-commit) Tpipes are not synchronized.

In IMS, commit mode 1 (CM1) transactions send their output replies before syncpoint.

It is possible that a CM1 transaction is unable to send its reply because:

- ▶ The Tpipe on which the reply is to be sent is stopped.
- ▶ OTMA is stopped.
- ▶ The OTMA client (that is, MQSeries) has gone away.
- ▶ The reply-to queue and dead-letter queue are unavailable.

For all of the above reasons, the IMS application sending the message will pseudo-abend with code U0119. The IMS transaction and program are not stopped in this case.

These reasons often prevent messages being sent into IMS, as well as replies being delivered from IMS. A U0119 abend can occur if:

- ▶ The Tpipe, or OTMA, or MQSeries are stopped while the message is in IMS.
- ▶ IMS replies on a different Tpipe to the incoming message, and that Tpipe is stopped.

- ▶ IMS replies to a different OTMA client, and that client is unavailable.

Whenever a U0119 abend occurs, both the incoming message to IMS and the reply messages to MQSeries are lost. If the output of a CM0 transaction cannot be delivered for any of the above reasons, it is queued on the Tpipe within IMS.

7.4.6 Security considerations for using MQSeries with IMS

If you are using RACF to protect resources in the OPERCMDS class, ensure that your MQSeries system has authority to issue the MODIFY command to any IMS system to which it can connect.

There are four aspects you should consider when deciding your security requirements for the IMS bridge, as follows:

- ▶ What security authorization is needed to connect MQSeries to IMS
- ▶ How much security checking is performed on applications using the bridge to access IMS
- ▶ Which IMS resources these applications are allowed to use
- ▶ What authority is to be used for messages that are put and got by the bridge

When you define your security requirements for the IMS bridge, consider the following:

- ▶ Messages passing across the bridge might have originated from applications on platforms that do not offer strong security features.
- ▶ Messages passing across the bridge might have originated from applications that are not controlled by the same enterprise or organization.

Connecting to IMS

The IMS bridge is an OTMA client. The connection to IMS operates under the user ID of the MQSeries for OS/390 address space. This is normally defined as a member of the started task group. This user ID must be granted access to the OTMA group (unless the /SECURE OTMA setting is NONE).

To do this, define the following profile in the FACILITY class:

```
IMSXCF.<xcfgname>.<mqxcfmname>
```

where xcfgname is the XCF group name and mqxcfmname is the XCF member name of MQSeries.

You must give your MQSeries subsystem user ID read access to this profile.

Note: If you change the authorities in the FACILITY class, you must issue the RACF command SETROPTS RACLIST(FACILITY) REFRESH to activate the changes. If profile hlq.NO.SUBSYS.SECURITY exists in the MQADMIN class, no user ID will be passed to IMS and the connection will fail unless the /SECURE OTMA setting is NONE.

Application access control

For each IMS system that the IMS bridge connects to, you can define the following RACF profile in the FACILITY class to determine how much security checking is performed for each message passed to the IMS system:

```
IMSXCF.<xcfgname>.<imsxcfmname>
```

where `xcfgname` is the XCF group name and `imsxcfmname` is the XCF member name for IMS. (You need to define a separate profile for each IMS system.)

The access level you allow for the MQSeries subsystem user ID in this profile is returned to MQSeries when the IMS bridge connects to IMS, and indicates the level of security that is required on subsequent transactions. For subsequent transactions, MQSeries requests the appropriate services from RACF and, where the user ID is authorized, passes the message to IMS.

OTMA does not support the IMS /SIGN command; however, MQSeries allows you to set the access checking for each message to enable implementation of the necessary level of control.

See *MQSeries System Setup Guide*, SC34-5651, for further details on access level information.

Security checking on IMS

Each MQSeries message that passes across the bridge contains the following security information:

- ▶ A user ID contained in the UserIdentification field of the MQMD structure
- ▶ The security scope contained in the SecurityScope field of the MQIIH structure (if the MQIIH structure is present)
- ▶ A Utoken (unless the MQSeries subsystem has CONTROL or ALTER access to the relevant IMSXCF.xcfgname.imsname profile)

The security checks made depend on the setting by the IMS command /SECURE OTMA. See Appendix E, “Command reference” on page 391 for more details.

Security checking done by the bridge

When the bridge puts or gets a message, the following authorities are used:

- ▶ Getting a message from the bridge queue: No security checks are performed.
- ▶ Putting an exception, or COA report message: Uses the authority of the user ID in the UserIdentifier field of the MQMD structure.
- ▶ Putting a reply message: Uses the authority of the user ID in the UserIdentifier field of the MQMD structure of the original message.
- ▶ Putting a message to the dead-letter queue: No security checks are performed.

Note: If you change the MQSeries class profiles, you must issue the MQSeries command `REFRESH SECURITY(*)` to activate the changes. If you change the authority of a user, you must issue the MQSeries command `RVERIFY SECURITY` to activate the change.

Using RACF passtickets in the IMS header

If you want to use a passticket instead of a password in the IMS header (MQIIH), you should use an application name as if you were creating a passticket for an OS/390 batch job. That is, the APPL field should be of the form `MVSxxxx`, where `xxxx` is the SMFID of the OS/390 system on which the target queue manager runs.

A passticket is built from a user ID, the target application name (APPL), and a secret key. The key is an 8-byte value containing uppercase alphabetic and numeric characters. It can be used only once, and is valid for a 20-minute period. For full information about passtickets, see *Security Server (RACF) Security Administrator's Guide*.

Passtickets in IMS headers are given to RACF by MQSeries, not IMS.

7.4.7 When to use the MQSeries-IMS bridge

You might want to use the MQSeries-IMS bridge to connect from a WebSphere for z/OS Java application to IMS if the IMS application is a connection-less program-to-program communication where programs send and retrieve messages from queues. The network and programs do not have to be available at the same time (asynchronous). The queuing mechanism ensures that the messages get delivered.

For more details on how to write a Java program accessing MQSeries, refer to *Java Programming Guide for OS/390* and *Volume III* of this cookbook series. More information can also be found in *e-business Application Solutions on OS/390 Using Java: Volume 1*, SG24-5342.

7.5 IMS Connect

We start with an overview of IMS Connect and describe how to install and configure it considering circumstances and prerequisites.

We go into details about customizing IMS Connect via user message exits and explain the Local option support.

To be able to confirm the installation and configuration of IMS Connect, we provide descriptions of how to run the Installation Verification Program and how to use the sample Java client provided.

7.5.1 IMS Connect overview

IMS Connect is a separately priced program product that provides connectivity to IMS TM from any TCP/IP client, so you can access IMS TM resources from a TCP/IP network. The current version of IMS Connect, at the time of writing, was Version 1 Release 1 (Program Number 5655-E51). Since then, IMS Connect version 1.2 has become available, which supports the official JCA-compliant connectors.

Connect (formerly known as ITOC or IMS TOC) provides enhanced communication linkages between remote workstations and IMS using the Open Transaction Manager Access (OTMA) facility introduced in IMS Version 5.

IMS Connect provides improved performance and access for users of TCP/IP in accessing IMS applications and data. It supports multiple TCP/IP clients accessing multiple datastore resources (IMS systems). It also provides access from TCP/IP-supported environments into IMS on z/OS.

Figure 7-5 on page 235 shows a general view of the IMS Connect environment. The IMS Connect architecture is designed to support any TCP/IP client communicating with socket calls. IMS Connect also supports the TCP/IP clients using the IMS Connector for Java, which is a collection of Java beans, that enable a Java application to communicate data requests, via TCP/IP and the IMS Connect to IMS.

TCP/IP application programmers can choose to write programs using the direct sockets interface or they can take advantage of code-generating packages such as the IMS Connector for Java, which ships with the VisualAge for Java Enterprise Edition toolkit.

These programs, whether they are user-written or generated, can communicate with IMS Connect in a variety of ways. The IMS Connector for Java, for example, converts the application input data into OTMA message format before shipping the data stream to IMS Connect and, on the reply, takes the OTMA message format and converts it to the output data stream that can be understood by the application. User-written programs can also do this. Those with no knowledge of OTMA can take advantage of a defined IMS Connect application protocol (set of architected headers) that relies on IMS Connect to deal with OTMA message format on behalf of the TCP/IP program.

You can also utilize the Local Option support for connections between IMS Connector for Java and IMS Connect. Local Option provides non-socket access to IMS transactions and data from z/OS or OS/390 WebSphere Application servers.

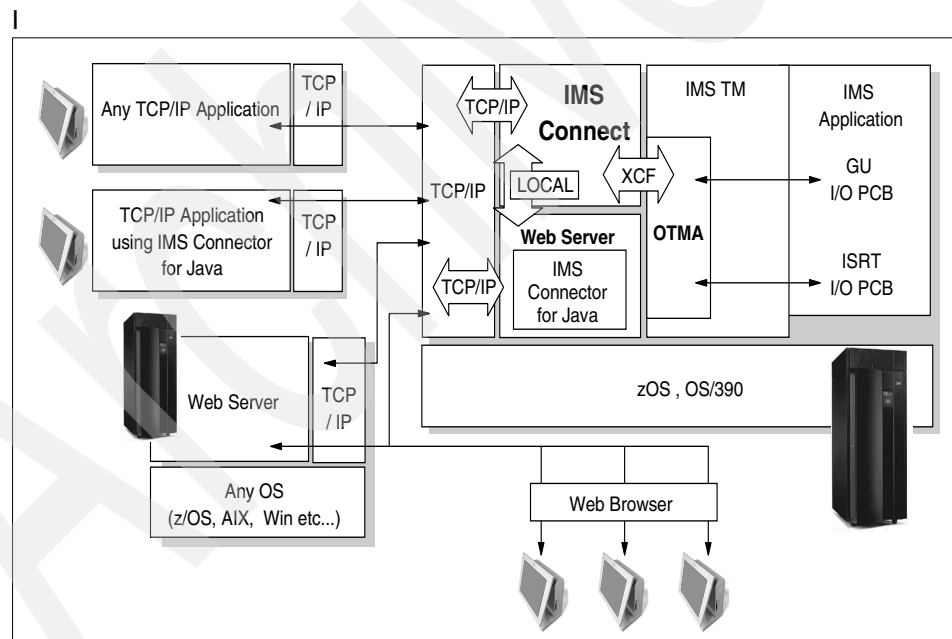


Figure 7-5 IMS Connect system overview

IMS Connect executes in its own address space and consists of three core components:

- TCP/IP communication component (CCC)

This component processes communication between TCP/IP clients and IMS Connect.

- Datastore communication component (DCC)

This component handles communication between IMS Connect and the IMS datastores.

- Command component (CMD)

This component processes commands that are received from the MVS console operator to "work on" IMS Connect resources.

In addition to these core components, IMS Connect uses a communication driver facility to isolate the core components from the communication software, a TCP/IP driver to communicate with TCP/IP clients, and an IMS OTMA driver to communicate with the datastores. A Local driver is used to communicate with clients by using the Program Call (PC) for local connection between IMS Connect and Web Server using IMS Connector for Java. Communication between components takes place via the call interface service, which provides the encapsulation and isolation of structures between the components. Each IMS Connect component provides its own set of functions, which are registered with the call interface. Because the IMS OTMA Communications protocol is based on XCF from MVS, IMS Connect can communicate with any IMS Transaction Manager system on any MVS system of an MVS sysplex.

7.5.2 Installing IMS Connect V7

Release 7.1 of IMS contains IMS Connect as a separately priced component, FMID(HMK7707), which is installed using SMP/E.

Note: IMS Connect is available for IMS V5, V6 and V7 and is a replacement of the IMS TOC connector. IMS TOC is not available any more and support was withdrawn in spring 2001.

Most of the function in IMS Connect runs with IMS V5, V6 and V7, though some function (Asynchronous output, Local and Unicode support) only works with IMS V7.

For the latest software prerequisites, refer to the IMS Connect product documentation *The IMS Connect Guide and Reference*, SC27-0946, which you can find in the library on this Web site:

<http://www.ibm.com/software/data/ims/>

This redbook does not go into any detail about the SMP/E installation of IMS Connect V7, since this is “business as usual” for S/390 and zSeries customers. The IMS Connect distribution tape has a sample JCL for the installation, which you can modify to your own environment. Refer to *Program Directory for IBM IMS Connect for OS/390*, GI10-8275. The following sections focus on customizing IMS Connect.

7.5.3 Considerations for the IMS Connect configuration

The TCP/IP client application requests a connection by specifying the host DNS name (resolves to the IP address of the target host), the IMS Connect port number and the target datastore ID. The datastore ID identifies a **DATASTORE** statement in the IMS Connect configuration file, which directs the request to the specific IMS system. Although the datastore ID and the target name (TMEMBER) of the IMS system can be different, specifying them as the same name can make it easier to identify the actual IMS that will process the transaction.

The IMS Connect configuration file consists of three types of statements:

- ▶ HWS, which defines the IMS Connect name.
- ▶ TCPIP, which provides the information for the TCP/IP connectivity (ports and user message exits).
- ▶ Datastore, which identifies a target IMS system.

IMS Connect can be configured to allow for your availability, capacity, security and performance requirements. One IMS Connect can support one or multiple IMS systems. This configuration results in high performance when several workloads for multiple IMS systems can be processed parallel-supported by one IMS Connect. It also allows a message to be routed to a different IMS system in the event of an unavailable target IMS system. This assumes that all IMS systems can receive any transaction request, e.g., the IMS definitions are cloned along with data sharing and/or shared queues, or MSC is available to route the messages to the correct IMS.

Multiple IMS Connects can be used to access one IMS system. This type of configuration could be used for security reasons or to support different message exit actions, for example. IMS Connects placed on different OS/390 images could be used as backups for each other assuming a sysplex environment.

7.5.4 Configuring IMS Connect

This section describes how to prepare the environment for IMS Connect. To use the information provided here, you need a working knowledge of IMS transaction processing, RACF, IMS OTMA, and TCP/IP.

IMS Connect supports communication between one or more TCP/IP clients and IMS systems. IMS Connect uses TCP/IP for communication with clients and IMS OTMA for communication with IMS. It also provides a mechanism to start or stop TCP/IP clients or datastores with the use of commands.

You can configure multiple IMS systems on multiple MVS systems and distribute the client request to the IMS datastores. To configure IMS Connect, perform the following actions:

1. Create an IMS Connect job or started task.
2. Authorize the IMS Connect load library with the Application Program Facility (APF).
3. Update the MVS Program Properties Table (PPT) to run IMS in authorized supervisor state and in key 7.
4. Create an IMS Connect configuration member to hold the configuration statements for initialization.
5. Configure the Base Primitive Environment (BPE).

Then you need to customize IMS Connect by installing user message exits and modifying them if needed.

Creating an IMS Connect job or started task

You can use the sample JCL for the IMS Connect started task procedure. You could also run the IMS Connect as an MVS job (Example 7-9).

Example 7-9 Sample job for starting IMS Connect

```
//HWS01 JOB MSGLEVEL=1,TIME=1440,CLASS=Y,USERID=&USERID
//HWS01 EXEC HWS710A,SOUT=A
//*****
//HWS710A PROC RGN=4096K,SOUT=S,
//          BPECFG=BPECFGHT,
//          HWSCFG=HWSCFG00
//*****
//* BRING UP IMS CONNECT *
//*****
//STEP1 EXEC PGM=HWSHWS00,REGION=&RGN,TIME=1440,
//          PARM='BPECFG=&BPECFG,HWSCFG=&HWSCFG'
//STEPLIB DD DSN=IMS710A.SHWSRESL,DISP=SHR
//          DD DSN=IMS710A.SDFSRESL,DISP=SHR
```



```
//PROCLIB DD DSN=IMS710A.PROCLIB,DISP=SHR
//SYSTCPD DD DSN=TCPIPOE.SC59.TCPPARMS(TCPDATA),DISP=SHR
//SYSPRINT DD SYSOUT=&SOUT
//SYSUDUMP DD SYSOUT=&SOUT
/*****
/* TRACE DS MUST BE FB, LRECL=1440, BLKSIZE=14400 *
/*****
//HWSRCORD DD DSN=IMS710A.TRACE,DISP=SHR
```

SHWSRESL is the resident library (RESLIB) for IMS Connect.

If the IMS datastore is RACF-protected, start IMS Connect as a job with the JOB card specifying a valid USERID in order to make the connection from IMS Connect to IMS, or you can use the RACF-started procedure table.

The USERID=&userid parameter specified in the JOB card of the IMS Connect job JCL is used as the security vehicle to ensure IMS Connect access to IMS. The USERID must have READ access to IMSXCF.group.member. IMS OTMA provides security for the IMS XCF connection by defining and permitting IMSXCF.group.member in the RACF FACILITY class. For details, see *IMS Version 7 Open Transaction Manager Access Guide*, SC26-9434.

Important: To configure security for the local option using RACF, you must add HWS.ICON_NAME as the SAF facility class name (whether you configured security with the IMS Connect configuration member or SETRACF command). ICON_NAME is how IMS Connect is defined in the ID parameter of the HWS statement in the IMS Connect configuration member. The resource that must access IMS Connect is WebSphere Application Server, and UPDATE authority is required to update the RACF profile.

Authorizing IMS Connect and BPE to the APF

The resident library SHWSRESL in which the IMS Connect (HWS...) and BPE (BPE...) modules reside, must be authorized to the APF. Create and run a JCL job that authorizes this RESLIB to the APF.

Updating the MVS PPT

Because IMS Connect is executed in supervisor state and key 7, add an entry for it in the MVS Program Properties Table (PPT), as follows:

1. Edit the SCHEDxx member of the SYS1.PARMLIB data set.
2. Add the following in the MVS PPT:

```
PPT PGMNAME(HWSHWS00) /* PROGRAM NAME = HWSHWS00 */
      CANCEL          /* PROGRAM CAN BE CANCELED */
      KEY(7)          /* PROTECT KEY ASSIGNED IS 7 */
```

```

SWAP      /* PROGRAM IS SWAPPABLE          */
NOPRIV    /* PROGRAM IS NOT PRIVILEGED      */
DSI       /* REQUIRES DATA SET INTEGRITY   */
PASS      /* CANNOT BYPASS PASSWORD PROTECTION */
SYST      /* PROGRAM IS A SYSTEM TASK       */
AFF(NONE) /* NO CPU AFFINITY                        */
NOPREF    /* NO PREFERRED STORAGE FRAMES    */

```

If you are using the local option for client communications, either by itself or with TCP/IP communications, change the SWAP entry to NOSWAP in the properties above.

3. To make the changes effective, either re-IPL your z/OS system or issue the MVS SET SCH= command to make the update dynamically.

Creating the IMS Connect configuration member

Specify the environment for IMS Connect as a member in your IMS PROCLIB data set. We used IMS710A.PROCLIB(HWSCFG00).

IMS Connect uses the information it retrieves from the member to establish communication with IMS and TCP/IP. You can define several configuration members in the PDS to choose from during IMS Connect startup.

For the IMS Connect configuration, specify the member name to use in the HWSCFG= parameter of the IMS Connect startup JCL.

```

//HWS      PROC  RGN=4096K,SOUT=A,
//          HWSCFG=HWSCFG00

```

This member is located in the IMS PROCLIB, referred to by the PROCLIB DD statement in the startup JCL for IMS Connect. In our case, we used the following:

```

//PROCLIB DD  DSN=IMS710A.PROCLIB,DISP=SHR

```

Our HWSCFG00 member contains the following three lines, which specify the statement parameters. They define the way in which IMS Connect is to communicate with TCP/IP and IMS OTMA in the IMS Connect configuration member:

```

HWS (ID=HWS710A,RACF=N,XIBAREA=20)
TCPIP (HOSTNAME=TCPIP0E,PORTID=(6001),EXIT=(HWSMPL0))
DATASTORE (ID=IMSB,GROUP=IMS71XCF,MEMBER=HWS710A,TMEMBER=SCSIMS7A)

```

In the following IMS Connect configuration member, the IMS Connect ID is defined as HWS. The IMS Connect is configured to include the ports defined for TCP/IP communications and the IMS Connect group and member names for communication with IMS.

The TCP/IP configuration defines the HOSTNAME as MVSTCPIP, the RACFID as RACFID, the PORTID as 9999, and the EXIT as HWSSMPL0.

The datastore configuration defines the ID as IMS, the GROUP as XCFCGROUP, the MEMBER as HWSMEM, and the TMEMBER as IMSMEM.

Example 7-10 shows a sample IMS Connect configuration file, for no RACF security, port ID of 9999, and an XCF group name of IMSGROUP.

Example 7-10 Sample IMS Connect configuration file

```
MVS TCP/IP           : MVSTCPIP, PORT:9999
IMS Connect ID       : HWS
IMS Connect XCF Member Name : HWSMEM
XCF Group Name       : XCFCGROUP
DATASTORE XCF Member Name : IMSMEM
DATASTORE ID        : IMS

*****
* Example: IMS Connect Configuration File *
*****
HWS (ID=HWS,RACF=N,XIBAREA=20)
TCP/IP (HOSTNAME=MVSTCPIP,RACFID=RACFID,PORTID=(9999),MAXSOC=2000,TIMEOUT=8888,
      EXIT=(HWSSMPL0))
DATASTORE (ID=IMS,GROUP=XCFCGROUP,MEMBER=HWSMEM,TEMBER=IMSMEM,DRU=HWSYDRU0)
```

Configuring the Base Primitive Environment (BPE)

Base Primitive Environment (BPE) is a common system service component base upon which IMS Connect is built. It provides the following base services:

- ▶ Environment
- ▶ Dispatch
- ▶ Storage
- ▶ Serialization
- ▶ Tracing

IMS Connect acts as a concurrent server program that supports multi-threading for transactional requests. For each port ID defined in the HWSCFG configuration member, IMS Connect creates a port thread. When a request message arrives, a socket thread is generated to process the request. For each datastore defined in the HWSCFG configuration member, IMS Connect creates a scheduling thread. It then creates a receive thread to receive the input message from the socket thread and send it to IMS OTMA, and a transmit thread to send back the output message to the socket thread.

Generally, you do not need to work with the BPE. However, your IBM service representative could request that you change the default settings for certain BPE functions such as storage management, internal tracing, dispatching, and other system service functions. IMS Connect supplies a configuration data set member for BPE system service functions that you can modify.

To change the settings, you can modify a member of the PDS that the PROCLIB DD card specifies. You select the member to use in the BPECFG= parameter of the IMS Connect startup JCL.

```
EXEC HWSHWS00,PARM='BPECFG=BPECFGHW'
```

The following example shows a BPE configuration member and the statement parameters. In Example 7-11, the TCP/IP to IMS trace includes entries for all component events.

Example 7-11 BPE configuration member

```
*****
* CONFIGURATION FILE FOR BPE WITH IMS CONNECT *
*****
LANG=ENU                /* LANGUAGE FOR MESSAGES */
                        /* (ENU = U.S. ENGLISH) */
/* SEE THE IMS VERSION 7 COMMON QUEUE SERVER AND BASE PRIMITIVE /*
/* ENVIRONMENT GUIDE AND REFERENCE FOR EXAMPLES OF BPE SYSTEM TRACES /*
#
# DEFINITIONS FOR IMS CONNECT BPE SYSTEM TRACES
#
TRCLEV=(*,LOW,BPE)      /* DEFAULT TRACES TO LOW */
TRCLEV=(AWE,HIGH,BPE)   /* AWE SERVER TRACE ON HIGH*/
TRCLEV=(CBS,MEDIUM,BPE) /* CTRL BLK SRVCS TRC ON MEDIUM*/
TRCLEV=(DISP,HIGH,BPE,PAGES=12) /* DISPATCHER TRACE ON HIGH*/
                        /* WITH 12 PAGES=48KB*/
#
# DEFINITIONS FOR IMS CONNECT TRACES
#
TRCLEV=(*,HIGH,HWS)     /* DEFAULT ALL IMS CONNECT TRACES TO HIGH */
TRCLEV=(HWSI,MEDIUM,HWS) /* BUT RUN IMS CONNECT TO IMS OTMA TRACE... */
TRCLEV=(HWSW,MEDIUM,HWS) /* AND SERVER TO IMS CONNECT TRACE AT MEDIUM */
```

As shown in the previous example of a BPE configuration member, you can specify parameters for LANG and TRCLEV.

Recommendation: Avoid coding statements in the IMS Connect BPE configuration member that specify definitions for the same resources more than one time. For example, multiple TRCLEV statements for the same trace table type. BPE uses the last statement it encounters in the member. Any values that are specified on earlier duplicate statements are ignored. The message BPE0017I is issued for each duplicate found.

For complete information and instructions on changing the BPE configuration member, BPE parameters, and creating trace tables, see Part 2, “Base Primitive Guide and Reference” in *IMS Version 7 Common Queue Server and Base Primitive Environment Guide and Reference*, SC26-9426.

7.5.5 Customizing IMS Connect

IMS Connect receives data from a TCP/IP client, performs basic editing and translation, invokes security, and prepares the message in the OTMA format that IMS understands. Response messages from IMS are also prepared into a format that the TCP/IP client understands. This functionality is done in the IMS Connect User Message Exits. A base set of Message Exits are delivered with the product. There is no requirement to write or alter any of the exits unless there is a perceived need for customization.

You can customize various aspects of IMS Connect to fit your specific business needs by modifying any or all of the exits that IMS Connect provides. These exits are described in Table 7-1.

Table 7-1 IMS Connect (V7) user exits

Exit name	Type	Associated Macro files	Purpose
HWSIMSO0 HWSIMSO1	User Message exits	N/A	Replaces the EZAIMSO0 exit that TCP/IP previously provided. HWSIMSO1 replaces HWSIMSO0.

Exit name	Type	Associated Macro files	Purpose
HWSSMPL0 HWSSMPL1	User Message exit for non-IMS Connector for Java clients only.	HWSIMSCB HWSIMSEA HWSEXPXM HWSOMPFX	Enables users to use their own message formats. The sample program HWSSMPL0 returns the MOD name to the client for message formatting, if the name is supplied by the IMS transaction. HWSSMPL1 also passes a fullword length field preceding the message.
HWSJAVA0	User Message exit for IMS Connector for Java clients only.	HWSIMSCB HWSIMSEA HWSEXPXM HWSOMPFX	Enables IMS Connector for Java users to edit their messages and to do their own security checking.
HWSYDRU0 (new for IMS Connect)	Sample OTMA DRU exit	N/A	A DRU exit is required to support IMS Connect's asynchronous output features.
HWSUNIT (new for IMS Connect)	User Initialization exit	HWSXIB HWSXIBDS	Enables users to perform their own processing during IMS Connect initialization and termination.

Important: Do not define the exits HWSJAVA0, HWSYDRU0 nor HWSUNIT in the IMS Connect configuration file, specifically on the EXIT= parameter of the TCP/IP statement. HWSJAVA0 is dynamically loaded by IMS Connect; HWSYDRU0 is loaded by OTMA during IMS Connect initialization; HWSUNIT is dynamically loaded by IMS Connect during IMS Connect initialization.

Installing and modifying user message exits

The exits HWSJAVA0, HWSUNIT, HWSSMPL0, HWSSMPL1, and HWSYDRU0 are installed into AHWSSRC (the source library) during the IMS Connect installation process. They were not automatically installed into your IMS Connect load library and were not link-edited into your IMS Connect resource library (SHWSRESL). This is to ensure that subsequent IMS Connect installations and SMP/E maintenance do not replace your copies of these exits in either the load library or in your current IMS Connect resource library.

You *must* install the following two exits into your IMS Connect resource library, regardless of whether you intend to customize them, because IMS Connect automatically loads these exits when it executes:

HWSJAVA0
HWSUINIT

You need to install the HWSSMPL0, HWSSMPL1, and HWSYDRU0 exits only if you want to use the features that they support.

To install an exit into the IMS Connect resource library, compile and link-edit the exit into IMS Connect resource library SHWSRESL. Example 7-12 shows sample JCL to assist you when link-editing and compiling these exits. Note the following data set/member names:

<hws>.SHWSMAC Your IMS Connect macro library.
<hws>.SHWSSRC Your IMS Connect source library.
<hws>.SHWSRESL Your IMS Connect resource library.
exitname User exit routine name (HWSJAVA0 or HWSUINIT).

Example 7-12 User message exit and user INIT exit link-editing sample JCL

```
//HWSINIT JOB (ACTINF01), 'PGMRNAME',  
//CLASS=A,MSGCLASS=Z,MSGLEVEL=(1,1),REGION=4M  
//UINIT1 EXEC PGM=ASMA90,REGION=32M,  
//PARM='DECK,NOOBJECT,NOLIST,SIZE(MAX,ABOVE),FLAG(NOPUSH) '  
//SYSLIB DD DSN=<hws>.SHWSMAC,DISP=SHR  
//DD DSN=SYS1.MACLIB,DISP=SHR  
//SYSPUNCH DD UNIT=SYSVIO,DISP=(,PASS),SPACE=(TRK,(1,1,1)),DSN=&&TEXT(exitname)  
//SYSPRINT DD SYSOUT=*,DCB=(BLKSIZE=605),SPACE=(605,(100,50),RLSE,,ROUND)  
//SYSUT1 DD UNIT=SYSDA,DISP=(,DELETE),DCB=BLKSIZE=13024,SPACE=(CYL,(16,15))  
//SYSIN DD DSN=<hws>.SHWSSRC(exitname),DISP=SHR  
//UINIT2 EXEC PGM=IEWL,  
//PARM='SIZE=(880K,64K),RENT,REFR,NCAL,LET,XREF,LIST,TEST'  
//SYSPRINT DD SYSOUT=A  
//SYSLMOD DD DSN=<hws>.SHWSRESL,DISP=SHR  
//SYSUT1 DD UNIT=SYSVIO,DISP=(,DELETE),SPACE=(CYL,(10,1),RLSE)  
//TEXT DD UNIT=SYSVIO,DISP=(OLD,DELETE),DSN=&&TEXT  
//SYSLIN DD *  
INCLUDE TEXT(exitname)  
ENTRY exitname  
NAME exitname(R)  
/*
```

For HWSSMPL0, HWSSMPL1, HWSJAVA0 and HWSUINIT use the IMS Connect resource library SHWSRESL. For HWSYDRU0 use the IMS resource library SDFSRESL.

Important: Compile and link-edit the HWSYDRU0 exit into your IMS resource library SDFSRESL, not the IMS Connect resource library SHWSRESL. Otherwise, OTMA will not be able to use the HWSYDRU0 exit.

Table 7-2 describes the link-editing requirements for installing each of the five exits.

Table 7-2 Link-editing requirements

Exit Name	Installation Required	Linkedit Requirements
HWSSMPL0 HWSSMPL1	No	You can link-edit this exit using its given name or you can supply your own name. Specify the name used for the exit on the TCPIP statement for EXIT= in the IMS Connect configuration file.
HWSJAVA0	Yes	Link-edit this exit using its given name.
HWSUINIT	Yes	Link-edit this exit using its given name.
HWSYDRU0	No	You can link-edit this exit using its given name or you can supply your own name. Specify the name used for the exit on the DATASTORE statement for DRU= in the IMS Connect configuration file.

All five of these exits can be used as shipped or can be modified (customized).

Important: Before you modify an exit, make a copy of the exit and rename the copy so you can restore it if there is a problem.

For more details on how to modify user exits, refer to *IMS Connect Guide and Reference V1R2*, SC27-0946-01 and *IMS e-business Connectors: A Guide to IMS Connectivity*, SG24-6514.

7.5.6 IMS Connect Local Option support

IBM introduced Local Option, a non-TCP/IP communication path between IMS Connector for Java and IMS Connect. Local Option, as its name implies, can be used when both the application or servlet that is running IMS Connector for Java and IMS Connect reside in the same z/OS image, as shown in Figure 7-6 on page 247.

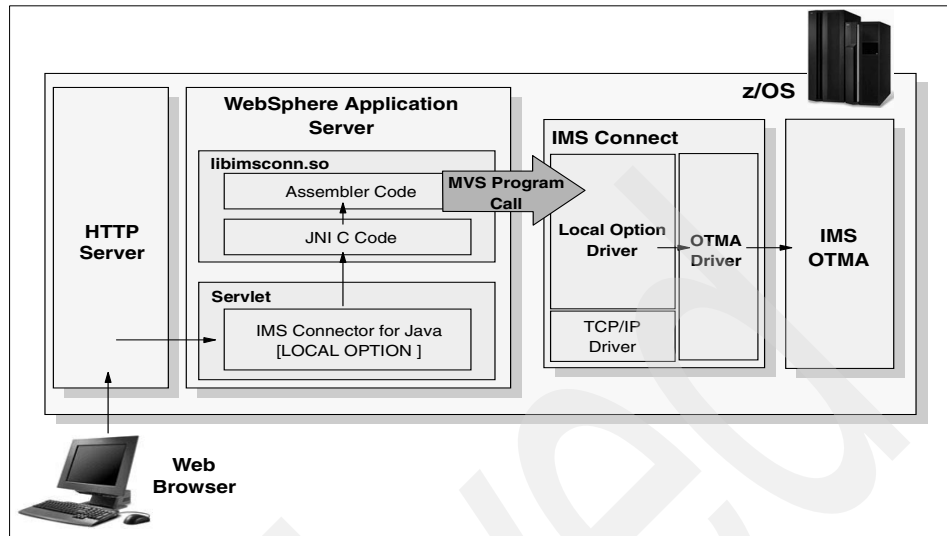


Figure 7-6 IMS Connect Local Option support overview

Functional overview

A simple approach was taken for implementing Local Option in order to make it as easy as possible to retrofit existing customer-written IMS Connector for Java applications. Any newly written or existing IMS Connector for Java application can be adapted for use with Local Option by simply changing the host name parameter that the application uses.

If the host name is of the form `localhost: ims_connect_name`, Local Option is used for all communications between IMS Connector for Java and the specified IMS Connect instance. If the host name is anything other than the first ten characters of `localhost`, IMS Connector for Java opens a TCP/IP connection to that host name. If the host name property is made externally available during application development, the host name can be specified at runtime and therefore changed dynamically between Local Option and TCP/IP as the need arises.

In IMS Connector for Java, the architectural approach that IBM used to implement Local Option was to wrap existing TCP/IP-specific code in conditional blocks and to add separate blocks for Local Option-specific code represented by the Local Adapter block. The Local Adapter invokes JNI (Java Native Interface) calls, which transfer control from the Java code to native C code, which in turn makes calls to native assembler code routines. Both the C code and the assembler routines are packaged in `libimsconn.so`, which is an OS/390 UNIX System Services shared library that is shipped with IMS Connector for Java. The

assembler routines execute in the same address space where the Java Virtual Machine (JVM) that runs IMS Connector for Java is running. These assembler routines execute the OS/390 program calls to communicate with IMS Connect, which is running in its own address space.

Local Option in IMS Connect has been implemented by adding a Local Option driver and PC request handler routines, one for each type of request. Local Option is defined to IMS Connect as a "port." The structure of the LOCAL "port" is very similar to the structure of a TCP/IP port, except that this special port is handled by the Local Option driver rather than by the TCP/IP driver.

If a LOCAL port has been defined in the IMS Connect configuration member, this LOCAL port is activated automatically during IMS Connect initialization. Also, during IMS Connect initialization, the Local Option driver is initialized and registered with OS/390. After IMS Connect initialization has completed successfully, IMS Connect is ready to accept Local Option connections.

Software required

The first task in the setup of Local Option involves ensuring that the proper levels of IMS Connect and IMS Connector for Java are installed and configured correctly.

The minimum software requirements for using Local Option are:

- ▶ IMS Version 7
- ▶ IMS Connect Version 1.1 with APAR PQ45057 (Local Option)
- ▶ OS/390 Version 2, Release 6 or higher including WebSphere and WebSphere APAR PQ25544 (for connection pooling)
- ▶ The IBM JDK for OS/390 Version 1.2.2

We strongly encourage you to ensure that all of this software is at the current service level.

Setting up IMS Connect

The setup of IMS Connect for Local Option consists of four steps:

- ▶ Apply Local Option enabling APAR PQ45057 to IMS Connect Version 1.1.
- ▶ Add a "LOCAL" port to the TCPIP statement in the IMS Connect configuration member (see "Creating the IMS Connect configuration member" on page 240).
- ▶ Ensure that IMS Connect is set to NOSWAP in the MVS PPT. This is a new IMS Connect requirement, specifically for Local Option (see "Updating the MVS PPT" on page 239).

- Add HWS.ims_connect_name as the SAF facility class name to configure security for the local option using RACF (whether you configured security with the IMS Connect configuration member or the SETRACF command). IMSConnect_name is how IMS Connect is defined in the ID parameter of the HWS statement in the IMS Connect configuration member. The resource that must access IMS Connect is the WebSphere Application Server, and UPDATE authority is required to update the RACF profile.

After setup has been completed successfully, start IMS Connect in the normal manner. See the *IMS Connect Guide and Reference*, SC27-0946 for more information on setting up and running Local Option in IMS Connect.

If the Local Option function is activated normally, you see the following message on the system console when IMS Connect starts.

```
HWSS0790I Listening on Port=LOCAL started;M=SOC3
```

You can verify the Local Option function status by using the following IMS Connect command:

```
/Rxx, VIEWHWS
```

The following message contains the Local Option function status on your system console.

Example 7-13 An example of Local Option function status (VIEHWS)

```
R 230,VIEWHWS
IEE600I REPLY TO 230 IS;VIEWHWS
HWSC0001I HWS ID=HWS01 Racf=N
*931 HWSC0000I *IMS CONNECT READY*HWS01
HWSC0001I Maxsoc=2000 Timeout=0
HWSC0001I Datastore=IM71 Status=ACTIVE
HWSC0001I Group=IMSXCF Member=HWSMEM
HWSC0001I Target Member=IMSAV7
HWSC0001I Datastore=IM61 Status=NOT ACTIVE
HWSC0001I Group=ITSOIMS Member=HWSMEM
HWSC0001I Target Member=SCSIMS6T
HWSC0001I Port=6001 Status=ACTIVE
HWSC0001I No active Clients
HWSC0001I Port=LOCAL Status=ACTIVE
HWSC0001I No active lients
```

See Appendix E, “Command reference” on page 391 for more information about IMS Connect commands.

7.5.7 Running the Installation Verification Program (IVP)

The IVP is run using an IMS terminal that has already been predefined, as shown in Figure 7-7.

```
DFS3650I SESSION STATUS FOR IMS IM71

DATE: 11/10/02      TIME: 12:51:11
NODE NAME:          TCP61030
USER:
PRESET DESTINATION:

CURRENT SESSION STATUS:

NO OUTPUT SECURITY AVAILABLE
```

Figure 7-7 IMS terminal

Once in IMS, clear the preceding screen (Pause key) and enter the following command to run the IVP:

/FOR IVTN0

This brings up the IVP screen as shown in Figure 7-8.

```
*****
*      IMS INSTALLATION VERIFICATION PROCEDURE      *
*****

TRANSACTION TYPE : NON-CONV (OSAM DB)
DATE              : 11/10/2002

PROCESS CODE (*1) :
LAST NAME         :
FIRST NAME        :
EXTENSION NUMBER  :
INTERNAL ZIP CODE :

(*1) PROCESS CODE
ADD
DELETE
UPDATE
DISPLAY
TADD

SEGMENT# :
```

Figure 7-8 IMS Connect IVP screen

Enter display after PROCESS CODE and last1 after LAST NAME. The IVP response is the screen shown in Figure 7-9.

```
*****
*      IMS INSTALLATION VERIFICATION PROCEDURE      *
*****

                                TRANSACTION TYPE : NON-CONV (OSAM DB)
                                DATE              : 11/10/2002

PROCESS CODE (*1) :  DISPLAY

LAST NAME      :  LAST1

FIRST NAME     :  FIRST1

EXTENSION NUMBER :  8-111-1111

INTERNAL ZIP CODE :  D01/R01

ENTRY WAS DISPLAYED

                                (*1) PROCESS CODE
                                ADD
                                DELETE
                                UPDATE
                                DISPLAY
                                TADD

                                SEGMENT# :  0053
```

Figure 7-9 IMS Connect IVP screen with results

Clear the screen (Pause Key) and enter /RCL to exit the IVP.

7.5.8 Confirming IMS Connect install with the sample Java client

You can download the IMS Client for Java program from the following IBM Redbook Web site.

<ftp://www.redbooks.ibm.com/redbooks/SG246514>

The program was shipped with IMS TCP/IP OTMA Connection (ITOC) through the Web, and its purpose was installation verification by sending a transaction to the IMS via ITOC, and receiving the message that was returned from the IMS application. This program is designed to use the HWSSMPL0 user exit, and therefore, if you install HWSSMPL0 to your IMS Connect, you can also use it for IMS Connect installation verification. However, this program does not support any new functions of IMS Connect (for example, persistent socket connection and asynchronous output support). If you want to verify these functions, you must modify the sample program.

The following steps identify what you need to install, implement and use the Java sample program, and hence confirm your IMS Connect operations.

Installing the IMS Connect sample Java program

This IMS client for Java sample program can be installed on any platform on which a Sun-compatible Java Virtual Machine has been installed.

1. Move the JAVASAMP.exe file from the current directory (the directory into which it was extracted) to a directory on an OS/2 or Windows system where the sample client will be installed, or to another temporary directory.

Attention: This directory (where you move the JAVASAMP.exe file) must be located on an OS/2 or Windows drive that supports the long file names used for the Java files.

Optionally, you can enter JAVASAMP[.exe] -t at an OS/2 or Windows command prompt. This will cause an integrity check of the JAVASAMP.exe zip file to execute without actually extracting any of the files from the zip file. If the IMS client for Java sample program needs to be installed on a platform other than the one where it was expanded, copy its files to the platform where the sample program will be installed.

2. Expand the file JAVASAMP.exe. The expanded files will be placed in the current directory where JAVASAMP.exe is executed.
3. Modify the source code to match your environment. You must modify the FramelInput.java file to construct input data that matches your environment (hostname, port number, transaction, and so forth). You will need to consider changing the statements:

Port ID The default files come with port ID 9999 and 9998. You need to change all occurrences of these in the file to valid port IDs, and possibly add options of more (if you require more than 2), by adding extra entries, similar to those found in Example 7-14.

Example 7-14 Java sample - FramelInput.java changes for Port Number

```
groupPort = new CheckboxGroup();
radioButtonPort1 = new Checkbox("9999 ",groupPort,true);
radioButtonPort1.setBounds(getInsets().left +312,getInsets().top
+36,100,40);
add(radioButtonPort1);
radioButtonPort2 = new Checkbox("9998 ",groupPort,false);
radioButtonPort2.setBounds(getInsets().left +312,getInsets().top
+66,100,40);
add(radioButtonPort2);
```

RACFUser The default user IDs come as USER01-4, and the select(0) statement, indicating the first one is brought up by default. You

will need to modify this to reflect valid user IDs for use. Refer to Example 7-15.

Example 7-15 Java sample FrameInput.java changes for RACF USERID

```
choiceSAFID =new Choice();
choiceSAFID.addItem("SAFUsrID ");
choiceSAFID.addItem("USER01 ");
choiceSAFID.addItem("USER02 ");
choiceSAFID.addItem("USER03 ");
choiceSAFID.addItem("USER04 ");
```

Similar changes to those identified for the RACFUser, are also needed for: Tran, DS (datastore-id), GRP (RACF user ID group), Client name (client ID), HostName (IP address).

The HWSSMPL0 program does not impose any limitations on the number of input and output message segments. The Java client uses multi-segment input and output text areas. IMS Connect requires that all active clients have a unique client ID that, for the IMS Client for Java, is taken from the client ID field defined in FrameInput.java. Therefore, if you intend to allow multiple IMS Client for Java to run simultaneously, which is usually the case, you must either modify the FrameInput.java file so that the client ID will be unique for each active client at any given time, or ensure in some other way that the client ID for each active client is unique. For test purposes, just be sure that you use a unique client ID for each Java client when you click **Submit**.

4. Go to the command prompt, and the directory containing all these expanded Java client source files for the sample program. Enter the command:

```
javac *.java.
```

This creates the IMS client for Java class files in the same directory.

5. Install the HWSSMPL0 user exit to your IMS Connect resource library. Compile and link-edit HWSSMPL0, then modify the IMS Connect configuration file to include the HWSSMPL0 user exit in the TCPIP statement as follows:

```
TCPIP=(...,EXIT=(HWSSMPL0,...),...)
```

If you use the HWSSMPL1 user exit, modify the following source code and also install the HWSSMPL1 user exit:

- IRM_ID (in SampleTran class). The default IRM_ID comes as *SAMPLE*. You need to modify this to reflect *SAMPL1* for HWSSMPL1.

```
final static String identifier ="*SAMPL1*";
```

- Add “New message header” (to SampleTran class, STReceive() method). Add the 4-byte I/O area to receive the total message length header. Refer to Example 7-16.

Example 7-16 Java sample SampleTran.java changes for total message header

```
public void STReceive(){
    short recLength;
    byte b [] =new byte [8 ];
    byte data [] =new byte [1024 ];
    short sh;
    int totLength;
    String tempID,tempSeg;
    int i =0;
    try {
        DataInputStream in =new DataInputStream(s.getInputStream());
        totLength =in.readInt();//read new LLLL header
        recLength =in.readShort();//read total length
        totalLength =(int)recLength;//convert short to int
        sh =in.readShort();//read ZZ
    }
```

Install the HWSSMPL1 user exit to your IMS Connect resource library. Compile and link-edit HWSSMPL1, then modify the IMS Connect configuration file to include the HWSSMPL1 user exit in the TCPIP statement as follows:

```
TCPIP=(...,EXIT=(HWSSMPL1,..),..)
```

Refer to *IMS Connect Guide and Reference*, SC27-0946-01, for more information about HWSSMPL1.

Using the sample Java program

After the IMS Client for Java sample program source code has been compiled, you can run the IMS Client for Java program as a standalone Java application by entering the following command:

```
java ClientLauncher
```

You can also run the IMS Client for Java sample program as an applet, either locally or remotely. If you want to run the applet as a local applet, then you just have to open ClientLauncher.html as a local file from a Java-enabled Web browser that allows an applet to execute socket calls.

If you want to run the applet as a remote applet, then you have to copy all .class, .gif, and .html files to your Web server's HTML directory and bring up the ClientLauncher (ClientLauncher.html) in a Java-enabled Web browser.

ClientLauncher starts a new Transaction Input client frame (Figure 7-10) every time you click the Start A Client button. If you start more than one client window, the newer windows hide the existing client window. All of these client frame windows run independently, so you must close each window individually if you are running the IMS Client for Java as an applet. If you are running the IMS Client for Java as a Java application, you can either close the client frame windows individually (as you must do if IMS Client for Java is running as an applet) or all at the same time by closing the original window (the ClientLauncher running as an application). When you close the ClientLauncher application, all client frame windows that were generated by that application close automatically.

The screenshot shows a window titled "Transaction Data" with a standard Windows-style title bar. The window is divided into two main sections: "INPUT DATA" on the left and "OUTPUT DATA" on the right. Below these sections are eight pushbuttons: "Send", "Receive", "Ack", "Nack", "Resume Tpipe", "Send Only", "Disconnect", and "Close".

INPUT DATA Section:

- HostName: 9.12.6.117 (dropdown)
- Transaction: IVTNO (dropdown)
- Client ID: CLIENT01 (dropdown)
- SAFID: USER01 (dropdown)
- Sync Level: NONE (dropdown)
- Password: (text field)
- Port: 6001 (radio button selected), 6002 (radio button)
- DataStore ID: IMSB (dropdown)
- GROUP: SYS1 (dropdown)
- Commit Mode: 1 - SEND THEN COMMIT (dropdown)
- Input Text: A text area containing "IVTNO DISPLAY LAST1"

OUTPUT DATA Section:

- TRAN: (text field)
- RC = (text field)
- MOD: (text field)
- RS = (text field)
- OUTPUT TEXT: A large text area for displaying transaction output.

Figure 7-10 Java Transaction input client frame

The IMS Client for Java program provides pulldown lists and a text area for supplying the information for a transaction request message that will be used to contract a message to be sent to IMS Connect. The values associated with the list items can be changed as described above under Modifying the IMS Client for Java program. The functions of the eight pushbuttons at the bottom of the IMS Client for Java window are:

- Send** Connect to IMS Connect if a TCP/IP socket connection does not exist and send a transaction request message.
- Receive** Request output from the current transaction request.

Ack	Send a message acknowledging (accepting) the transaction output just received.
Nack	Send a message rejecting the transaction output just received and then automatically receive and display an abnormal termination message from IMS Connect.
Resume Tpipe	Reconnect to IMS Connect to receive asynchronous output only.
Send Only	Connect to IMS Connect if a TCP/IP socket connection does not exist and send a message containing a non-response transaction request to IMS Connect.
Disconnect	Send a request to deallocate the TCP/IP socket connection to IMS Connect and then automatically receive and display an abnormal termination message from IMS Connect.
Close	Close the IMS Client for Java program window.

The IMS Client for Java program is intended to allow users to manually execute simple transactions such as those that are part of the IMS INSTALL/IVP sample application. The program is designed to allow you to manually start multiple clients if you choose to do so. However, when using multiple clients simultaneously, make sure that each client uses a unique client ID. Otherwise, if you try to send messages to IMS from two clients both of which use the same client ID, you will get a duplicate client error for the second client assuming the first client is still active when the second client attempts to connect (i.e., send a message) to IMS Connect.

To run the IVTNO transaction from the form shown, simply type the transaction code followed by the function code and last name, for example:

```
IVTNO DISPLAY LAST1
```

Restriction: The IMS Client for Java program is sample code and is not intended to represent a robust program suitable for production environments.

As a result, it is relatively easy to put the IMS Client for Java program into states where, for instance, it cannot continue a conversation.

Another specific example is: A conversation using one of the IMS INSTALL/IVP transactions has been ended and the sample program does not have the ability to recognize that the conversation has ended. As a result, it attempts to process the send event as the "next" iteration of the already ended conversation. Since

the conversation has ended, a Java exception is thrown. In this case and in other such cases, it is usually sufficient to press the Disconnect button and start over. In other cases, it may be necessary to close the client that is not responding correctly and start a new client in its place.

7.5.9 Starting an IMS Connect trace

IMS Connect provides a line trace function to capture data that is received from and sent to a client. The line trace contains a copy of the first 670 bytes of the data as it is passed to the user message exit and upon return from the user message exit. Line traces are intended for use in problem resolution.

Allocate the trace data set from TSO using settings similar to:

```
Data Set Information
Command ==>
Data Set Name ....: yourname
General Data Current Allocation
Volume serial....: USER0 Allocated cylinders:
Device type ....: 3390 Allocated extents. :
Organization ....:PS
Record format....:FB
Record length....: 1440
Block size .....: 14400 Current Utilization
Extent cylinders: Used cylinders ...:
Secondary cylinders : 5 Used extents ...:
Creation date . . .: 2001/11/03
Expiration date . .: ***None***
```

We added the following statements to our IMS Connect startup procedure:

```
//*****
//*  TRACE DS MUST BE FB, LRECL=1440, BLKSIZE=14400
//*****
//HWSRCRD DD DSN=IMS71.TRACE,DISP=SHR
```

Use the RECORDER command:

```
/R nnn,RECORDER ON
```

to activate and terminate the line trace function. Example 7-17 illustrates how to print the line trace data set:

Example 7-17 Sample JCL for line trace function

```
//IDCAMS JOB JOB 1, IDCAMS,MSGLEVEL=1,CLASS=K,TIME=1440
//SELECT EXEC PGM=IDCAMS
//DD1 DD DSN=IMS71.TRACE,DISP=SHR
//SYSPRINT DD SYSOUT=*
```

```
//SYSIN DD *  
PRINT INFILE(DD1)
```

7.6 IMS Connector for Java based on IMS Connect

The IMS Connector for Java provides a way to create Java applications that access IMS transactions using IMS Connect. In conjunction with the VisualAge for Java development environment, IMS Connector for Java enables you to rapidly develop Java applications that run your IMS transactions. With additional support from IBM WebSphere Application Server, you can build and run Java servlets that access your IMS transactions over the Internet.

IMS Connector for Java uses IMS Connect to access IMS. IMS Connect is a component, separately provided and installed, that runs on the host IMS machine and supports TCP/IP communication to IMS. A Java application accesses IMS Open Transaction Manager Access (OTMA) through IMS Connect.

The Java application acts as a TCP/IP client to IMS Connect. IMS Connect accepts messages from its TCP/IP clients and routes them to the IMS OTMA using the Extended Communication Facility (XCF).

IMS Connector for Java is comprised of two components:

- ▶ The development component of IMS Connector for Java is provided with VisualAge for Java. The VisualAge for Java Enterprise Access Builder (EAB) environment enables you to develop Enterprise Access Builder (EAB) commands, Enterprise Java Beans (EJBs), servlets, and Java applications that use IMS Connector for Java to access IMS transactions. VisualAge for Java also provides a WebSphere test environment that you can use to test the programs you develop. The WebSphere Studio environment enables you to develop Java servlets from the EAB commands that you develop in VisualAge for Java.
- ▶ The runtime component of IMS Connector for Java is provided as a component of IMS Connect Version 1 Release 2 (Program Number 5655-E51). The Java2 Platform (J2EE) Connector Architecture (JCA) implementation of this runtime component is also referred to as the IBM WebSphere Adapter for IMS. It is packaged as a RAR file, `imsico.rar`, for deployment into the JCA-compliant WebSphere Application Server. The RAR file is installed to a target directory from the IBM IMS Connect, Version 1 Release 2. WebSphere Application Server provides an environment in which you can run the servlets and EJBs that you developed (in VisualAge for Java, WebSphere Studio, or with other development tools).

TCP/IP application programmers can choose to write programs using the direct sockets interface or they can take advantage of code-generating packages such as the IMS Connector for Java, which ships with the VisualAge for Java Enterprise Edition toolkit. These programs, whether they are user-written or generated, can communicate with IMS Connect in a variety of ways.

The IMS Connector for Java converts the application input data into an OTMA message format before shipping the data stream to IMS Connect and, on reply, takes the OTMA message format and converts it to the output data stream that can be understood by the application. User-written programs can also do this but as an alternative, especially for those with no knowledge of OTMA, they can take advantage of a defined IMS Connect application protocol (set of architected headers) that relies on IMS Connect to deal with OTMA message format on behalf of the TCP/IP program.

Another option to connect between IMS Connector for Java and IMS Connect is the Local Option support. Local Option provides non-socket access to IMS transactions and data from z/OS or OS/390 WebSphere Application Server when the WebSphere Application Server, IMS Connector for Java, and IMS Connect are all running in the same image.

For more details about using the connector, see *e-business Cookbook, Volume III: Application Development*, SG24-5980.

The following sections are solely based on two books:

- ▶ *IMS Connector for Java User's Guide and Reference*, SC27-1559
- ▶ *IMS e-business Connectors: A Guide to IMS Connectivity*, SG24-6514

7.6.1 J2EE Connector support in IMS Connector for Java

IMS Connector for Java includes both a Common Connector Framework (CCF) implementation and a Java 2 Platform (J2EE) Connector Architecture (JCA, sometimes also used as J2C) implementation.

Release 1.2.2 of IMS Connector for Java introduces JCA support for the z/OS and OS/390 platform. See “Prerequisites for using IMS Connector for Java” in *IMS Connector for Java User's Guide and Reference*, SC27-1559 for more information on getting this support.

The J2EE Connector Architecture defines a standard architecture for connecting the J2EE platform to heterogeneous Enterprise Information Systems (EISs) such as IMS. The JCA architecture uses a J2EE connector (also known as a resource adapter), such as IMS Connector for Java. By conforming to the J2EE Connector architecture, which is defined and standardized as part of the Java 2 Platform, Enterprise Edition (J2EE), the resource adapter no longer needs to be customized for each J2EE application server to connect to the EIS.

In addition, the J2EE Connector Architecture defines a set of standard programming interfaces that determine how a J2EE application interacts with a J2EE Resource Adapter to connect to the EIS. As a result, by using the J2EE Connector Architecture, the J2EE applications do not require the addition of custom code to connect to different EISs.

For more information on the J2EE architecture and its concepts, see the J2EE Connector Architecture Specification at :

<http://java.sun.com/j2ee/download.html>

7.6.2 Elements of the IMS Connector for Java

All classes mentioned below are in the IMS Connector for Java package `com.ibm.connector2.ims.ico`. Interfaces beginning with `javax.resource` are part of the JCA architecture.

To view the Javadoc for these classes, go to VisualAge for Java and click **Help > API Reference > Reference > IBM APIs > Connectors > IMS Connector**.

IMS Connector for Java's implementation of the JCA architecture is provided in phases. Classes currently provided are summarized below. IMS Connector for Java uses these classes in collaboration with WebSphere Application Server to provide connection pooling in the JCA environment. Connection pooling is transparent to the servlet or EJB.

IMS Connector for Java implements the JCA System Contracts by providing the following classes (JCA interfaces are noted in parentheses):

- ▶ `com.ibm.connector2.ims.ico.IMSManagedConnectionFactory`
(`javax.resource.spi.ManagedConnectionFactory`)

An `IMSManagedConnectionFactory` instance is a factory of both `IMSManagedConnection` instances and `IMSConnectionFactory` instances.

- ▶ `com.ibm.connector2.ims.ico.IMSManagedConnection`
(`javax.resource.spi.ManagedConnection`)

An abstract class that represents the physical connection to IMS Connect. The `IMSManagedConnection` has two subclasses:

- `com.ibm.connector2.ims.ico.IMSTCPIPManagedConnection`
- `com.ibm.connector2.ims.ico.IMSLocalOptionManagedConnection`

An `IMSTCPIPManagedConnection` instance represents a TCP/IP connection to IMS Connect. An `IMSLocalOptionManagedConnection` instance represents a connection using Local Option to communicate with IMS Connect.

- `com.ibm.connector2.ims.ico.IMSManagedConnectionMetaData`
(`javax.resource.spi.ManagedConnectionMetaData`)

Provides information about an `IMSManagedConnection`.

IMS Connector for Java implements the JCA Common Client Interface (CCI) by providing the following classes (JCA interfaces are noted in parentheses):

- Classes that represent a connection factory and an application level connection:

- `com.ibm.connector2.ims.ico.IMSConnectionFactory`
(`javax.resource.cci.ConnectionFactory`)

An `IMSConnectionFactory` instance is a factory for `IMSConnection` instances.

- `com.ibm.connector2.ims.ico.IMSConnection`
(`javax.resource.cci.Connection`)

An `IMSConnection` instance is an application level handle that is used by a component to access IMS using IMS Connect.

- `com.ibm.connector2.ims.ico.IMSConnectionSpec`
(`javax.resource.cci.ConnectionSpec`)

An `IMSConnectionSpec` instance is used by components, when obtaining a connection, to provide specific values for `UserName`, `Password`, and `GroupName`. These values are used instead of the default values provided by the `IMSConnectionFactory` instance.

- Classes that enable a component to drive an interaction (specified through an `InteractionSpec`) with IMS using IMS Connect:

- `com.ibm.connector2.ims.ico.IMSInteraction`
(`javax.resource.cci.Interaction`)

An `IMSInteraction` instance enables a component to interact (run a transaction) with IMS using IMS Connect.

- `com.ibm.connector2.ims.ico.IMSInteractionSpec`

(javax.resource.cci.InteractionSpec)

An IMSInteractionSpec instance provides properties for driving the interaction with IMS using IMS Connect.

- ▶ Classes that provide basic meta information about the IMS Connector for Java implementation and a connection to IMS using IMS Connect:
 - com.ibm.connector2.ims.ico.IMSConnectionMetaData
(javax.resource.cci.ConnectionMetaData)
Provides information about an IMSConnection.
 - com.ibm.connector2.ims.ico.IMSResourceAdapterMetaData
(javax.resource.cci.ResourceAdapterMetaData)
Provides information about IMS Connector for Java.
- ▶ Additional classes and interfaces provided by IMS Connector for Java are:
 - IMSInteractionSpecProperties
IMSInteractionSpecProperties is an interface that defines named constants that are used to configure an IMSInteractionSpec instance.
 - IMSTraceLevelProperties
IMSTraceLevelProperties is an interface that defines named constants that are used for controlling the level of tracing and logging that is performed by IMS Connector for Java.
 - IMSConnectorMigrator
IMSConnectorMigrator is a class that is used by VisualAge for Java to migrate a CCF EAB command to a J2EE EAB command.
 - DFSMsg
A DFSMsg bean is used to capture "DFS" messages returned by IMS.

7.6.3 Prerequisites for using IMS Connector for Java

The following is a list of minimum software requirements for IMS Connector for Java:

- ▶ VisualAge for Java Enterprise Edition Version 4.0
The installation of VisualAge for Java 4.0 provides you with the Common Connector Framework (CCF) implementation of IMS Connector for Java. To use the J2EE Connector Architecture (JCA) implementation of IMS Connector for Java, you need to add the following updates to your VisualAge for Java 4.0 environment:

- VisualAge for Java Beta J2EE Connector support is required to use the JCA support. Follow the instructions in “IBM Enterprise Access Builder BETA NOTES 4.0” on the VisualAge for Java 4.0 Additional Features installation CD. This information is in the readme.eab file in the \extras\BetaJ2EEConnectors folder on the Additional Features installation CD.

- The JCA implementation of IMS Connector for Java is provided as an update to IBM VisualAge for Java Enterprise Edition Version 4.0. This update is available on the VisualAge Developer Domain at:

<http://www7.software.ibm.com/vad.nsf>.

- ▶ WebSphere Application Server V4.0.1 for z/OS and OS/390 with APAR PQ55873
- ▶ IMS Connect Version 1 Release 2

The runtime component of IMS Connector for Java is provided as an update to the IMS Connector for Java component of the IMS Connect Version 1 Release 2 product. If you plan to use the JCA implementation of IMS Connector for Java running WebSphere Application Server on the z/OS and OS/390 platform, the following APARs for IBM IMS Connect, Version 1 Release 2 (Program Number 5655-E51) are required:

- APAR PQ57192

This APAR updates the IMS Connector for Java component with JCA and 2-phase commit support for WebSphere Application Server for z/OS and OS/390. It installs the IMS Connector for Java runtime component (the RAR file for WebSphere Application Server for z/OS and OS/390) to a target directory on your system.

- APAR PQ57191

This APAR updates the IMS Connect component with 2-phase commit support to be used by IMS Connector for Java.

- ▶ IMS Version 7 Release 1

- APAR PQ57868

This APAR must be applied to your IMS V7.1 system if you plan to use JCA and global transaction (2-phase commit) support on WebSphere Application Server on z/OS and OS/390.

To run a Java application that uses the JCA implementation of IMS Connector for Java, IMS and IMS Connect must be running on your host machine.

- ▶ IMS Connect Version 1 Release 2

Ensure that the outstanding IMS Connect reply appears on the target machine's system console. For example:

```
*09.34.41 STC00156 *16 HWSC0000I *IMS CONNECT READY*ims_connect_name
```

Note: IMS Connect Name is referred to in IMS Connect as “HWS ID.” Verify that the PORT and DATASTORE are ACTIVE by entering the IMS Connect command VIEWHWS at the IMS Connect outstanding reply. Verify that the eyecatcher in module HWSHWS00 in the HWS RESLIB is at least “120.” For example:

```
...HWSHWS00+0120+...
```

► **IMS Version 7 Release 1**

Ensure that the IMS outstanding reply appears on the target machine’s system console. For example:

```
14.26.55 JOB00175 *15 DFS996I *IMS READY*
```

Verify that the XCF status of both the IMS and IMS Connect members is ACTIVE by entering the IMS command /DISPLAY OTMA at the outstanding IMS reply:

```
DFS000I GROUP/MEMBER XCF-STATUS USER-STATUS SECURITY SYS3
DFS000I GROUPNM SYS3
DFS000I -IMSMEM ACTIVE SERVER NONE SYS3
DFS000I -ICONNMEM ACTIVE ACCEPT TRAFFIC SYS3
DFS000I *01218/111559*SYS3
```

When using an IVP, you can use either the TCP/IP or Local Option communication protocols. If you are using TCP/IP, ensure that you can successfully “ping” the target host machine from your VisualAge for Java workstation. If you are using Local Option, ensure that IMS Connect and the IVP are running on the same MVS image.

7.6.4 Preparing to use IMS Connector for Java

This section describes how to create an environment in which you can develop and run Java programs that use the J2EE Connector Architecture (JCA) implementation of IMS Connector for Java.

IMS Connector for Java provides two types of Installation Verification Programs (IVPs) that you can run from the following environments:

- The IVP GUI, which is run in VisualAge for Java for Windows, verifies that VisualAge for Java is correctly configured to access your IMS and IMS Connect system.
- The IVP Servlet, which is run in WebSphere Application Server for z/OS and OS/390, verifies that WebSphere Application Server is correctly configured to access your IMS and IMS Connect system.

Each of these IVPs invokes IMS commands without requiring a host IMS application.

Preparing your VisualAge for Java environment

IMS Connector for Java is included with VisualAge for Java Enterprise Edition; it does not usually require a separate installation process. Because IMS Connector for Java is part of VisualAge for Java, its software prerequisites are the same as those for VisualAge for Java.

IMS Connector for Java features both a J2EE Connector Architecture (JCA) implementation and a Common Connector Framework (CCF) implementation. To use IMS Connector for Java, perform the following steps in the order indicated:

1. Install VisualAge for Java Enterprise Edition 4.0.

Important: If you do not select the “complete installation” option of VisualAge for Java, you must include the Transaction Access Builder feature in your installation. If you have already installed VisualAge for Java 4.0 and an earlier version of IMS Connector for Java, you can skip this step.

2. To use the JCA implementation of IMS Connector for Java, ensure that the components related to the J2EE Connector Architecture are installed in VisualAge for Java. Instructions for installing these components can be found on VisualAge for Java’s installation media in the file `extras/BetaJ2EEConnectors/readme.eab`.
3. Download and install the IMS Connector for the Java 1.2.2 update from the VisualAge Developer Domain at:
<http://www7.software.ibm.com/vad.nsf>
4. Add to your VisualAge for Java workspace the VisualAge for Java feature IMS Connector 1.2.2 and, optionally, IMS Connector Samples 1.2.2. The IMS Connector Samples 1.2.2 project will replace any previous versions of the project in your VisualAge for Java workbench.
 - a. Start the Workbench in the VisualAge for Java IDE. Click **Start > Programs > IBM VisualAge for Java for platform > IBM VisualAge for Java**, where *platform* is the platform on which you have VisualAge for Java. If the VisualAge Welcome to VisualAge dialog box appears, click **Go to the Workbench** and then OK. The IDE workbench appears.
 - b. From the File menu, click **Quick Start**.
 - c. In the Quick Start window, click **Features** on the left pane. Then, click **Add Feature** on the right pane and click **OK**.

- d. Select the feature **IBM EJB Development Environment 3.5.3** and click **OK** to add it to the VisualAge for Java workspace. This step is required.
- e. Repeat the step above to add the features IMS Connector 1.2.2 and IMS Connector Samples 1.2.2. The feature IMS Connector will be added to your workbench as project Connector IMS 1.2.2 and the feature Connector IMS Samples 1.2.2 will be added as project Connector IMS Samples.

Check the **All Problems** tab of your workbench when adding a feature to see if additional projects (such as the IBM Common Connector Framework, IBM Common Connector Framework2, and J2EE Connector Architecture) need to be added in order to resolve any problems.

After you add the IMS Connector 1.2.2 features to your VisualAge for Java workspace, you have access to the following:

- ▶ Connector IMS, a VisualAge for Java project that consists of four packages:
 - com.ibm.connector2.ims.ico
This includes the classes of IMS Connector for Java's JCA implementation.
 - com.ibm.ims.ico
This includes the classes that implement the internal logic of IMS Connector for Java's JCA implementation.
 - com.ibm.connector.imstoc
This includes the classes of IMS Connector for Java's CCF implementation. Many of the classes in this package are "support" classes, and are not used by VisualAge for Java application developers.
 - com.ibm.imstoc
This includes the classes that implement the internal logic of IMS Connector for Java's CCF implementation. All of the classes in this package are "support" classes, and are not used by VisualAge for Java application developers.

The classes and methods that are used in the VisualAge for Java application development process are documented in the Reference section of the VisualAge for Java Web help. To view this documentation, click **Help > API Reference > IBM APIs > Connectors > IMS Connector**.
- ▶ IMS Connector for Java Online Help in HTML format
The PDF format of the *IMS Connector for Java User's Guide and Reference* is available on the IMS Connector for Java Web page in the IMS Web site at:

www.ibm.com/ims.
- ▶ An IMS Connector for Java IVPs

You can use this to verify that a Java application using the JCA implementation of IMS Connector for Java can access IMS from within the VisualAge for Java environment. This program is in the `IMSConnIVPMainDialog` class in the Connector IMS Samples project in the package `com.ibm.connector2.ims.ico.sample.ivp`.

► An IMS Connector for Java (IVP) servlets

You can use this to verify that a servlet using the JCA implementation of IMS Connector for Java can access IMS from within a WebSphere Application Server environment. The source for this program is in the Connector IMS Samples project in the package `com.ibm.connector2.ims.ico.sample.ivp.servlet`.

The source for the JCA IMS Connector for Java IVP servlet is provided with VisualAge for Java for illustration purposes only. The runtime of this servlet is included with the IMS Connector for Java runtime component packaged with the IMS Connect product. Instructions on how to use this servlet to verify your WebSphere Application Server environment are included with the installation instructions for the IMS Connector for Java runtime component and in 7.6.5, “Preparing your WebSphere Application Server environment” on page 269.

Sample code is available to aid in the development of your Java applications using the JCA implementations of IMS Connector for Java. This code is available in the Connector IMS Samples project.

Note: The IVP is a VisualAge for Java for Windows program and therefore does not use the Local Option connections. Instead, use TCP/IP for communication and to run the IVP.

Running the VisualAge for Java IVP

Update VisualAge for Java’s class path to point to the locations of the required class libraries. Do the following:

1. Right-click the **IMSICOIVPMainDialog** class in package `com.ibm.connector2.ims.ico.sample.ivp`, then select **Properties**.
2. Ensure that the Class path tab is selected on the Properties window and click **Compute now** to compute the class path for the IVP. Click **OK**.

To run the IVP, select the **IMSICOIVPMainDialog** class in package `com.ibm.connector2.ims.ico.sample.ivp`. From the Selected menu, click **Run > Run main...** The IMS Connector for Java IVP window appears (Figure 7-11 on page 268).

Because execution of the IVP causes the IMS /STA OTMA command to be sent to the target IMS machine, you must provide values for the fields described below before clicking **SUBMIT**.

Figure 7-11 The VisualAge for Java IVP GUI

- Host name** The TCP/IP host name of the machine running IMS Connect.
- Port** The TCP/IP port assigned to IMS Connect.
- Datastore** The Datastore ID you specified when you installed IMS Connect.
- User ID** The SAF user ID to be passed to OTMA. For example, the RACF user ID. User ID may be optional depending on how you specify IMS OTMA and IMS Connect security. However, you can provide it, along with a password, if you want to verify your installation's security configuration.
- Password** Password can be optional (see User ID).
- Group name** Group name can be optional (see User ID).

The objective of the IVP is to receive a message from IMS in reply to the /STA OTMA command. One of two possible messages is returned, depending on the security configuration of your host environment and whether the given user ID is permitted to issue the /STA command. Either message indicates successful execution of the IVP:

```
DFS1292E SECURITY VIOLATION
DFS058I 17:33:32 START COMMAND COMPLETED
```

This test does not verify that a particular IMS transaction can run, only that the path to IMS is available.

If an exception is returned in response to the command, check the console window. The exception and stack trace provide diagnostic information. The following are common causes for IVP failure:

- ▶ Host name is misspelled or is not sufficiently qualified.
- ▶ IMS Connect is not active.
- ▶ An incorrect port.
- ▶ A port is not active.
- ▶ A TCP/IP failure. Always ensure a successful “ping” prior to running the IVP.
- ▶ The datastore name is incorrect or misspelled. The datastore name must be in uppercase characters.
- ▶ Datastore is not active.
- ▶ Wrong level of IMS Connect running on the host. See “Prerequisites for using IMS Connector for Java” for the required level of IMS Connect.

Important: If you do not compute the class path for the class `IMSConnIVPMainDialog` as described above, the IMS Connector for Java IVP throws an exception, indicating that one of the classes cannot be found. In some cases, the IVP might not respond.

7.6.5 Preparing your WebSphere Application Server environment

To prepare to run Java applications (for example, servlets or EJBs) that use the JCA implementation of IMS Connector for Java in WebSphere Application Server for z/OS and OS/390, complete the following steps:

1. Install the runtime component IMS Connector for Java with JCA support on the target z/OS and OS/390 system.
2. Configure IMS Connector for Java with JCA support on the target z/OS and OS/390 system.
3. Prepare to run the Installation Verification Program (IVP).
4. Deploy and configure IMS Connector for Java as a J2EE server resource in WebSphere Application Server for z/OS and OS/390.
5. Deploy the IMS Connector for Java IVP.
6. Start the server to run the IVP.
7. Run the IVP to access the target IMS.

Each of these steps is described here in detail.

Related Reading: More information about Resource Recover Services (RRS) transaction support is available in the WebSphere Application Server for OS/390 and z/OS documentation. The documentation for APAR PQ55873 for WebSphere Application Server V4.0.1 for z/OS and OS/390 has been integrated into the formal publications for WebSphere for z/OS.

To access the latest publications, go to the product library page at:

<http://www.ibm.com/software/webservers/appserv/>

Before you can deploy and run the IVP servlet, the following components and configurations are required:

- ▶ WebSphere Application Server V4.0.1 or higher for z/OS and OS/390 that has been tested by the supplied J2EE Web Application IVPs

The IMS Connector for Java sample J2EE applications require a definition of a J2EE application server and the initial modifications to the Web container configuration files of this server. They also require additional modifications to the environment variables of the HTTP server. For details, see *WebSphere Application Server V4.0.1 for z/OS and OS/390 Installation and Customization*, GA22-7834.

This manual provides an example of defining a J2EE application server (BBOASR2) and creating an instance of this server (BBOASR2A) for running the WebSphere Application Server IVPs. The initial modifications to the Web container configuration files and HTTP server environment variables can also be found within the steps for setting up the server's J2EE Web application verification programs.

Additionally, be sure that you have met the prerequisites for WebSphere Application Server described in 7.6.3, "Prerequisites for using IMS Connector for Java" on page 262.

- ▶ WebSphere for z/OS and OS/390 Administration application

The Administration application is packaged in a self-extracting executable file supplied by the WebSphere Application Server for z/OS and OS/390. This executable file also provides the Operations application. Together these programs are often called the "system management enhanced user interface" (SMEUI). For details on installing the Administration and Operations applications, see *WebSphere Application Server V4.0.1 for z/OS and OS/390 Installation and Customization*, GA22-7834.

The Administration application manages administration tasks for the WebSphere Application Server from the terminal side and defines an IMS Connection Factory, creates an instance of this resource, and installs J2EE applications bundled within enterprise archive (EAR) files.

Step 1: Install the runtime component on the target system

Using SMP, install the required APARs (APAR PQ57191 and corequisite APAR PQ57192) for the IMS Connect 1.2 product on the z/OS or OS/390 system image where you will start the WebSphere Application Server for z/OS and OS/390 server for running EJBs and servlets that use the IMS Connector for Java.

Successful completion of the installation places the following files in directory `imsico` of a target directory (`<target_dir>`) of your choice or the default target directory. The default target installation directory is `/usr/lpp` on the z/OS and OS/390 system.

`<target_dir>/imsico/README`

Information about the files installed for IMS Connector for Java

`<target_dir>/imsico/J2C/README`

Information about the files in the J2C directory

`<target_dir>/imsico/J2C/imsico.rar`

Resource Adapter Archive file for IMS Connector for Java (J2C)

`<target_dir>/imsico/J2C/HowTo.html`

Information about IMS Connector for Java

`<target_dir>/imsico/J2C/IMSICOIVPzOS.html`

Information about the IVP

`<target_dir>/imsico/J2C/IMSICOIVPzOS.ear`

The Enterprise Application Archive file for IMS Connector for Java's IVP

`<target_dir>/imsico/CCF/README`

Information about the files in the CCF directory

`<target_dir>/imsico/CCF/imsiconn.jar`

IMS Connector for Java class libraries for CCF

`<target_dir>/imsico/CCF/libimsconn.so`

IMS Connector for Java Local Option native library for CCF

Step 2: Configure IMS Connector for Java on the target system

1. Create an HFS work directory that allows read, write, and execute authority. Use a meaningful name for the directory (for example: `/usr/lpp/connectors`).

2. In the install directory for IMS Connector for Java, look for the file `imsico.rar` and copy it into the work directory you created in the previous step.

Tip: If your installation used the default directory for installing IMS Connector for Java, you will find the `imsico.rar` in the directory `/usr/lpp/imsico/J2C`.

3. Under the work directory, expand the `imsico.rar` file by entering the following command:

```
jar -xvf imsico.rar
```

Result: The expansion extracts the following files into the directory:

`imsico.jar`

`libimsico.so`

Note: Additional files are also extracted, but they are not used for the deployment of the connector.

4. Use the following command to give execute permission to the `imsico.jar` and `libimsico.so` files under the work directory:

```
chmod ugo+x imsico.jar  
chmod ugo+x libimsico.so
```

Step 3: Prepare to run the IVP

See 7.6.3, “Prerequisites for using IMS Connector for Java” on page 262 for information on IVP prerequisites.

Step 4: Deploy and configure a J2EE server resource

To run the IVP it is necessary to create and configure a Connection Factory for IMS Connector for Java as a J2EE server resource to your target WebSphere Application Server for z/OS and OS/390. The Connection Factory will be used by the IVP to create a connection between the IVP, your IMS Connect, and IMS.

1. Verify that the bootstrap daemon is started.
 - a. On the host machine, display the active jobs by entering the command:

```
/D A,L
```
 - b. If `BBODMN` is not an active job displayed in the list, start the bootstrap server by issuing the command:

```
/S BBODMN.DAEMON01
```
 - c. Verify that these messages are displayed:

```
BBOU0016I  INITIALIZATION COMPLETE FOR DAEMON DAEMON01.  
BBOU0020I  INITIALIZATION COMPLETE FOR CB SERIES CONTROL REGION SYSMGT01.  
BBOU0020I  INITIALIZATION COMPLETE FOR CB SERIES CONTROL REGION NAMING01.  
BBOU0020I  INITIALIZATION COMPLETE FOR CB SERIES CONTROL REGION INTFRP01.
```

2. Start the Operations application and connect to the servers.
 - a. Open the Operations application. For example: click **Start > Programs > IBM WebSphere for z/OS > Operations** and wait for the Login window to appear.
 - b. Fill in the dialog with the Bootstrap server IP name (for example, the DAEMON_IP environment variable or the name specified for the bootstrap server's IP address in the host file on the workstation), Port (default 900), Userid (default CBADMIN), and Password (default CBADMIN).
 - c. Click **OK** to log in and establish connections to the servers. Wait for the "WebSphere for Application Server for z/OS and OS/390 Operations" window and the icons for the servers available on the host machine to appear.
3. Ensure that the Web server is not active:
 - a. On the host machine, display all active jobs by issuing the following command:

```
/D A,L
```
 - b. If the Web server (for example, IMWEBSRV) is an active job, issue the following command to stop it:

```
/STOP IMWEBSRV
```
4. Ensure that the J2EE server is not active:
 - a. In the WebSphere Application Server for z/OS and OS/390 window, highlight the instance of the J2EE server (for example, BBOASR2A).
 - b. Click **Selected > Stop**.

Note: You can also stop the server manually by issuing the following command:

```
STOP BBOASR2.BBOASR2A
```

5. Start the Administration application and connect to the servers:
 - a. Open the Administration application. For example: click **Start > Programs > IBM WebSphere for z/OS > Administration** and wait for the Login window to appear.
 - b. Fill in the dialog with the Bootstrap server IP name (for example, the DAEMON_IP environment variable or the name specified for the bootstrap

- server's IP address in the host file on the workstation), Port (default 900), Userid (default CBADMIN), and Password (default CBADMIN).
- c. Click **OK** to login and establish connections to the servers. Wait for the "WebSphere for Application Server for z/OS and OS/390 Administration" window to appear.
 6. Add a new conversation:
 - a. Select the **Conversations** folder.
 - b. Under the Selected drop-down menu, choose **Add** and wait for the conversation name and description fields to appear.
 - c. In the Conversation Name text box, enter the name for the new conversation (for example, IMSICO).
 - d. Click **Selected > Save**.
 - e. Wait for the completion message (for example, BBON0515I Conversation IMSICO was added.) to appear in the status text box above the status bar in the Administration window.
 7. Ensure that connection management is configured into the sysplex by completing the following steps:
 - a. Expand the Conversations folder (by clicking the plus sign next to it).
 - b. Expand the new conversation (for example, IMSICO).
 - c. Expand the Sysplexes folder and highlight the sysplex name (for example, PLEX1).
 - d. Click **Selected > Modify** to update the sysplex definition.
 - e. Under the "Configuration Extensions" section of the sysplex definition, check the box labeled "Connection Management."
 - f. Click **Selected > Save** to save your changes.
 8. Set up the IMS Connector for Java classes to be used by WebSphere Application Server by setting up the following CLASSPATH and LIBPATH environment variable values. Do the following:
 - a. Expand the sysplex name in the conversation (for example, PLEX1).
 - b. Expand the J2EEServers folder and highlight the J2EE server name (for example, BBOASR2).
 - c. Click **Selected > Modify** to update the J2EE server definition.
 - d. Under the Environment variable list section of the J2EE server definition, double-click any environment variable (for example, the CLASSPATH variable) to open the Environment Editing Dialog window and modify the following environment variables:

CLASSPATH

Add the full directory name for the file `imsico.jar` as noted in “Step 2: Configure IMS Connector for Java on the target system” on page 271. To do so, complete the following steps:

Under the Name drop-down box in the Environment Editing Dialog window, select the **CLASSPATH** variable.

Click **Modify**.

Append the full directory name for the file `imsico.jar` to the current classpath in the Variable Value textbox (for example, `<current_classpath>:/usr/lpp/connectors/imsico.jar`).

LIBPATH

Add the full name of the work directory that contains the `libimsico.so` file as noted in “Step 2: Configure IMS Connector for Java on the target system” on page 271. To do so, complete the following steps:

Under the Name drop down box in the Environment Editing Dialog window, select the **LIBPATH** variable.

Click **Modify**.

Append the work directory that contains the `libimsico.so` file to the current libpath (for example, `<current_libpath>:/usr/lpp/connectors`).

- e. Click **OK** to close the Environment Editing Dialog window (to complete the environment variable changes).
 - f. Click **Selected > Save**.
9. Add an IMS Connector for Java Connection Factory as a new J2EE Server Resource:
- a. Select the **J2EE Resources** folder under your sysplex name (for example, `PLEX1`).
 - b. Click **Selected > Add** and wait for the J2EE resource description fields to appear in the right panel.
 - c. In the J2EE Resource Name text box, enter the name that will represent an IMS connection factory resource for this example (for example, `IMSRes`).
 - d. Set the J2EE Resource type to `IMSConnectionFactory`.
 - e. Click **Selected > Save**. Wait for the completion message (for example, `BBON0515I J2EE Resources IMSRes was added.`) to appear in the status text box above the status bar in the Administration window.
 - f. Expand the J2EE Resources folder.
 - g. Expand the resource name that was added (for example, `IMSRes`).

- h. Select the **J2EE Resource Instances** folder.
- i. Click **Selected > Add** and wait for the J2EE resource instance description fields to appear in the right panel.
- j. In the IMSConnectionFactory text box, enter the name of an instance of an IMS connection factory resource (for example, IMSRes_Instance).
- k. Verify that the appropriate System name is displayed in its drop-down box (for example, SY1).
- l. Fill in the appropriate value for the remaining fields:

Datastore name

The name of the target IMS datastore. It must match the ID parameter of the Datastore statement that is specified in the IMS Connect configuration member when IMS Connect is installed. It also serves as the XCF member name for IMS during internal XCF communications between IMS Connect and OTMA. You must replace the default value myDStrNm with a value that is valid for your IMS environment.

Datastore name is a 1 to 8 alphanumeric character name of the target IMS datastore defined in IMS Connect. **Note:** This information can be acquired by executing the nn VIEWHWS command, where nn is the response number to the IMS Connect reply message on the host.

Group name

The default IMS group name that will be used for all connections created by this Connection Factory. Whether or not you provide a value for GroupName will depend on the level of security checking in place for your installation. If, additionally, you wish to use the IVP to verify your installation's security environment, and your security environment requires that you use GroupName, you must replace the default value of 8 blanks with a valid value.

GroupName is a 1 to 8 alphanumeric character value to be used as the default GroupName for the user if neither the container nor the application provides a group name to IMS Connector for Java.

Host name

This field is the TCP/IP host name of the target IMS Connect. This property applies only when using TCP/IP communication between IMS Connector for Java and IMS Connect to run the IVP.

Note: This property is ignored if the IMS Connect name property is specified or if you are using Local Option.

IMS Connect name

The IMS Connect name must match the 'HWS ID' of the target IMS Connect. This property applies only when using Local Option communication between IMS Connector for Java and IMS Connect to run the IVP. Fill out this field with the appropriate information. A 1 to 8 alphanumeric character name of the target IMS Connect.

Note: If you are using TCP/IP, you must leave this field blank.

LogWriter Recording

You can either enable or disable LogWriter recording for use by IMS Connector for Java in WebSphere Application Server for z/OS and OS/390.

Password

The default password that will be used for all connections created by this Connection Factory. Whether or not you provide a value for Password will depend on the level of security checking in place for your installation. If, additionally, you wish to use the IVP to verify your installation's security environment, you must replace the default value of 8 blanks with a valid value.

Password is a 1 to 8 alphanumeric character value to be used as the default password for the user if neither the container nor the application provides a password to IMS Connector for Java.

Port number

This field is the TCP/IP port number of the target IMS Connect. This property applies only when using TCP/IP communication between IMS Connector for Java and IMS Connect to run the IVP.

User name

The default security authorization facility (SAF) user name that will be used for all connections created by this Connection Factory. For example, for some installations this would be the RACF user ID. Whether or not you provide a value for UserName will depend on the level of security checking in place for your installation. If, additionally, you wish to use the IVP to verify your installation's security environment, you must replace the default value of 8 blanks with a valid value.

UserName is a 1 to 8 alphanumeric character value to be used as the default name for the user if neither the container nor the application provides a user name to IMS Connect for Java.

Trace level

This is the level of information to be traced. See the Javadoc for the `IMSTraceLevelProperties` class or *IMS Connector for Java User's Guide and Reference*, SC27-1559, for a description of trace levels.

- m. Click **Selected > Save**. Wait for the completion message (for example, BBON0515I J2EE Resource instance IMSRes_Instance was added.) to appear in the status text box above the status bar in the Administration window.

Step 5: Deploy the IMS Connector for Java IVP

The IVP is packaged as an Enterprise Archive (EAR) file. An EAR file represents a J2EE application that can be deployed in a WebSphere application server. This file contains one Web module and its WebSphere configuration files.

This Web module consists of one Web Archive (WAR) file that packages its deployment configuration files and the following basic components:

```
IMSIcoIVPServlet.class  
IMSIcoIVPServletError.jsp  
IMSIcoIVPServletResults.jsp  
IMSIcoIVPServletInput.html
```

Before you begin, you must have the IVP EAR file (`IMSIcoIVPzOS.ear`) in a directory of a workstation system that can be accessed by your WebSphere for z/OS Administration application.

This IVP EAR file has been installed with the IMS Connector for Java runtime component on your host z/OS or OS/390 server. For more information on this, see “Step 1: Install the runtime component on the target system” on page 271. You can download this IVP EAR file to your workstation machine. Be sure to download it in binary format. In the instructions below, the IVP EAR file is assumed to be the `c:\Temp` directory of your workstation.

To deploy the IVP EAR file in your WebSphere Application Server for z/OS and OS/390 environment, complete the following steps in the same conversation in the WebSphere for z/OS Administration application that you started in “Step 4: Deploy and configure a J2EE server resource” on page 272.

1. Install the enterprise application:
 - a. Select the J2EE server on which this application will be installed (for example, BBOASR2). Select this server by expanding the following folders in order: Conversations folder, the conversation (for example, IMSICO), the Sysplexes folder, the sysplex in which your sample J2EE server has been installed (for example, PLEX1), and the J2EEServers folder.

- b. Click **Selected > Install J2EE application...** and wait for its reference window to appear (for example, the Install J2EE application on server: BBOASR2 window).
 - c. Click **Browse EAR File...** and wait for the Open window to appear. Locate and select your EAR file (for example, select the IMSICOIVPzOS.ear file that you downloaded in the C:\Temp directory). Click **Open**.
 - d. Verify that the destination and EAR file names appear in the EAR Filename text field. Click **OK** and wait for the Reference and Resource Resolution window to appear.
 - e. In the Reference and Resource Resolution dialog box, do this:
 - i. On the left side of the dialog, under the IMSICOIVPApp folder, expand the imsicovp_WebApp.jar folder and select the J2EE application bean (for example, imsicovp_WebApp).
 - ii. On the right side of the dialog, click the **EJB** tab and then click **Set Default JNDI Path & Name**.
 - iii. Click the **J2EE Resource** tab, click the empty box under J2EE Resource, then select the Connection Factory you just created for the Server (for example, IMSRes). Click **OK**.
2. You have now deployed IMS Connector for Java as J2EE resources. You have also deployed the IMS Connector for Java IVP. You can complete and activate the changes in the conversation to your WebSphere Application Server for z/OS and OS/390 with the following steps:
 - a. Validate the new conversation you just created in the previous step by selecting the conversation name (for example, IMSICO) to highlight it and clicking **Build > Validate**.
 - b. Commit the new conversation by selecting the conversation name to highlight it and clicking **Build > Commit**.
 - c. Ensure that all preparation tasks for the new conversation are completed. Then select the conversation name to highlight it and click **Build > Complete > All Tasks** to indicate completion.
 - d. Activate the new conversation by selecting the conversation name to highlight it and clicking **Build > Activate**.
3. Configure the HTTP server for the context root of the IVP by modifying the following files:
 - a. The webcontainer.conf file, with the following information:


```
host.default_host.contextroots=/
```
 - b. The httpd.conf file, by adding the following information:


```
Service
```

```
/imsicoivp/*<was_dir>/WebServerplugin/bin/was400plugin.so:service_exit
```

where <was_dir> is the installation root directory of WebSphere Application Server (for example, /usr/lpp/WebSphere).

Step 6: Start the server to run the IVP

The following steps show how to start the Web and WebSphere Application servers to run the IVP servlet from a browser.

1. Start the J2EE server:
 - a. In the WebSphere Application Server for z/OS and OS/390 Operations window, highlight the instance of the J2EE server (for example, BBOASR2A).
 - b. Click **Selected > Start**.
 - c. Wait for a message indicating that the server has been issued a start command (for example, BBON0600I Start issued for server instance BBOASR2A.) to appear in the status text box above the status bar in the Administration window.
 - d. Click **View > Refresh**.
 - e. Wait for a green check mark to appear over the icon of the instance of the J2EE server indicating that the server has become active.

Note: The server can be started manually by issuing the command:

```
/S BBOASR2.BBOASR2A
```

2. Start the Web server:
 - a. Issue the command to start the Web server on the host machine. For example (if the default Web server setup is used):

```
/S IMWEBSRV
```
 - b. Wait for a message indicating that the Web server has completely started. For example, there may be messages similar to these:

```
IMW3534I PID:16777263 SERVER STARTING
IMW3536I SA 16777263 0.0.0.0:8080 **READY
```

Step 7: Run the IVP to access the target IMS

1. From a Web browser, enter the following URL:

```
http://<servername>:<port>/imsicoivp/IMSICOIVPServletInput.html
```

where servername and port are the name and port number for your Web server machine and imsicoivp is the context root of the IVP Web module.

2. Click **Submit** on the input form. This action causes the IMS /STA OTMA command to be sent to IMS using the properties you specified in the Connection Factory named IMSRes.

The objective of the IVP is to receive a message from IMS in reply to the /STA OTMA command. One of two possible messages is returned, depending on the level of security checking in place for your installation and whether the UserName/Password/GroupName is permitted to issue the /STA command. Either message indicates successful execution of the IVP, since it is IMS that issues the "DFS" message.

```
DFS1292E SECURITY VIOLATION
DFS058I 17:33:32 START COMMAND COMPLETED
```

The IVP does not run an IMS application program. Therefore, it does not verify that IMS transactions can run successfully. It only verifies that the path to IMS is available.

Troubleshooting IVP failures

If an error occurs, check for information in WebSphere Application Server's logs in the folder \WebSphere\AppServer\logs. This file lists exceptions and stack traces that are useful for diagnosing the failure.

In addition, the following are possible causes for IVP failure:

- ▶ In the webcontainer.conf file, verify that host.default_host.alias points to the correct IP address of the WebSphere Server.
- ▶ Verify that the webcontainer.conf file includes the appropriate context root (for example, host.default_host.contextroots=/).
- ▶ Verify that the httpd.conf file has the Service Directives set up for the IVP sample. For example:

```
Service /imsicoivp/*
    /usr/lpp/websphere/WebServerPlugin/bin/was400plugin.so:service_exit)
```

- ▶ The IBM Http Server is not started.
- ▶ The WebSphere Application Server used for the IVP (for example, default_server) is not started.
- ▶ Invalid data was specified when configuring the IMS Connector for Java Configuration Factory as J2EE Resource instance. For example:
 - The IMS Connect name (for Local Option communication) is incorrect.
 - HostName is misspelled or not sufficiently qualified (for TCP/IP communication).
 - An incorrect PortNumber is specified for the target IMS Connect (for TCP/IP communication).

- The datastore name is invalid for the target IMS or is misspelled. The datastore name must be in uppercase characters.
- ▶ IMS is not running.
- ▶ IMS Connect is not running.
- ▶ The IMS Connect port is not active. Use the IMS Connect command VIEWHWS to determine if the port is active. Use the IMS Connect command OPENPORT to activate an IMS Connect port.
- ▶ The target IMS datastore is not active. Use the IMS Connect command VIEWHWS to determine if the datastore is active. Use the IMS Connect command OPENDS to activate an IMS datastore.
- ▶ There is a TCP/IP failure. Always ensure a successful "ping" prior to running the IVP.
- ▶ The wrong level of IMS Connect is running on the host. See Prerequisites for running the IVP for the required level of IMS Connect.

7.6.6 References

For more details on IMS Connector for Java and how to build and run Java applications for IMS transactions and conversations, refer to *IMS Connector for Java User's Guide and Reference*, SC27-1559, in the pdf library in IBM Visual Age for Java, V4 on the workstation, and to the redbook *IMS e-business Connectors: A Guide to IMS Connectivity*, SG24-6514.

Also consider *IMS Connect Guide and Reference*, SC27-0946, and *IMS V7 and Java Application Programming*, SG24-6123, as valuable sources for further reading and information.

z/OS UNIX System Services enablement

This chapter introduces you to the setup and configuration of z/OS UNIX System Services in full-function mode, which is required by WebSphere.

If you specify on the OMVS= statement a BPXPRMxx member that specifies "FILESYSTEM TYPE(HFS)", the kernel services start up in full-function mode when the system is IPLed. You have to set up the full-function mode of z/OS UNIX to use HFS.

In this chapter we discuss the required steps to enable z/OS UNIX, and to run in full-function mode. These steps are:

- ▶ Installing z/OS UNIX
- ▶ Customizing z/OS UNIX
- ▶ z/OS UNIX Security
- ▶ Setting up z/OS UNIX Daemons

Some of the steps might not pertain to you. Customize the relevant tasks according to your site requirements.

Installing z/OS UNIX

In this section we explain Hierarchical File System (HFS) concepts and the methods used for installation.

Hierarchical File System concepts

A Hierarchical File System (HFS) consists of the following entities:

- Hierarchical File System files, which contain data or programs. A file containing a load module, shell script, or REXX program is called an *executable* file. Files are stored in *directories*.
- Directories, which contain files, other directories, or both. Directories are arranged hierarchically, in a structure that resembles an upside-down tree, with the root directory at the top and the branches at the bottom. The root is the first directory for the file system at the top of the tree and is designated by a slash (/).
- Additional local or remote file systems, which are mounted on directories of the root file system or additional file systems.

Figure A-1 illustrates the logical view of the HFS for end users:

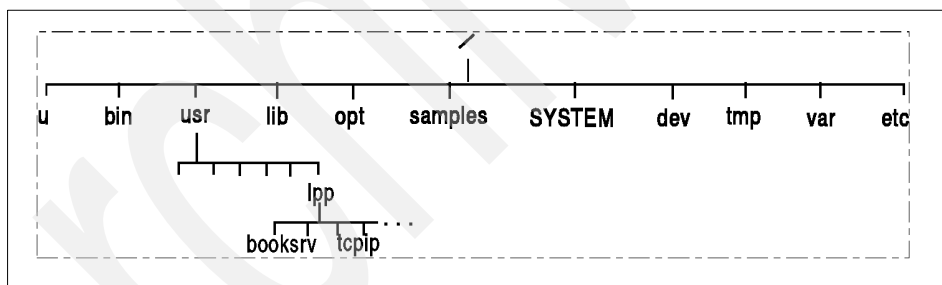


Figure A-1 Logical view of the HFS for the end user

Methods of installation

Two methods of installing z/OS are provided with your z/OS license: ServerPac and CBPDO. For each of these, *z/OS Planning for Installation* describes what IBM does for you, what you receive from IBM, and what actions you need to take. Refer to that publication for information about how to actually install z/OS UNIX, along with the other elements and features.

The custom-built *z/OS ServerPac: Installing Your Order* describes the installation jobs that you run to replace an existing system or install a new one.

For CBPDO users, the Program Directory describes how to use the SMP/E RECEIVE, APPLY, and ACCEPT commands to install your order.

ServerPac installation requires superuser (UID=0) authority. For CBPDO users, You must be a superuser with UID(0) or have access to the BPX.SUPERUSER FACILITY class.

Also refer to “Security requirements for installation/applying maintenance” on page 311 for a complete description of the security requirements necessary to perform your install.

DFSMS/MVS considerations

HFS data sets no longer needs to be SMS managed, see information APAR I112221. But SMS must be setup at least in a null configuration in order to apply PTFS that update the HFS

Customizing z/OS UNIX

To customize z/OS UNIX, perform the following tasks:

- Use the z/OS UNIX Configuration Wizard tool to help you do the customization
- Perform the security preparation
- Customize BPXPRMxx PARMLIB members
- Allocate other file systems
- Define BPXPRMxx PARMLIB members in IEASYSxx
- Customize other PARMLIB members
- Prepare PROCLIB members
- Add z/OS UNIX ISPF data sets
- Customize the Shell
- Define the Kernel to WLM
- Create HFS data sets for z/OS UNIX users
- Check for setup errors

In the following sections, we discuss these tasks in greater detail.

Use the z/OS UNIX Configuration Wizard

Use the z/OS UNIX Configuration Wizard to help you set up z/OS UNIX in full-function mode. This wizard is a Web-based tool that begins with a series of interviews, and after you finish answering all of the interview questions, you ask the wizard to build the output. Then the wizard produces a checklist of steps for you to follow, as well as customized jobs and other data sets for you to use.

After completing these tasks, you'll be able to bring up your UNIX System Services in full-function mode.

You should use this wizard to configure z/OS UNIX for the first time, or to check and verify your configuration settings. To begin with, however, you can accept the default values that are provided by the wizard.

To use the wizard, go to the Web site:

<http://www.ibm.com/servers/eserver/zseries/zos/wizards>

Then select **z/OS V1R2 UNIX Customization Wizard**.

Perform the security preparation

In order to set up security, complete the following tasks.

Define superusers

First define superusers to perform z/OS UNIX administrative activities (refer to “Defining a superuser” on page 308).

Define the kernel user ID and group

In our examples, OMVSKERN refers to the kernel user ID and OMVSGRP refers to the kernel group ID.

- Define the kernel user ID and group as follows:

```
ADDGROUP OMVSGRP SUPGROUP(SYS1) OWNER(SYS1)
ALTGROUP OMVSGRP OMVS(GID(1))
ADDUSER OMVSKERN DFLTGRP(OMVSGRP) OWNER(OMVSGRP)
ALTUSER OMVSKERN OMVS(UID(0) HOME('/') PROGRAM('/bin/sh')) NOPASSWORD
```

Important: OMVSKERN must be superuser (UID=0). By having no TSO segment on the RACF ADDUSER command and using the NOPASSWORD option, OMVSKERN is a protected user ID and cannot be used to log on to the system.

Define the started procedures

Two procedures need to be associated with a RACF user ID that is a superuser (UID=0):

- ▶ OMVS started procedure - runs a program that initializes the kernel address space
- ▶ BPXOINIT started procedure - runs the initialization process

The RACF user ID, along with a RACF group that has a valid GID, needs to be added to the RACF started class.

1. Define the OMVS cataloged procedure.

Define the OMVS cataloged procedure as *trusted*:

a. Activate the started class:

```
SETROPTS CLASSACT(STARTED) GENERIC(STARTED) RACLIST(STARTED)
```

b. Add the OMVS STARTED task:

```
RDEFINE STARTED OMVS.* STDATA(USER(OMVSKERN) GROUP(OMVSGRP) +  
TRUSTED(YES))
```

c. Refresh the STARTED class:

```
SETR GENERIC(STARTED) RACLIST(STARTED) REFRESH
```

Note: If you did not make the OMVS (the kernel) trusted, you have to give the Kernel user ID UPDATE access to any HFS data set that it needs to mount.

2. Define the BPXOINIT cataloged procedure.

Define BPXOINIT as *untrusted*:

```
RDEFINE STARTED BPXOINIT.* STDATA(USER(OMVSKERN) GROUP(OMVSGRP) +  
TRUSTED(NO))
```

```
SETR GENERIC(STARTED) RACLIST(STARTED) REFRESH
```

3. Define the BPXAS cataloged procedure:

```
ADDUSER BPXAS DFLTGRP(OMVSGRP) OWNER(OMVSGRP)  
ALTUSER BPXAS OMVS(UID(0) HOME('/') PROGRAM('/bin/sh')) +  
NOPASSWORD  
RDEFINE STARTED BPXAS.* OWNER(OMVSGRP) STDATA(USER(BPXAS) +  
GROUP(OMVSGRP) TRUSTED(NO))  
SETROPTS RACLIST(STARTED) REFRESH
```

Define the TTY group

Certain shell commands, such as **mesg**, **talk**, and **write** require pseudoterminals to have a group name of TTY. When a user logs in, or issues the OMVS command from TSO/E, the group name associated with these terminals is changed to TTY. You need to define a group named TTY to RACF

This group should be given a unique GID, such as GID(2); do not connect users to this group. Issue the following RACF command using a GID of 2:

```
ADDGROUP TTY SUPGROUP(OMVSGRP) OWNER(OMVSGRP) OMVS(GID(2))
```

Note: If the group name TTY doesn't conform to your site's naming conventions, use a different group name and place this name in the TTYGROUP statement in the BPXPRMxx PARMLIB member.

Protect HFS data sets

Create a RACF profile for the Root HFS and other HFS data sets, and give, for example, SYSPROG alter authority for maintaining these data sets.

For the Root HFS, issue the following commands:

```
ADDGROUP OMVS
ADDSD 'OMVS.**' UACC(NONE)
PERMIT 'OMVS.**' ACCESS(ALTER) ID(SYSPROG)
```

For other HFS data sets, issue the following commands:

```
ADDGROUP HFS
ADDSD 'HFS.**' UACC(NONE)
PERMIT 'HFS.**' ACCESS(ALTER) ID(SYSPROG)
```

Customize BPXPRMxx PARMLIB members

The BPXPRMxx PARMLIB member contains the parameters that control z/OS UNIX processing and the file system. The system uses these values when initializing z/OS UNIX System Services.

We recommend that you have two BPXPRMxx PARMLIB members, as this makes it easier to migrate from one release to another.

In our example, we have two BPXPRMxx members; member BPXPRMF1 will have the file system setup, and BPXPRM01 will have the system limits.

The BPXPRMF1 member is based on a single system for shared HFS; refer to *z/OS UNIX System Services Planning*, GA22-7800 .

Define the file systems

In BPXPRMF1 member, add the new HFS (root, etc.) that you just built (from the installing z/OS UNIX step) according to the sample shown in Figure A-2:

```
FILESYSTYPE TYPE(HFS)          /* Filesystem type HFS          */
      ENTRYPPOINT(GFUAINIT) /* Entrypoint for defining HFS */
      PARM(' ')              /* Null PARM for physical file */
                              /* system                      */
FILESYSTYPE TYPE(TFS)          /* Type of file system to start */
      ENTRYPPOINT(BPXTFS)      /* Entry Point of load module  */
                              /*                              */
ROOT      FILESYSTEM('OMVS.&SYSNAME..&SYSR1..ROOT.HFS')
      /* z/OS root filesystem */
      TYPE(HFS)              /* Filesystem type HFS        */
      MODE(RDWR)             /* Mounted for read/write     */
                              /*                              */
MOUNT FILESYSTEM('OMVS.&SYSNAME..ETC.HFS')
      /* HFS for /etc directory */
      MOUNTPPOINT('/etc')
      TYPE(HFS)              /* Filesystem type HFS        */
      MODE(RDWR)             /* Mounted for read/write     */
                              /*                              */
/*      UDS Socket File System - Local Sockets          */
FILESYSTYPE TYPE(UDS) ENTRYPPOINT(BPXTUINT)
NETWORK DOMAINNAME(AF_UNIX)
      DOMAINNUMBER(1)
      MAXSOCKETS(2000)
      TYPE(UDS)

/*      Single INET Socket File System - IP Network Sockets */
FILESYSTYPE TYPE(INET) ENTRYPPOINT(EZBPFINI)
NETWORK DOMAINNAME(AF_INET)
      DOMAINNUMBER(2)
      MAXSOCKETS(64000)
      TYPE(INET)
```

Figure A-2 The new BPXPRMF1 file system member

Define the system limits

To customize BPXPRM01 for system limits, you have two choices: either start with the default values in the BPXPRM00 that is shipped to you, or use the z/OS UNIX Configuration Assistant Wizard to set other values.

If you decide to start with the default values in BPXPRM00 that is shipped to you, analyze your z/OS UNIX environment, find how many interactive users will use z/OS UNIX and what type of work they will do (compile links, editing, running shell command/scripts, etc.), and what type of applications you will be running. Then make the following initial “rules of thumb” assumptions:

- ▶ Each user will consume up to double the system resources required for a TSO/E user.
- ▶ At most, four PTYs will be required per average user.
- ▶ The starting point for maximum processes per user is 25.
- ▶ Four concurrent processes will be required by an average active user.
- ▶ Five processes will be required for various daemons.
- ▶ Three concurrent address spaces will be required by the average active user. (This number will be high if your users are running with the _BPX_SHAREAS environment variable set to YES or REUSE.)

If you have a few users who need a large number of processes, you should set the process limits for these users by using the PROCUSERMAX keyword in the OMVS segment.

The above assumptions and the number of z/OS UNIX users that you have will affect the following fields in BPXPRMxx:

- ▶ MAXUIDS
- ▶ MAXPTYS
- ▶ MAXPROCUSER
- ▶ MAXPROCSYS

Note: Use this information as a *guideline* for z/OS UNIX base values. You need to check the documentation of the applications that will run in your system to find out the UNIX resources required. Then adjust the BPXPRMxx values. Setting these values either very low or very high might adversely affect your system.

Example

Assume that you want to allow 20 TSO/E users to access z/OS UNIX System Services. The initial settings might be:

- ▶ In BPXPRMxx:
 - MAXUIDS(20)
 - MAXPTYS(80)
 - MAXRTYS(80)
 - MAXPROCUSER(25)
 - MAXPROCSYS(185)

An explanation of these values follows. All values are based on a total number of 20 users.

- MAXUIDS (20)** If you allow 20 concurrent TSO/E users to access z/OS UNIX, each of them could consume twice the resource they normally used for TSO/E. This would require all your remaining system resources.
- MAXPTYs(80)** Assume that 4 PTYs are needed per user. Users can log in with multiple sessions at the same time.
- MAXPROCUSER(25)** This should normally be a reasonable starting point. Some users may require more processes, depending on the work they are doing. This value can be set only on a system-wide basis.
- MAXPROCSYS(185)** Assume that you need 4 processes per user and 5 processes for daemons: $(20 \text{ users} * 4) + 5 \text{ daemons} + 100$ for a basic TCP/IP configuration = 185 processes.

Allocating other file systems

All z/OS elements and features that store into the HFS are installed into a consolidated HFS data set, instead of having separate product-related HFS data sets. This makes maintaining and cloning the file system easier, and it simplifies the MOUNT statements in the BPXPRMxx PARMLIB member.

It is required that you maintain a separate HFS data set for each of the following directories:

- ▶ **/etc**, which contains customization data. Keeping the /etc file system in an HFS data set separate from other file systems allows you to separate your customization data from IBM's service updates. It also makes migrating to another release easier.
- ▶ **/dev**, which contains character-special files that are used when logging into the OMVS shell environment. Prior to OS/390 V2R7, these character-special files were created by running the BPXISMKD REXX exec, or would be part of your ServerPac order.

Beginning with OS/390 V2R7, /dev is shipped empty. The necessary files are created when the system is IPLed, and on a per-demand basis.

- ▶ **/tmp**, which contains temporary data that is used by products and applications. The /tmp directory is created empty, and temporary files are created dynamically by different elements and products. You have the option of mounting a temporary file system (TFS) on /tmp.
- ▶ **/var**, which contains dynamic data that is used internally by products and by elements and features of z/OS. Any files or directories that are needed are

created during execution of code (an example of this is caching data). In addition, you can be assured that IBM products will only create directories or files when code is executed.

Defining /dev and /var file systems

To define the /dev and /var file systems, perform the following tasks:

1. Allocate /dev and /var HFSs. An example of an allocation job is shown in Example A-1:

Example: A-1 Sample job to allocate HFS for /dev and /var file systems

```
//ALOC HFS JOB (MVS,POK),CLASS=A,MSGCLASS=H,
//          MSGLEVEL=(1,1),REGION=0M,NOTIFY=&SYSUID
//*
//ALLUSHFS EXEC PGM=IEFBR14
//DD1      DD DSN=OMVS.TC1.DEV.HFS,
//          DISP=(NEW,KEEP,DELETE),
//          DSNTYPE=HFS,
//          SPACE=(CYL,(10,10,1))
//DD2      DD DSN=OMVS.TC1.VAR.HFS,
//          DISP=(NEW,KEEP,DELETE),
//          DSNTYPE=HFS,
//          SPACE=(CYL,(10,10,1))
//*
```

2. Update SYS1.PARMLIB(BPXPRMF1) to add the mount statements, so that /dev and /var HFS data sets will be mounted over a system IPL:

```
MOUNT FILESYSTEM('OMVS.&SYSNAME..DEV.HFS')
      MOUNTPOINT('/dev')
      TYPE(HFS)  MODE(RDWR)

MOUNT FILESYSTEM('OMVS.&SYSNAME..VAR.HFS')
      MOUNTPOINT('/var')
      TYPE(HFS)  MODE(RDWR)
```

After the IPL, make sure that permission bit settings for the /dev and /var file systems are set to 755, to allow users on the system to read and execute files from them.

If the setting is not 755, enter **TSO OMVS**, then issue:

```
chmod 755 /var
chmod 755 /dev
```

Defining a Temporary File System (TFS) for /tmp

The /tmp (temporary) directory contains temporary data used by products and applications. The Temporary File System (TFS) is an in-memory physical file system that supports in-storage mountable file systems.

Normally, a TFS runs in the kernel address space, but it can be run in a logical file system (LFS) colony address space.

Because it is an in-storage file system, the temporary file system delivers high-speed I/O. Even if you are using kernel services in full-function mode with a hierarchical file system, you may want to mount a TFS over /tmp in an HFS, so that it can be used as a high-speed file system for temporary files.

To mount a 10 MB in-storage file system mounted over /tmp, update BPXPRMF1 with the following:

```
FILESYSTYPE TYPE(TFS) ENTRYPPOINT(BPXTFS)
MOUNT FILESYSTEM('/TMP')
      MOUNTPPOINT('/tmp')
      TYPE(TFS)
      Parm('-s 10')
```

Defining BPXPRMxx PARMLIB members in IEASYSxx

Update the OMVS definition in IEASYSxx member to the BPXPRMxx PARMLIB member suffixes:

```
OMVS=(01,F1)
```

Check the syntax of a BPXPRMxx PARMLIB member

Before doing an IPL, use SETOMVS SYNTAXC:

```
SETOMVS SYNTAXCHECK=(01)
SETOMVS SYNTAXCHECK=(F1)
```

Note: You cannot use that command to verify whether HFS data sets or mount points are valid.

Customizing other PARMLIB members

This task customizes the following PARMLIB members:

- ▶ PROGxx
- ▶ LPALSTxx
- ▶ COFVLFxx
- ▶ ALLOCxx
- ▶ CTnBPXxx

- ▶ IEADMR00
- ▶ SMFPRMxx

PROGxx

Make sure that the following Language Environment data sets are added to LINKLIST:

```
LNKLST ADD NAME(LNKLST00) DSN(CEE.SCEERUN)
LNKLST ADD NAME(LNKLST00) DSN(CEE.SCEERUN2)
```

Make sure that the following Language Environment data sets are added to the APF list:

```
APF ADD
  DSNAME(CEE.SCEERUN)
  VOLUME(*****)
```

```
APF ADD
  DSNAME(CEE.SCEELPA)
  VOLUME(*****)
```

LPALSTxx

Make sure that the following Language Environment data set is added to the LPALSTxx

```
CEE.SCEELPA
```

COFVLFxx

Update the VLF PARMLIB member COFVLFxx with the following statements. Caching of the z/OS UNIX System Services UID and GID information improves performance.

```
CLASS NAME(IRRGMAP)      /* OpenMVS-RACF GMAP table */
EMAJ(GMAP)                /* Major name = GMAP */
CLASS NAME(IRRUMAP)      /* OpenMVS-RACF UMAP table */
EMAJ(UMAP)                /* Major name = UMAP */
CLASS NAME(IRRGTS)       /* RACF GTS table */
EMAJ(GTS)                 /* Major name = GTS */
CLASS NAME(IRRACEE)      /* RACF saved ACEEs */
EMAJ(ACEE)               /* Major name = ACEE */
CLASS NAME(IRRSMAP)      /* Security packet */
EMAJ(SMAP)               /* Major name = SMAP */
```

The following operator command starts VLF, where xx indicates the updated COFVLFxx member:

```
START VLF,SUB=MSTR,NN=xx
```


ALLOCxx

Forked address spaces are perceived to be batch jobs by MVS allocation. If a forked address space attempts to allocate a data set on a volume that is not mounted, the request either waits (with or without an operator prompt), or it fails. The ALLOCxx PARMLIB member controls the behavior of allocation requests of this type. If you do not want the request to wait, specify ALLOCxx statements as follows:

```
VOLUME_ENQ POLICY (CANCEL)
VOLUME_MNT POLICY (CANCEL)
```

Use this policy so that forked address spaces do not go into allocation waits. Be aware that using this policy can disrupt your system, because it will cause a failure rather than a wait.

CTnBPXxx

The CTnBPXxx PARMLIB member specifies the tracing options for a component trace of z/OS UNIX events.

One member should control initial tracing, which automatically starts when the OMVS address space is started. It should store trace records in a buffer, which could be read if a dump is written. This member should be considered the operating system's default member. The CTRACE parameter in BPXPRMxx specifies the CTnBPXxx member.

Note: One member can be set up to trace all z/OS UNIX events.

CTIBPX00 traces minimum information:

```
TRACEOPTS
ON
BUFSIZE(4M)
```

To change the tracing to collect data needed for a particular problem, ask the operator to enter a TRACE CT command that specifies a different, customized CTnBPXxx member that you have placed in PARMLIB.

The following is an example of a member, CTCBPX06, with an OPTIONS statement that requests tracing of events in files and pipes.

```
TRACEOPTS
WTR(CTWTR)
WTRSTART(CTWTR)
ON
BUFSIZE(4M)
OPTIONS('FILE','PIPE')
```

The WTRSTART statement specifies a CTWTR cataloged procedure, which the installation wrote and which starts a component trace external writer. The buffer size is set at 4 M.

When re-creating a problem for IBM service, we generally recommend you increase the buffer size to its maximum.

IEADMR00

Change PARMLIB IEADMR00 (SYSMDUMP and core dump defaults) to specify:

`SDATA=(RGN,SUM,TRT,LPA)`

This gathers adequate data without an excessive dump size.

SMFPRMxx

The JWT value in the SMF PARMLIB member SMFPRMxx specifies how long an idle address space is allowed to wait before it is terminated.

When an address space is dubbed a process, or when a forked or spawned process is created, the process may go into signal-enabled waits. In a signal-enabled wait, the address space is made exempt from long-wait timeouts as specified by the JWT value in the PARMLIB member SMFPRMxx. This enables parent processes to wait forever while child processes are running. Otherwise, if the parent process is terminated due to job wait timeout, a SIGHUP signal is sent to the running process and work is lost.

However, shell users, whether logged on through TSO/E and the OMVS command, or via rlogin or telnet, are exempt from job wait timeout because the shell is in a signal-enabled wait while waiting for a command from the user. To have shell users be timed out and logged off, you need to specify the TMOUT environment variable in /etc/profile. The TMOUT environment variable contains the number of seconds before user input times out. If user input is not received, the shell ends.

If a shell started by the TSO/E OMVS command times out, then the TSO address becomes enabled for job wait timeout processing. This means that if you have JWT=30 (30 minutes) and you have TMOUT=600 (10 minutes), then TSO users who leave their terminals in the shell will time out and be logged off in about 40 minutes.

Preparing PROCLIB members

Make sure OMVS, BPXOINIT, and BPXAS started procedures are in your PROCLIB.

The kernel uses WLM to create child processes while running in either goal mode or compatibility mode. When programs issue `fork()` or `spawn()`, the BPXAS PROC found in SYS1.PROCLIB is used to provide a new address space. For a `fork()`, the system copies one process, called the parent process, into a new process, called the child process. The forked address space is provided by WLM.

Note: You are now able to IPL the system with the z/OS UNIX kernel in full-function mode and perform system testing. After that you can continue with the remaining z/OS UNIX customization tasks.

Adding z/OS UNIX ISPF data sets

To make certain TSO/E commands (such as OEDIT, OBROWSE, and ISHELL) and some shipped REXX execs available to users, concatenate the following libraries to the appropriate ISPF data definition names (ddnames). The following data sets are for the English panels, messages, and tables:

- ▶ SYS1.SBPXPENU concatenated to ISPLIB
- ▶ SYS1.SBPXMENU concatenated to ISPLIB
- ▶ SYS1.SBPXTENU concatenated to ISPTLIB
- ▶ SYS1.SBPXEXEC concatenated to SYSEXEC or SYSPROC

Customizing the shell

Copy the following files that are provided in the /samples into the /etc directory:

```
init.options
magic
mailx.rc
ohelp.ENU
profile
rc
startup.mk
yylex.c
yparse.c
csh.cshrc
csh.login
```

If you are migrating from an earlier release, you rename old files, copy new ones and then change the variables to your environment.

To copy the files you can use either the z/OS UNIX Shell or ISHELL.

Setting environment variables is optional. If you choose to set environment variables, the following list shows where to set them, and in the order that the system sets them:

- ▶ RACF user profile
- ▶ /etc/profile
- ▶ \$HOME/.profile
- ▶ The file named in the ENV environment variable
- ▶ A shell command or shell script

Later settings takes precedence.

The default settings of the environment variables in all five places are sufficient to run z/OS UNIX in a basic full-function mode. If you choose to customize any of the environment variables, it is most likely that you will customize them in /etc/profile and/or \$HOME/.profile.

You may need to change the following -e environment variable options in /etc/init.options depending on where you are located geographically:

- ▶ -e TZ (timezone)
- ▶ -e LANG (language)
- ▶ -e NLSPATH (locale)

Alternatively, you can specify the TZ, NLSPATH and LANG operands in /etc/profile.

For more information see “Customizing the z/OS UNIX Environment” in *z/OS UNIX System Services Planning*, GA22-7800 .

Prioritizing the kernel

There are two Workload Manager (WLM) modes for prioritizing kernel work in your system: goal mode and compatibility mode. This section discusses both modes.

The nice() and setpriority() functions use definitions in BPXPRMxx for performance groups (compatibility mode) and goals (goal mode). These definitions are optional, but if they are not specified, the nice() and setpriority() functions do not change performance level. The following are some reasons for enabling nice() and setpriority() functions:

- ▶ If you are running applications that require the ability to control the priority of different processes, you must define appropriate priority levels for the application to use. This is typically done for real-time systems that are dedicated to running a single application.
- ▶ If you have enabled the batch, at, and cron shell functions, you need to define priority groups or goals that are appropriate for running batch jobs as in a UNIX system.

Running in goal mode

Installations that run in goal mode can take the following steps to customize service policies in their WLM definition:

- ▶ Define a workload for kernel work.
- ▶ Define service classes for kernel work:
 - Define a service class for forked children. You should specify a number of performance periods. Performance periods for short-running work can be given response-time goals or percentage response-time goals. Performance periods for long-running work should be given velocity goals.
 - Define a service class for startup processes, which are forked by the initialization process BPXOINIT. This service class should be given a velocity goal that is higher than that of other forked children.
- ▶ Define classification rules:
 - By default, put forked child processes (subsystem type OMVS) into the service class defined for forked children.
 - Put the kernel (with TRXNAME=OMVS) into a high-priority Started Task (subsystem type STC) service class. Another option is to keep the OMVS started procedure in the default started class category, which generally has high priority.
 - Put the initialization process BPXOINIT (with TRXNAME=BPXOINIT) into a high-priority Started Task (subsystem type STC) service class. Another option is to keep the BPXOINIT started procedure in the default started class category, which generally has high priority.
 - Startup processes that are forked by the initialization process, BPXOINIT, fall under SUBSYS=OMVS. These processes are identified by USERID=OMVSKERN. Put them in a separate service class as defined above.
 - Other forked child processes (under subsystem type OMVS) can be assigned to different service classes based on USERID, ACCTINFO, or TRXNAME.
 - Put the DFSMS buffer manager SYSBMAS (with TRXNAME=SYSBMAS) into a high-priority Started Task (subsystem type STC) service class. Another option is to allow the SYSBMAS started procedure to remain in the default started class category, which generally has high priority.

Defining service classes for kernel work

Define a service class for forked child address spaces. This service class should normally have three performance periods, because it must support all types of kernel work, from short interactive commands to long-running background work.

You can set duration values using the service-units-per-second value reported in the RMF Monitor I Workload Activity report. If your OMVS work is meeting performance objectives in SRM compatibility mode, you can use the VELOCITY field in the Workload Activity report to set velocity goals for each period.

The following is a sample service class for forked children. Change these values as appropriate for your installation.

* Service Class OMVS - OMVS forked children

Base goal:

#	Duration	Imp	Goal description
1	2000	2	Response Time 80% 1 second
2	4000	3	Response Time 60% 2 seconds
3		5	Execution velocity of 10

Also, define a service class for daemons. This service class should normally have only one period with a velocity goal higher than the velocity goals of other forked children.

* Service Class OMVSKERN - OMVS startup processes

Base goal:

#	Duration	Imp	Goal description
1		1	Execution velocity of 40

Your installation may have other special classes of users. If so, you may want to define other service classes for kernel work.

If you have used the PRIORITYGOAL statement in the BPXPRMxx PARMLIB member to enable the nice(), setpriority(), and chpriority() functions, additional service classes for kernel work must be added

Defining classification rules as needed

Specify the classification rules needed to separate daemons (for example, inetd) from other forked children. The following is a sample classification for subsystem type OMVS:

* Subsystem Type OMVS

Classification:

Default service class is OMVS
There is no default report class.

Qualifier # type	Qualifier name	Starting position	Service Class	Report Class
1 UI	OMVSKERN		OMVSKERN	

Forked spaces can be classified by transaction name.

If you do not define any classification rules, OMVS and BPXOINIT will run under the rules for subsystem type STC, which typically is defined to have high priority.

If needed, you can define a classification rule for subsystem type STC to ensure that the kernel, the initialization process BPXOINIT, and the DFSMS buffer manager SYSBMAS run as a high-priority started tasks. In the following example, STC1 is a service class for high-priority started tasks:

* Subsystem Type STC

Classification:

Default service class is STC2.

There is no default report class.

Qualifier #	Qualifier type	Qualifier name	Starting position	Service Class	Report Class
1	TN	OMVS		STC1	
1	TN	BPXOINIT		STC1	
1	TN	SYSBMAS		STC1	
.	.	.		STC1	
.	

Running in compatibility mode

Installations that run in compatibility mode should set up performance groups for kernel services by customizing the IEAICSxx and IEAIPSxx PARMLIB members. If you do not, the system puts all forked processes in the system default performance group and this may result in a wait condition at startup.

- Update the IEAICSxx PARMLIB member.
- Update the IEAIPSxx PARMLIB member to specify performance attributes for the performance groups added to IEAICSxx.

Note: If you change these system PARMLIB members, use the SET ICS and SET IPS operator commands to put the members into effect.

Customizing the IEAICSxx PARMLIB member

The IEAICSxx PARMLIB member specifies installation control for the work in the system. Example A-2 is a simplified IEAICSxx member with statements needed for kernel services. Statements related to the kernel services are shown in bold highlighting.

Example: A-2 Parameters in the IEAICSxx PARMLIB member

```
SUBSYS=STC,PGN=9
  TRXNAME=OMVS,PGN=10      /* OpenMVS kernel */
  TRXNAME=BPXOINIT,PGN=10  /* OpenMVS INIT process */
  TRXNAME=SYSBMAS,PGN=11   /* DFSMS buffer manager */
.
SUBSYS=TSO,PGN=2
```

```

USERID=SUPER1,PGN=20
.
SUBSYS=OMVS,PGN=5          /* OpenMVS forked children */
USERID=OMVSKERN,PGN=40     /* OpenMVS startup processes */
USERID=SUPER1,PGN=50       /* Special for user super1 */
ACCTINFO=D001(1),PGN=60    /* Special for acct D001 */

```

In the SUBSYS=STC section:

- ▶ TRXNAME=OMVS specifies a performance group for the kernel address space. The kernel is nonswappable, but its performance group and dispatching priority are defined by the IEAICSxx and IEAIPSxx PARMLIB members.
- ▶ TRXNAME=BPXOINIT specifies the performance group for the OMVS initialization process.
- ▶ TRXNAME=SYSBMAS specifies the performance group for the DF/SMS buffer manager, used for HFS I/O.

Note: A TRXNAME statement is the jobname for the OMVS address space. By default, fork and spawn set jobname values to the user ID with a number (1-9) appended. However, daemons or users with appropriate privileges can set the `_BPX_JOBNAME` environment variable to change the JOBNAME for forked or spawned children. In this way, servers and daemons in OMVS address spaces can easily be assigned different performance attributes than other OMVS address spaces. TRXNAME requires OS/390 Release 5.

The SUBSYS=OMVS section assigns performance groups to forked address spaces. These performance group assignments do not apply to dubbed address spaces, such as batch programs that issue z/OS callable services. If a TSO/E, batch, or started task address space uses kernel services, it does not change subsystem type; that is, it does not use SUBSYS=OMVS.

In the SUBSYS=OMVS section, use USERID, ACCTINFO, and TRXNAME statements to assign performance groups. The section cannot contain TRXCLASS parameters.

- ▶ USERID statements can be used to provide different users with different performance groups. Specify USERID=OMVSKERN to provide a performance group for startup processes, those which are forked by the kernel or by the initialization process BPXOINIT.

A forked child address space inherits the user ID of its parent. It may still be in a different performance group. For example, if TSO/E user SUPER1 with PGN=20 issues a fork, the forked address space has PGN=50 from the USERID=SUPER1 statement in the SUBSYS=OMVS section.

- ▶ ACCTINFO statements: Fork processing always propagates accounting data from the parent to the child. In addition, when a daemon creates a process for another user, accounting data is taken from the WORKATTR segment of the RACF user profile. A daemon can specify account information in the environment variables passed to the spawn or exec callable services.

In the example, the forked child processes have been placed in performance group 60. The ACCT section begins with D001.

- ▶ TRXNAME statements can be used to isolate OMVS work by jobname. By default, fork and spawn processing sets jobname values to the user ID with a number (1-9) appended. However, users with appropriate privileges can set the _BPX_JOBNAME environment variable to assign a different jobname. This way, servers or daemons running in forked or spawned address spaces can easily be assigned different performance attributes than other SUBSYS=OMVS address spaces.

Customizing the IEAIPSxx PARMLIB member

The IEAIPSxx PARMLIB member provides installation performance specifications for the performance groups in the IEAICSxx PARMLIB member; see Example A-3:

Example: A-3 Parameters for z/OS UNIX in the IEAIPSxx PARMLIB member

```

DMN=40,CNSTR=(4,8), ...           /* OpenMVS init processes */
PGN=9,(DMN=9,DP=F83)
PGN=10,(DMN=10,DP=F81)           /* OpenMVS kernel and BPX0INIT */
PGN=11,(DMN=10,DP=F81)          /* SYSBMAS */ . . .
PGN=2,(DMN=2,DP=F54,DUR=400)
    (DMN=3,DP=F52,DUR=1400)
    (DMN=4,DP=F44)
PGN=20,(DMN=20,DP=F54,DUR=400)
    (DMN=3,DP=F52,DUR=1400)
    (DMN=4,DP=F44)

PGN=5,(DMN=5,DP=F53,DUR=30K)      /* OpenMVS forked children */
    (DMN=6,DP=F51,DUR=200K)
    (DMN=7,DP=M4)
PGN=40,(DMN=40,DP=F64)           /* OpenMVS startup processes */
PGN=50,(DMN=5,DP=F54,DUR=30K)    /* Special for user super1 */
    (DMN=6,DP=F52,DUR=200K)
    (DMN=7,DP=M5)
PGN=60,(DMN=5,DP=F54,DUR=30K)    /* Special for acct D001 */
    (DMN=6,DP=F51,DUR=200K)
    (DMN=7,DP=F43)

```

Make sure all the sample domains have been defined.

When customizing the IEAIPSxx PARMLIB member statements:

- ▶ Give the kernel and the OMVS initialization process, BPXOINIT, high priority.
- ▶ The priority of the following started task, which is the DFSMS buffer manager, must be higher than that of any users accessing HFS files:

```
SUBSYS=STC,TRXNAME=SYSBMAS
```

- ▶ Allow the startup processes a multiprogramming level (MPL) high enough to support the /usr/sbin/init process, and any forked child processes (daemons). Daemons that run forever should be in a performance group with a single domain.
- ▶ The MPL value for started tasks must be high enough to include all fork initiators provided by WLM.
- ▶ Provide normal forked children with more than one performance period. Work can range from quick, built-in shell commands to long-running background processes. The service requirements for some of these address spaces may be similar to medium-length TSO/E transactions, and for others, similar to long-running batch jobs.
- ▶ If you have used the PRIORITYPG statement in the BPXPRMxx PARMLIB member to enable the nice, setpriority, and chpriority functions, additional performance groups must be added.

Note: We recommend that you refer to the tuning performance chapter of *z/OS UNIX System Services Planning*, GA22-7800 for more information. You can also refer to the following site:

<http://www.ibm.com/servers/eserver/zseries/zos/unix>

Creating HFS data sets for z/OS UNIX users

Keep system HFS data sets separate from user HFS data sets by creating separate SMS storage groups to segregate the HFS data sets, or by allocating non-SMS managed HFS data sets on separate volumes.

The following is a sample JCL to allocate a user HFS:

```
//LABHFS1B JOB (MVS,P0K),CLASS=A,MSGCLASS=H,  
//          MSGLEVEL=(1,1),REGION=0M,NOTIFY=&SYSUID  
//*  
//UNIXUSR EXEC PGM=IEFBR14  
//DD1 DD DSN=OMVS.SYS7.AMR.HFS,  
//      DISP=(NEW,KEEP,DELETE),  
//      DSNTYPE=HFS,  
//      SPACE=(CYL,(10,10,1))  
/*
```

Making user file systems available

After the user's HFS data set is allocated, you need to mount it at a mount point off the root directory to make it available. The preferred place to mount all user HFS data sets is the /u mount point. In z/OS, there are two ways to accomplish this:

- ▶ Direct mount
- ▶ Automount facility (the preferred method to manage user HFS data sets because it saves administration time)

We recommend using the Automount facility. For more information, refer to *z/OS UNIX System Services Planning*, GA22-7800 .

Customizing user directories

Perform the following steps to customize each user directory:

1. Copy the .profile from your site customized member or from /samples to the user directory, for example:

```
cp /samples/.profile /u/amr/.profile
```

2. Change the permission bit settings for the new user directory and the ownership of all the files in that directory /u/userid, so that they are owned by the user ID (you must be a superuser to issue these commands):

```
cd /u/amr  
chmod 755 .  
chown -R AMR .
```

(Remember to include the dot in these commands.)

Checking for setup errors

After you complete the customization process, you might want to run the Setup Verification Program (SVP) to check for potential setup errors.

Check the z/OS UNIX Web site for the program:

```
http://www.ibm.com/servers/eserver/zseries/zos/unix/bpxa1svp.html
```

z/OS UNIX security

Defining security is an important part of setting up z/OS UNIX. Any program or user requesting z/OS UNIX functions needs to have a valid user identifier (UID) and a valid group identifier (GID) before they can use any z/OS UNIX System Services.

The UID/GID is a numeric value between 0 and 2 147 483 647.

Userid	Default Group	Connect Groups		TSO	DFP	OMVS		
						UID	Home	Program
SMITH	PROG1	PROG1	PROG2	15	/u/smith	/bin/sh

Group profile

Groupid	Superior Group	Connected Users				OMVS GID
PROG1	PROGR	SMITH	BROWN	25

Group profile (no OMVS segment)

Groupid	Superior Group	Connected Users			
PROG2	PROGR	SMITH	WHITE

Figure A-3 RACF OMVS segments

HFS security is based on permission bits associated with an HFS file, UID and GID values associated with the file, and the requesting RACF user ID. The user's UID is a number specified in the RACF user ID OMVS segment. The user's GID is a number specified in the OMVS segment of his/her RACF default group or current connect group, as illustrated in Figure A-3.

When a GID is assigned to a group, all users connected to that group who have a z/OS UNIX user identifier (UID) in their user profile can use z/OS UNIX functions and can access z/OS UNIX files based on the GID and UID values assigned.

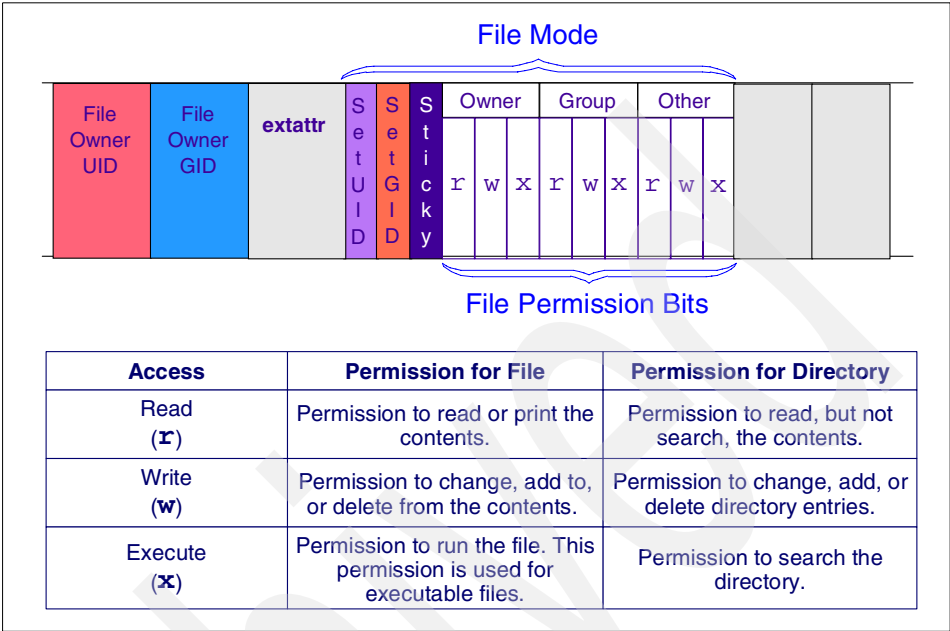


Figure A-4 UNIX security - file security packet

If a z/OS UNIX user tries to access an HFS file, the user's UID and GID are compared with the UID and GID associated with the file. Depending on whether the values are equal, z/OS UNIX grants the access rights of the file owner, the owner's group, or the rights that are granted to everyone, as shown in Figure A-4.

Figure A-5 on page 308 shows the UNIX file authorization checking decision tree.

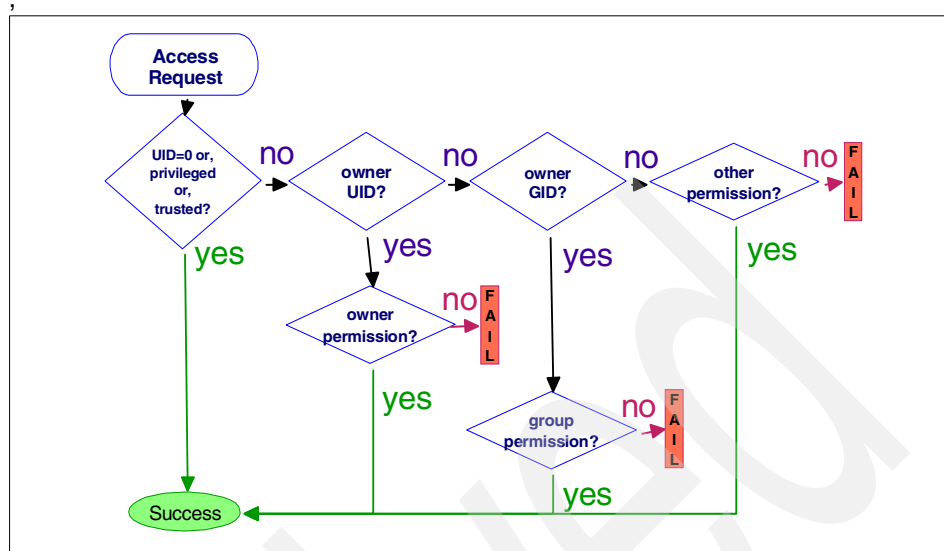


Figure A-5 UNIX file authorization checking

A SAMPLIB member, BPXISEC1, is provided with z/OS UNIX. This sample TSO/E CLIST provides all the RACF commands needed for the security setup discussed throughout this chapter.

Note: In this chapter, only RACF definitions are used. The commands listed are RACF TSO commands, and they have to be issued by a user with SPECIAL authority. The UNIX tasks requires superuser authority.

Defining a superuser

A superuser can do the following:

- ▶ Pass all security checks and access any file in the file system.
- ▶ Manage processes.
- ▶ Have an unlimited number of processes running concurrently. (For a started procedure, this is true only if it has a UID of 0. It is not true for a trusted or privileged process with a different UID.)
- ▶ Change identity from one UID to another.
- ▶ Use `setrlimit()` to increase any of the system limits for a process.

The UID of a parent process and the UID's trusted or privileged attributes are propagated to a forked child process. Thus, a UID of 0 is propagated to a forked child.

Assigning superuser privileges

Superuser privileges can be assigned in three ways:

- ▶ By assigning a UID of 0 (the least desirable way)
- ▶ By using the BPX.SUPERUSER FACILITY class profile
- ▶ By using the UNIXPRIV class profiles (the preferred way)

While some functions require a UID of 0, in most cases you can choose among the three ways. In making this choice, try to minimize the number of “human” user IDs (as opposed to started procedures) set up with UID(0) superuser authority.

The three methods of assigning superuser privileges are explained in more detail in the following sections. In our examples, SYSPROG refers to the system programmer user ID that will have superuser authority and that is connected to group SYS1.

Assigning a UID of 0

Execute the following commands to assign a user UID 0:

```
ALTGROUP SYS1 OMVS(GID(3))  
ALTUSER SYSPROG OMVS(UID(0)) HOME('/') PROGRAM('/bin/sh'))
```

The number of user IDs that you assign a UID of 0 depends on your site setup. We recommend you keep the number of user IDs with a UID of 0 to a minimum.

Using the BPX.SUPERUSER FACILITY class profile

Give users with a non-zero UID who require superuser authority read access to the BPX.SUPERUSER FACILITY class profile. Then they can use the **su** command to switch to superuser authority (effective UID if 0). For example, we want user ID AMR with a non-zero UID to be able to temporarily switch to become a superuser when it is required for administrative tasks:

1. Define the BPX.SUPERUSER profile:

```
RDEFINE FACILITY BPX.SUPERUSER UACC(NONE)
```

2. Permit users who need superuser to BPX.SUPERUSER:

```
PERMIT BPX.SUPERUSER CLASS(FACILITY) ID(AMR) ACCESS(READ)
```

3. Refresh the FACILITY class in memory:

```
SETROPTS GLOBAL(FACILITY) RACLIST(FACILITY) GENERIC(FACILITY) REFRESH
```

Using UNIXPRIV class profiles

Starting with OS/390 R8, you can reduce the number of people who have superuser authority at your installation by defining profiles in the UNIXPRIV class that grant RACF authorization for certain z/OS UNIX privileges. By defining profiles in the UNIXPRIV class, you can specifically grant certain superuser privileges with a high degree of granularity to users who do not have superuser authority. This allows you to minimize the number of assignments of superuser authority at your installation and reduce your security risk.

The resource names in the UNIXPRIV class are associated with z/OS UNIX privileges. Global access checking is not used for authorization checking to UNIXPRIV resources.

Following are two examples of authorizing superuser privileges to allow a user to perform superuser functions without being a superuser with UID=0.

1. To allow a user to use the **chown** command to change ownership of any file:

- a. Activate the UNIXPRIV class, if it is not currently active at your installation:

```
SETROPTS CLASSACT(UNIXPRIV)
```

- b. Define a profile in the UNIXPRIV class to protect the resource called SUPERUSER.FILESYS.CHOWN:

```
RDEFINE UNIXPRIV SUPERUSER.FILESYS.CHOWN UACC(NONE)
```

- c. Assign selected users or groups as appropriate:

```
PERMIT SUPERUSER.FILESYS.CHOWN CLASS(UNIXPRIV) +  
ID(appropriate-groups-and-users) ACCESS(READ)
```

- d. You must activate SETROPTS RACLIST processing for the UNIXPRIV class, if it is not already active:

```
SETROPTS RACLIST(UNIXPRIV)
```

If SETROPTS RACLIST processing is already in effect for the UNIXPRIV class, you must refresh SETROPTS RACLIST processing in order for new or changed profiles in the UNIXPRIV class to take effect.

```
SETROPTS RACLIST(UNIXPRIV) REFRESH
```

2. To allow a user to issue **mount** and **unmount** commands without being a superuser with UID=0:

- a. Define a profile in the UNIXPRIV class to protect the resource called SUPERUSER.FILESYS.MOUNT:

```
RDEFINE UNIXPRIV SUPERUSER.FILESYS.MOUNT UACC(NONE)
```

- b. Authorize selected users or groups as appropriate:

```
PERMIT SUPERUSER.FILESYS.MOUNT CLASS(UNIXPRIV) +  
ID(appropriate-groups-and-users) ACCESS(READ)
```


SETOPTS RACLIST(UNIXPRIV)

For more information about the UNIXPRIV class, refer to *z/OS UNIX System Services Planning*.

Security requirements for installation/applying maintenance

In order to install into the HFS, the user ID that you use must be a superuser and be connected to a group that has a GID.

- ▶ For ServerPac customers, the user ID must be UID=0.
- ▶ For CBDPO customers, the user ID may be UID=0, or permitted to the BPX.SUPERUSER FACILITY class.

In addition, regardless of your installation method, the user ID must be permitted read access to the FACILITY classes BPX.FILEATTR.APF and BPX.FILEATTR.PROGCTL (or BPX.FILEATTR.* if you choose to use a generic facility for both facility classes).

To define these FACILITY classes, use the following commands:

```
RDEFINE FACILITY BPX.FILEATTR.APF UACC(NONE)
RDEFINE FACILITY BPX.FILEATTR.PROGCTL UACC(NONE)
SETOPTS CLASSACT(FACILITY)
SETOPTS RACLIST(FACILITY)
PERMIT BPX.FILEATTR.APF CLASS(FACILITY) ID(your_userid) ACCESS(READ)
PERMIT BPX.FILEATTR.PROGCTL CLASS(FACILITY) ID(your_userid) ACCESS(READ)
SETOPTS RACLIST(FACILITY) REFRESH
```

Or, if you choose to use a generic facility, use these commands:

```
SETOPTS GENERIC(FACILITY)
RDEFINE FACILITY BPX.FILEATTR.* UACC(NONE)
SETOPTS CLASSACT(FACILITY)
SETOPTS RACLIST(FACILITY)
PERMIT BPX.FILEATTR.* CLASS(FACILITY) ID(your_userid) ACCESS(READ)
SETOPTS RACLIST(FACILITY) REFRESH
```

- ▶ In order to apply maintenance using SMP/E, the user ID should have UID=0 and be permitted read access to the FACILITY classes BPX.FILEATTR.APF and BPX.FILEATTR.PROGCTL (or BPX.FILEATTR.*).

Defining RACF group identifiers and user identifiers

RACF does not require the GID to be unique. However, we recommend you do not assign the same GID to multiple RACF groups, because you will lose control at the group level. During z/OS UNIX security checks, the GID is used and the RACF groups that have the same GID assignment are treated as a single group.

Define a GID to the RACF group

Define GIDs for z/OS UNIX groups with the following command:

```
ALTGROUP usrgp1 OMVS(GID(gid))
```

Following is an example:

```
ALTGROUP SYS2 OMVS(GID(5))
```

Define z/OS UNIX users to RACF

Regular z/OS UNIX users are those users that have a UID set to a value other than zero. Now that you have defined a group with an OMVS GID assigned to it, the next step is to define the OMVS segment in the user ID profile.

You can use different criteria to map UIDs. For example, if you have a group with a GID of 200, you could give all users belonging to this group a UID between 200 and 299. We recommend that you assign all users a unique non-zero UID. You can use the following command to do this:

```
ADDUSER user1 DFLTGRP(usrgp1) OMVS(UID(uid) HOME('/u/user1') +  
PROGRAM('/bin/sh'))
```

Following is an example:

```
ADDUSER AMR DFLTGRP(SYS2) OMVS(UID(15) HOME('/u/amr') PROGRAM('/bin/sh'))
```

Note: RACF allows UIDs/GIDs within the range of 0 to 2147483647. However, the `pax` and `tar` utilities cannot handle values above 16777216. Using `pax` or `tar` to copy files with UIDs/GIDs above 16777216 may result in an incorrect assignment of UIDs/GIDs to the restored files.

So, because `pax` and `tar` are commonly used utilities, consideration should be given to the potential occurrence of this problem in your environment before assigning UIDs/GIDs above 16777216. Using the USTAR format can avoid this, but only if the target system has the same user or group name defined.

Controlling access to HFS files and directories

The system provides security for HFS files by verifying that a z/OS UNIX user can access a directory, a file, and every directory in the path to the file.

Every file and directory has security information, which consists of:

- ▶ File access permissions
- ▶ UID and GID of the file
- ▶ Audit options that the file owner can control
- ▶ Audit options that the security auditor can control

The file access permission bits that accompany each file provide discretionary access control (DAC). These bits determine the type of access a user has to a file or directory.

The access permission bits are set for three classes: owner, group, and other. When a user's process accesses a file, the system determines the class of the process and then uses the permission bits for that class to determine if the process can access the file.

For a file, a process can be in only one class. The class for a process can be different for each file or directory.

The class of a user's process is one of the following:

- ▶ **Owner class:** Any process with an effective UID that matches the UID of the file.
- ▶ **Group class:** Any process with an effective GID or supplemental group GID that matches the GID of the file when the UIDs do not match.
- ▶ **Other class:** Any process that is not in the owner or group class, such as when neither the UIDs nor the GIDs match.

The system sets the UID and GID of the file when the file is created:

- ▶ The UID is set to the effective UID of the creating process.
- ▶ The GID is set to the GID of the owning directory.

Figure A-4 on page 307 shows the types of access and the permissions granted by the accesses.

To access HFS files, users need the following:

- ▶ Read and search permission to all directories in the pathnames of files the user should use. Read permission is required for some options of some commands.
- ▶ Write permission to all directories in which the user will be creating files or directories.
- ▶ Read permission, write permission, or read and write permission, as appropriate to all files that the user needs to access.
- ▶ Execute permission to executable files the user needs to run.

Security verification

To check the OMVS definitions for SYSPROG user ID, issue the following command; the results are shown in Figure A-6 on page 314.

```
1u sysprog omvs
```

```

USER=SYSPROG  NAME=AMR KHAFAGY          OWNER=WELLIE2   CREATED=99.071
DEFAULT-GROUP=SYS1      PASSDATE=99.071  PASS-INTERVAL=N/A
ATTRIBUTES=SPECIAL OPERATIONS
REVOKE DATE=NONE  RESUME DATE=NONE
LAST-ACCESS=01.288/13:04:03
CLASS AUTHORIZATIONS=NONE
NO-INSTALLATION-DATA
NO-MODEL-NAME
LOGON ALLOWED  (DAYS)          (TIME)
-----
ANYDAY                      ANYTIME
GROUP=SYS1      AUTH=CONNECT  CONNECT-OWNER=WELLIE2  CONNECT-DATE=99.071
CONNECTS=      73  UACC=NONE   LAST-CONNECT=01.288/13:04:03
CONNECT ATTRIBUTES=NONE
REVOKE DATE=NONE  RESUME DATE=NONE
SECURITY-LEVEL=NONE SPECIFIED
CATEGORY-AUTHORIZATION
NONE SPECIFIED
SECURITY-LABEL=NONE SPECIFIED

OMVS INFORMATION
-----
UID= 0000000000
HOME= /
PROGRAM= /bin/sh
CPUTIMEMAX= NONE
ASSIZEMAX= NONE
FILEPROCMAX= NONE
PROCUSERMAX= NONE
THREADSMAX= NONE
MMAPAREAMAX= NONE

```

Figure A-6 LU command - results

You will notice that SYSPROG has UID=0 since it is a superuser.

Note: If you encounter any access problems, check the syslog first to identify the problems.

Setting up a default OMVS segment

Some users need to access socket services but do not need other UNIX services, so rather than having the security administrator define UIDs and GIDs for all USER and GROUP profiles involved, a default OMVS segment can be defined instead.

The default OMVS segment resides in a USER profile and a GROUP profile. The names of these profiles are selected by the installation, using a profile in the FACILITY class. The name of the FACILITY class profile is BPX.DEFAULT.USER. The application data field in this profile contains the user ID, or the user ID/group ID, of the default profiles.

When you define the BPX.DEFAULT.USER profile, all users will be able to access z/OS UNIX, so you have to plan your security very carefully. To prevent certain users from being able to access z/OS UNIX, you can define an OMVS segment with no UID. Then the dub fails when those users attempt to use a UNIX service.

If users must be dubbed (for example, for FTP or other socket use) but you do not want them to use the shell, consider defining the initial program for the default user as /bin/echo. Then users with the default UID will not be able to use the shell.

To set up the default OMVS segment, perform the following steps:

1. Define a group ID that will be used as an anchor for the default OMVS group segment, as follows:

```
ADDGROUP OMVSLTG OMVS(GID(777777))
```

When defining the default group, the main consideration is what you put in the OMVS segment. You might want to make this GID sufficiently different so that it stands out when used. You should make it very high. The other fields related to this group ID are not likely to be used for anything.

2. Define a default user ID that will be used as an anchor for the default OMVS user segment, as follows:

```
ADDUSER OMVSLTU DFLTGRP(OMVSLTG) NAME('OMVS DEFAULT USER') NOPASSWORD +  
OMVS(UID(999999) HOME('/tmp') PROGRAM('/bin/echo'))
```

When you define the default user, consider the following entries:

- ▶ **UID:** You might want to make the UID different from other z/OS UNIX UIDs. You should make it very high—and do not set the UID of the default user to 0.
- ▶ **HOME:** There are several options when defining the home directory for the default user:
 - Define a directory just like any other user. This directory would then be used concurrently by many users that do not have an OMVS segment. However, this is *not* recommended.
 - Define the HOME directory where the default user does not have write permission, such as the root (/). The user should not need read and execute permission to the home directory.

- Define the HOME directory in the /tmp directory where nothing important is ever held.

- ▶ **PROGRAM:** Define the default shell in this field. If you do not want users running in a shell environment with the default UID and GID, define the PROGRAM parameter as:

```
PROGRAM('/bin/echo')
```

This will cause any attempt to enter the shell to terminate.

In addition to UID, HOME, and PROGRAM, the user limits value from the default OMVS segment will be used. If you expect to have a lot of users running with the default segment, you might want to set the user limits higher than the system limits to accommodate them.

3. Create a FACILITY class BPX.DEFAULT.USER profile. Place the name of a default user ID (or user ID/group ID) in the application data field of that profile, as follows:

```
RDEFINE FACILITY BPX.DEFAULT.USER APPLDATA('OMVSLTU/OMVSLTG')
SETROPTS RACLIST(FACILITY) REFRESH
```

To set up a default for the USER OMVS segment only, the format of the command is:

```
RDEFINE FACILITY BPX.DEFAULT.USER APPLDATA('OMVSLTU')
SETROPTS RACLIST(FACILITY) REFRESH
```

You cannot set up a default GROUP OMVS segment alone. If you take the time to set up individual OMVS segments for all users, it's likely that you would also want to set up OMVS segments for groups because there are usually far fewer groups than users defined to RACF.

The BPX.DEFAULT.USER FACILITY class profile is used as follows:

- ▶ A user requests a kernel service; the kernel then dubs the user.
- ▶ The kernel calls the security product to extract the UID, GID, HOME, and PROGRAM information.
- ▶ RACF attempts to extract the OMVS segment associated with the user. If it is not defined, it tries to extract and use the OMVS segment for the default user that was listed in the BPX.DEFAULT.USER profile.
- ▶ A similar process occurs to obtain a GID when the user's default group does not have an OMVS segment.

Setting up BPX.* FACILITY class profiles

You may need to define these FACILITY class profiles:

► **BPX.DAEMON**

This restricts access to the following services:

- seteuid (BPX1SEU service)
- setuid (BPX1SUI service)
- setreuid (BPX1SRU service)
- spawn (BPX1SPN service with a change in user ID requested)

Only daemons can use these services without first having specified the target user ID password on the passwd() service. In order to change the MVS identity without knowing the target user ID's password, the caller of these services must be a superuser. Additionally, if a BPX.DAEMON FACILITY class profile is defined and the FACILITY class is active, the caller must be permitted to use this profile.

► **BPX.DEBUG**

Users with READ access to BPX.DEBUG FACILITY can use ptrace (via dbx) to debug programs that run with APF authority or with BPX.SERVER authority.

► **BPX.FILEATTR.APF**

This controls which users are allowed to set the APF-authorized attribute in an HFS file. This authority allows the user to create a program that will run APF-authorized. This is similar to the authority of allowing a programmer to update SYS1.LINKLIB or SYS1.LPALIB.

► **BPX.FILEATTR.PROGCTL**

This controls which users are allowed to set the program control attribute in an HFS file. Programs marked with this attribute can execute in server address spaces that run with a high level of authority.

► **BPX.FILEATTR.SHARELIB**

This indicates that extra privilege is required when setting the shared library extended attribute via the chattr() callable service. This prevents the shared library region from being misused.

► **BPX.JOBNAME**

This controls which users are allowed to set their own job names by using the _BPX_JOBNAME environment variable or the inheritance structure on spawn. Users with READ or higher permissions to this profile can define their own job names.

► BPX.SAFFASTPATH

This enables faster security checks for file system and IPC constructs.

► BPX.SERVER

This restricts the use of the `pthread_security_np()` service. A user with read or write access to the BPX.SERVER FACILITY class profile can use this service. It creates or deletes the security environment for the caller's thread. This profile is also used to restrict the use of the BPX1ACK service, which determines access authority to z/OS resources.

► BPX.SMF

This checks if the caller attempting to cut an SMF record is allowed to write an SMF record. It also tests if an SMF type or subtype is being recorded.

► BPX.SRV. userid

This allows users to change their UID if they have access to BPX.SRV. userid, where userid is the MVS user ID associated with the target UID. BPX.SRV. userid is a RACF SURROGAT FACILITY class profile.

► BPX.STOR.SWAP

This controls which users can make address spaces nonswappable. Users permitted with at least READ access to BPX.STOR.SWAP can invoke the `__mlockall()` function to make their address space either nonswappable or swappable.

When an application makes an address space nonswappable, it may cause additional real storage in the system to be converted to preferred storage. Because preferred storage cannot be configured offline, using this service can reduce the installation's ability to reconfigure storage in the future. Any application using this service should warn the customer about this side effect in their installation documentation.

► BPX.SUPERUSER

This allows users to switch to superuser authority. For more information about BPX.SUPERUSER, see "Defining a superuser" on page 308.

► BPX.WLMSEVER

This controls access to the WLM server functions `_server_init()` and `_server_pwu()`. It also controls access to these C language WLM interfaces:

- QuerySchEnv()
- CheckSchEnv()
- DisconnectServer()
- DeleteWorkUnit()
- JoinWorkUnit()
- LeaveWorkUnit()

- ConnectWorkMgr()
- CreateWorkUnit()
- ContinueWorkUnit()

A server application with read permission to this FACILITY class profile can use the server functions, as well as the WLM C language functions, to create and manage work requests.

Setting up z/OS UNIX daemons

A *daemon* is a long-living process that runs unattended to perform continuous or periodic system-wide functions, such as network control. Daemons have superuser authority and can issue authorized functions such as `setuid()`, `seteuid()` and `spawn()` to change the identity of a user's process. Some daemons are triggered automatically to perform their task, and other daemons operate periodically.

z/OS UNIX supplies the following daemons:

- `inetd` - the network daemon
- `rlogind` - the remote login daemon
- `cron` - the clock daemon
- `uucpd` - the UUCP daemon

The `syslogd` daemon, which is used to route messages, is shipped with z/OS Communications Services.

Setting up security

To define security for a daemon, complete the following steps:

1. Define a daemon user ID as superuser.

We can run all or some daemons with the kernel user ID that has already been defined.

2. Define the BPX.DAEMON FACILITY class, as follows:

```
RDEFINE FACILITY BPX.DAEMON UACC(NONE)
```

3. Grant DAEMON authority to the daemon user ID.

Depending on how you start your daemons, some of them inherit their identities from the kernel address space if they were started from the `/etc/rc` shell script that is executed during the startup of z/OS UNIX. In this case give DAEMON authority to the kernel, as follows:

```
PERMIT BPX.DAEMON CLASS(FACILITY) ID(OMVSKERN) ACCESS(READ)
```

Refresh the FACILITY class:

```
SETOPTS GLOBAL(FACILITY) RACLIST(FACILITY) GENERIC(FACILITY) REFRESH
```

Note: You can assign a user ID to the daemon that is different from the kernel, in order to improve security auditing. See “Defining a daemon with a user ID other than the kernel” on page 321.

4. Activate RACF program control.

When you load a program that requires daemon authority into an address space, it must be a controlled program. All HFS programs must be program controlled.

However, programs loaded from MVS libraries do not have to be program controlled if the BPX.DAEMON.HFSCTL FACILITY class profile has been set up (see “Setting up HFS control” on page 321). In that case, only HFS files are checked for program control.

If a program that is not a controlled program is loaded, the address space is marked dirty and cannot perform daemon activities.

Perform the following tasks:

a. Activate program control, as follows:

```
SETOPTS WHEN(PROGRAM)
```

b. Protect load libraries:

```
ADDSD 'CEE.SCEERUN' UACC(READ)
ADDSD 'SYS1.LINKLIB' UACC(READ)
ADDSD 'TCPIP.SEZALINK' UACC(READ)
ADDSD 'TCPIP.SEZATCP' UACC(READ)
```

c. Mark the data sets as controlled libraries:

```
RDEFINE PROGRAM * ADDMEM('CEE.SCEERUN'//NOPADCHK +
                           'SYS1.LINKLIB'//NOPADCHK +
                           'TCPIP.SEZALINK'//NOPADCHK +
                           'TCPIP.SEZATCP'//NOPADCHK) UACC(READ)
```

d. Refresh the profile in memory:

```
SETOPTS WHEN(PROGRAM) REFRESH
```

5. Define the BPXROOT user ID.

The BPXROOT user ID is used when a daemon process invokes `setuid()` to change the UID to 0 and the user name has not been previously identified by `getpwnam()` or by the `_passwd()` function.

To define the BPXROOT user ID, issue:

```
ADDUSER BPXROOT OMVS(UID(0) HOME('/') PROGRAM('/bin/sh')) +
DFLTGRP(OMVSGRP) NOPASSWORD
```

The NOPASSWORD option indicates that BPXROOT is a protected user ID that cannot be used to enter the system by using a password. The user ID will not be revoked due to invalid logon attempts.

In the SUPERUSER statement in the BPXPRMxx PARMLIB member, specify the user ID that the kernel will use when you need a user ID for UID(0). For example, SUPERUSER(BPXROOT); if you do not specify the SUPERUSER statement, the default is BPXROOT.

Note: Do not permit the BPXROOT user ID to the BPX.DAEMON FACILITY class profile. This prevents the granting of daemon authority to a superuser who is not defined to the BPX.DAEMON FACILITY class profile.

Defining a daemon with a user ID other than the kernel

To run a daemon with a user ID other than kernel (for example, to run the uucpd daemon with user ID UUCPD), do the following:

1. Change the line in /etc/inetd.conf to:

```
uucp stream tcp  nowait  UUCPD /usr/sbin/uucpd uucpd -l0
```

2. Define user ID UUCPD to RACF:

```
ADDUSER UUCPD DFLTGRP(OMVSGRP) OMVS(UID(0) HOME('/') PROGRAM('/bin/sh')) +  
NOPASSWORD
```

3. Permit user ID UUCPD to the BPX.DAEMON FACILITY class profile:

```
PERMIT BPX.DAEMON CLASS(FACILITY) ID(UUCPD) ACCESS(READ)  
SETROPTS GLOBAL(FACILITY) RACLIST(FACILITY) GENERIC(FACILITY) REFRESH
```

Setting up HFS control

If you want only HFS files to be checked for program control, and do not want programs loaded from MVS libraries to be checked, you can set up the BPX.DAEMON.HFSCTL FACILITY class profile.

Steps for setting up HFS control

1. Make sure that the BPX.DAEMON is active.
2. Define the resource profile:

```
RDEFINE FACILITY BPX.DAEMON.HFSCTL UACC(NONE)
```

3. Give READ access to users:

```
PERMIT BPX.DAEMON.HFSCTL CLASS(FACILITY) ID(uuuuuu) ACCESS(READ)  
SETROPTS RACLIST(FACILITY) REFRESH
```

Daemon start options

Daemons can be started by JCL or by the shell.

- From the shell:

Put the command in `/etc/rc` to start the daemon automatically during kernel initialization:

```
_BPX_JOBNAME='SYSLOGD' /usr/sbin/syslogd -f /etc/syslog.conf
```

- As a cataloged procedure (started task):

```
//SYSLOGD PROC
//SYSLOGD EXEC PGM=SYSLOGD,REGION=30M,TIME=NOLIMIT,
// PARM='POSIX(ON) ALL31(ON)/ -f /etc/syslogd.conf'
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//SYSOUT DD SYSOUT=*
//SYSERR DD SYSOUT=*
//CEEDUMP DD SYSOUT=*
```

Note: For the cataloged procedure to get control with superuser and daemon authority, define it in the `STARTED` class.

To start `syslogd`, issue the following command from the MVS console:

S SYSLOGD

- As a cataloged procedure using `BPXBATCH` to invoke a daemon program located in the HFS:

```
//SYSLOGD PROC
//SYSLOGD EXEC PGM=BPXBATCH,REGION=30M,TIME=NOLIMIT,
// PARM='PGM /usr/lpp/tcpip/sbin/syslogd -f /etc/syslogd.conf'
/* STDIN and STDOUT are both defaulted to /dev/null
//STDERR DD PATH='/etc/log',PATHOPTS=(OWRONLY,OCREAT,OAPPEND),
// PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
```

Note: `syslogd` requires superuser and daemon authority.

Evaluating virtual storage needs

The kernel services use storage based on expected use as defined by the `BPXPRMxx PARMLIB` member, as well as by actual use. Here are guidelines to help you avoid running out of storage.

Extended Common Service Area (ECSA)

Use of ECSA is based on the following formulas:

```
#tasks_using_Openmvs * 150 bytes
#processes * 500 bytes
#dubbed_address_space * 500 bytes
```

For example, if your system supports 200 dubbed address spaces, 500 processes, and 2000 threads, the kernel service consumes an additional 650 KB of ECSA.

In addition to this ECSA usage:

- ▶ Workload Manager (WLM) also uses some ECSA for each initiator to satisfy a fork request.
- ▶ The OMVS address space uses an additional 20 KB of ECSA. The kernel also uses ECSA to process spawn requests. This storage is freed when no longer needed. Allocate an additional 100 K of ECSA for spawn usage.
- ▶ Each process that has a STEPLIB that is propagated from parent to child or across an exec will consume about 200 bytes of ECSA. If STEPLIBs are used for all processes and you have 400 processes, an additional 80 K of ECSA is required.

Extended System Queue Area (ESQA)

Kernel services use ESQA in support of several functions. Some of the ESQA usage can be predicted using formulas, and other areas can only be estimated. The following functions consume ESQA:

1. Signaling uses SRBs to notify the target of a signal. Signaling frequency is usually not very high and the SRBs are short-lived.

For most installations, additional ESQA does not need to be allocated in order to support signalling. If you run applications that use signals frequently, increase your ESQA allocation.

2. Using asynchronous socket services causes SRBs to be allocated. Allocate an additional 100 KB of ESQA if there is heavy use of asynchronous socket services.
3. The following functions use an MVS service called IARVSERV:
 - ptrace(), debugger support
 - shmat(), shared memory attach
 - mmap(), memory map files
 - fork() when fork is using Copy on Write (COW) mode
 - dllload(), when it is loading a user-shared library program

For each real page of shared storage affected by IARV SERV, RSM allocates a 32-byte anchor block in ESQA. For each virtual page connected to a shared real page, RSM allocates a 32-byte control block in ESQA. The following formulas and examples should assist you in predicting ESQA usage:

- a. Using the `dllload()` function to load a user-shared library program (filename with a suffix of `.so`) invokes IARV SERV to set up a map between the user address space and the kernel shared library region data space. The `dllload()` function to load a system-shared library program does not use as much ESQA as user-shared library programs.

As an example, if a `dllload()` of a user-shared library program that is 8 MB in length is done from 50 address spaces, it will consume:

$8\text{MB} * 256 \text{ pages/MB} * 52 \text{ connections} * 32 \text{ bytes/page}$
or 3.4 MB of ESQA

The 52 connections derive from 50 address spaces, 1 anchor block, and 1 connection to a kernel data space used to manage the storage

- b. Shared memory, `shmat()`, is typically used by server address spaces to communicate with clients.

The `__IPC_MEGA` option enables applications to use large quantities of shared memory without the system overhead described later in this section. If you have applications taking advantage of this `__IPC_MEGA` support, you do not need to be concerned with the following calculations. For example, if a server not using `__IPC_MEGA` allocates 8 MB of shared memory and has 500 clients connected to it, it will consume:

$8\text{MB} * 256 \text{ pages/MB} * 503 \text{ connections} * 32 \text{ bytes/page}$
or 33MB of ESQA

The 503 derives from 500 clients, 1 server, 1 anchor block, and 1 connection to a kernel data space used to manage the storage. The following examples have similar numbers.

Some servers like Lotus Notes use large amounts of shared memory that is shared by hundreds or thousands of clients. This can require large amounts of ESQA (up to one gigabyte).

- c. `mmap()` is typically used by a single process to map a file into virtual memory using the same sort of logic used by Data in Virtual (DIV). Used in this manner, each page of the file requires 3 RSM control blocks (anchor block, user page, and kernel data space page). Each additional user sharing a `mmap` page of a file will consume an additional control block.

The `__MAP_MEGA` option has been added to `mmap()`. It enables applications to map very large files without the system overhead in ESQA. If you have applications using the `__MAP_MEGA` option, you do not need to be concerned with the following calculation.

If you are not using `__MAP_MEGA` and issue `mmap()` for a 5 MB file:

$5\text{MB} * 256 \text{ pages/MB} * 3 * 32 \text{ bytes/page}$
or 123KB of ESQA

- d. `fork()` uses `IARV SERV` to capture the parent's pages for the child's use. Each page captured represents two pages involved (parent and child - P/C in the example, and an anchor block/page).

Since the child usually issues the `exec` call soon after the `fork`, the ESQA used is short-term. This is countered by the probability that there are multiple forks going on concurrently.

Assuming the Language Environment Run-Time Library is not in LPA, a typical shell will have 5 MB of private to copy on fork. If there are, on average, 10 forks running concurrently, then the following ESQA is needed:

$5\text{MB} * 256 \text{ pages/MB} * 10 \text{ forks} * 3 \text{ (P/C)} * 32 \text{ bytes/page}$
or 1.2MB of ESQA

If the run-time library does reside in LPA and each process has an average of 1 MB of private to copy, then:

$1\text{MB} * 256 \text{ pages/MB} * 10 \text{ forks} * 3 \text{ (P/C)} * 32 \text{ bytes/page}$
or .24MB of ESQA

- e. `ptrace()` uses captured storage to allow the debugger to map the program being debugged into private storage the debugger can refer to frequently. If a programmer is debugging a 1 MB program and a 200 KB automatic data stack that are both captured, you will need the following amount of storage:

$1.2\text{MB} * 256 \text{ pages/MB} * 3 * 32 \text{ bytes/page}$
or 29KB of ESQA

To predict the amount of ESQA required to support applications, you need to understand which applications use `shmat()`, `mmap()`, and `dllload()`. You need to approximate the amount of `fork()` and `dbx` debugger activity as well. Then plug your numbers into the equations for each function to determine the amount of ESQA needed.

In `BPXPRMxx`, specify the maximum number of shared storage pages that can be used on the `MAXSHAREPAGES` statement. By limiting the amount of shared storage pages used, `MAXSHAREPAGES` lets an installation control the amount of ESQA storage that is consumed by users.

This limit applies to the `mmap()`, `shmat()`, `ptrace()`, and `fork()` callable services, as well as the `dllload()` of shared library services.

The `fork()` and `ptrace()` callable services use shared storage pages to improve performance, as does a `dllload()` of shared library services. Because use of shared storage pages is not critical to completion of these functions, when the amount of shared storage pages in use reaches about 60% of the specified limit, these functions no longer use shared storage pages. The `mmap()` service continues to use the shared storage pages until the total resource consumption reaches about 80% of the limit. The `shmat()` callable service continues to use shared storage pages until the total resource consumption reaches the specified limit.

The `mmap()` and `shmat()` callable services return an out-of-memory condition when they can no longer obtain shared storage without exceeding their respective shared storage limits.

There is also a `FORKCOPY` parameter in `BPXPRMxx` that prevents `fork` from using the `IARVSERV` function.

Reducing the amount of ESQA needed to support servers

To reduce the excessive amounts of ESQA that are required to support a server like System Authorization Facility (SAF), which needs to access more than 2 GB of storage, you can use the following services:

- ▶ `_map_init`, which invokes the map service function
- ▶ `_map_service`, which enables applications to create new data blocks and to specify which map area block is to be used to view the new data block

TCP/IP enablement

In this appendix we discuss the steps required for TCP/IP enablement. We assume that TCP/IP is not customized on your system, and you have to customize it and get it running for z/OS UNIX System Services. The steps to prepare your TCP/IP environment are as follows:

- ▶ Customize the TCP/IP basic configuration
- ▶ Customize IBM-supplied daemons

Using the wizard

IBM provides a Web-based wizard called the z/OS IP Configuration Wizard. You can use it at any time to configure a single stack with simple instances of OMROUTE, FTP, and TN3270 servers, and with all device types, including static VIPA. If you use the wizard to complete the tasks defined in the output checklist, you'll be ready to use z/OS TCP/IP to communicate with other hosts in your network.

The z/OS IP Configuration Wizard can be found at:

<http://www.ibm.com/servers/eserver/zseries/zos/wizards/>

Customize the TCP/IP basic configuration

In the following sections, we describe the customizations you need to perform.

Customize TCP/IP configuration files

The TCP/IP stack uses two configuration files, PROFILE.TCPIP and TCPIP.DATA. PROFILE.TCPIP is used only for the configuration of the TCP/IP stack. TCPIP.DATA is used during configuration of both the TCP/IP stack and applications. The search order used to find TCPIP.DATA is the same for both the TCP/IP stack and applications.

PROFILE.TCPIP search order

- a. //PROFILE DD DSN=aaa.bbb.ccc(anyname).
We recommend that you use //PROFILE.
- b. *jobname.nodename.TCPIP*
- c. *hlq.nodename.TCPIP*
- d. *jobname.PROFILE.TCPIP*
- e. TCPIP.PROFILE.TCPIP

The search stops if one of these data sets is found.

TCPIP.DATA search order

1. If the resolver GLOBALTCPIPDATA statement has been configured, the TCPIP.DATA statements defined by it will be used first. The search for other TCPIP.DATA statements will continue as listed below.
2. The MVS data set or HFS file that is identified in the RESOLVER_CONFIG environment variable.

This environment variable is passed as a parameter to the TCP/IP stack in the TCP/IP started procedure JCL used to start TCP/IP. The following is an example of specifying this environment variable in the JCL:

```
//TCPIP    PROC  PARMS='CTRACE(CTIEZB00)'  
//TCPIP    EXEC  PGM=EZBTCPIP,REGION=0M,TIME=1440,  
//          PARM=('&PARMS',  
//*          'ENVAR("RESOLVER_CONFIG=//''TCPIP.TCPPARMS(TCPDATA)''')'  
//*
```

3. /etc/resolv.conf

This is the file /etc/resolv.conf that resides in the HFS.

4. //SYSTCPD DD DSN=ddd.eee.fff(anyname)

The //SYSTCPD DD card can be specified in the TCP/IP started procedure JCL.

5. userid.TCPIP.DATA, where userid is the user ID that is associated with the current security environment for the TCP/IP address space.
6. SYS1.TCPPARMS(TCPDATA)
7. hlq.TCPIP.DATA

The default hlq distributed with TCP/IP is the string TCPIP. So, effectively, this is TCPIP.TCPIP.DATA in the search order. This has implications for installations using multiple TCP/IP address spaces.

If the SYSTCPD DD is used, it is only searched for if the RESOLVER_CONFIG environment variable is not set and the HFS file /etc/resolv.conf does not exist.

In our example, we allocated a partitioned data set SYS1.TCPPARMS. We put the PROFILE.TCPIP definitions in a member called PROFILE, and the TCPIP.DATA definitions in a member called TCPDATA.

Obtain the following values from your network administrator:

- ▶ Your system IP address and hostname
- ▶ IP address for the virtual identifier (subnet value), subnet mask
- ▶ The device address of the z/OS network interfaces and adapter number
- ▶ The address of the router
- ▶ Your Domain name
- ▶ The IP address of the name resolution server

Customizing PROFILE.TCPIP

During TCP/IP address space initialization, a configuration profile data set (PROFILE.TCPIP) reads system operation and configuration parameters. Before you start your TCP/IP system, you have to define your TCP/IP configuration parameters in the PROFILE configuration file.

The PROFILE data set contains the following major groups of configuration parameters:

- ▶ TCP/IP operating characteristics
- ▶ TCP/IP physical characteristics
- ▶ TCP/IP reserved port number definitions (application configuration)
- ▶ TCP/IP network routing definitions
- ▶ TCP/IP diagnostic data statements

Copy the sample TCPIP.SEZAINST(SAMPPROF), and modify the following statements with your environment values:

- ▶ DEVICE and LINK statements identify your network interface to z/OS TCP/IP.
 - DEVICE: Defines the name (and sometimes the device number) for various types of network devices

- LINK: Defines a network interface to be associated with a particular device

In our case, LCS1 is the device name, 3002 is the hexadecimal device number, and TR1 is the name of the link; refer to Example B-1 on page 330.

- ▶ BEGINROUTES statement adds static routes to the IP route table. The BEGINROUTES statement is an alternative for the GATEWAY statement.
- ▶ The START statement starts your network interface.
- ▶ The HOME statement provides the list of home addresses and associated link names (called the HOME list).
- ▶ AUTOLOG: Supplies TCP/IP with the procedure names to start.
- ▶ The PORT statement is used to reserve port numbers for TCP/IP applications. It indicates to TCP/IP which ports are to be reserved and which applications are going to use them.

In our examples, we run two Telnet daemons: one for TN3270 (TSO access), and one for z/OS UNIX access. They cannot both listen on the same well-known port if you are running one TCP/IP stack and your system has one IP address, so we defined port 623 for the TN3270 Telnet daemon and port 23 for the z/OS UNIX Telnet daemon.

The special jobname of OMVS indicates that the PORT is reserved for any application—with the exception of those that use the Pascal API.

Example: B-1 TCPIP Profile data set sample

```

; PROFILE.TCPIP
;
DEVICE LC1 LCS 3002
LINK TR1 IBMTR 11 LC1
;
BEGINROUTES
ROUTE 9.12.3.64 255.255.255.240 = TR1 MTU 2048
ROUTE DEFAULT 9.12.3.65 TR1 MTU DEFAULTSIZE
ENDROUTES
;
START LC1
;
HOME
9.12.3.66 TR1
;
; AUTOLOG
; FTPD JOBNAME FTPD1 ; FTP Server
; ENDAUTOLOG
;
PORT
20 TCP OMVS NOAUTOLOG ; FTP Server

```

21 TCP FTPD1	; FTP Server
23 TCP OMVS	; TELNET SERVER
623 TCP INTCLIEN	; TN3270 Server
512 TCP OMVS	; Remote Execution Server
513 TCP OMVS	;
513 UDP OMVS	;
514 TCP OMVS	;
514 UDP OMVS	;
515 TCP LPSEVER	; LPD Server

Setting up the TN3270 Telnet server

The TN3270 Telnet server acts as an interface between IP and SNA networks. End users in an IP network connect to the server, which is also a VTAM application. Follow these steps to set up the TN3270 Telnet Server:

1. Customize TCPIP PROFILE.

a. TELNETPARMS

TELNETPARMS are used to define the TCP port (we used 623) and other parameters for Telnet 3270 sessions; refer to Example 7-18.

Example 7-18 TELNETPARMS statement in PROFILE.TCPIP

```

;
; TELNETPARMS: Configure the Telnet Server
;
; - TN3270(E) server port 623 options
;
TelnetParms
  Port 623 ; Port number 623
  CodePage ISO8859-1 IBM-1047 ; Linemode ASCII, EBCDIC code pages
  Inactive 0 ; Let connections stay around
  PrtInactive 0 ; Let connections stay around
  TimeMark 600
  ScanInterval 120
; SMFinit std
; SMFterm std
  WLMClusterName
    TN3270E
  EndWLMClusterName
EndTelnetParms
;

```

b. BEGINVTAM

This defines the VTAM parameters required for the Telnet server; refer to Example 7-18 on page 331.

```
;
; BEGINVTAM: Defines the VTAM parameters required for the Telnet server.
;
BeginVTAM
  Port 623 ;
  ; Define logon mode tables to be the defaults shipped with the
  ; latest level of VTAM
  TELNETDEVICE 3278-3-E NSX32703 ; 32 line screen -
                                ; default of NSX32702 is 24
  TELNETDEVICE 3279-3-E NSX32703 ; 32 line screen -
                                ; default of NSX32702 is 24
  TELNETDEVICE 3278-4-E NSX32704 ; 48 line screen -
                                ; default of NSX32702 is 24
  TELNETDEVICE 3279-4-E NSX32704 ; 48 line screen -
                                ; default of NSX32702 is 24
  TELNETDEVICE 3278-5-E NSX32705 ; 132 column screen-
                                ; default of NSX32702 is 80
  TELNETDEVICE 3279-5-E NSX32705 ; 132 column screen -
                                ; default of NSX32702 is 80

  ; Define the LUs to be used for general users.
  DEFAULTLUS
    TCP00001..TCP00015
  ENDDEFAULTLUS
EndVTAM
```

2. Customize VTAM definitions for Telnet:

- a. Add VTAM APPL definitions for the TCP/IP LUs that were previously defined in the TCPIP.PROFILE, in the DEFAULTLUS statement.

Create new member TCP in SYS1.VTAMLST to contain the following definitions:

```
*
* VTAMLST SAMPLE DEFINITION
*
TCP      VBUILD TYPE=APPL
TCP*     APPL AUTH=NVPACE,EAS=1,PARSESS=NO,MODETAB=ISTINCLM,SESSLIM=YES
```

Add the TCP member to the startup list in SYS1.VTAMLST(ATCCON00), so that the TCP/IP Telnet definitions are activated at IPLs:

```
*/ *
*/ * LIB: SYS1.VTAMLST(ATCCON00)
*/ *
COSAPPN,
TS059,                                TSO APPLICATIONS
```

TCP,
C1L08E0

TCP/IP APPLICATION
NON SNA LOCAL

To activate the TCP/IP VTAM definitions, issue the following command from the MVS console:

```
V NET,ACT,ID=TCP
```

Customizing TCPIP.DATA

A sample of TCPIP.DATA is provided in TCPIP.SEZAINST(TCPDATA). Copy the sample to SYS1.TCPPARMS(TCPDATA) and update it with your configuration; refer to Example B-3:

Example: B-3 Sample TCPIP.DATA

```
TCPIPJOBNAME TCPIP  
DATASETPREFIX TCPIP  
HOSTNAME TC1  
DOMAINORIGIN ITS0.IBM.COM  
NSINTERADDR 9.12.2.7
```

- ▶ TCPIPJOBNAME specifies the name of the started procedure that is used to start the TCPIP address space.
- ▶ DATASETPREFIX is used to set the High Level Qualifier for dynamic allocation of data sets in TCP/IP.
- ▶ The HOSTNAME statement specifies the TCP host name of this z/OS server. The fully qualified domain name for the host is formed by concatenating this host name with the domain origin (specified by the DOMAINORIGIN configuration statement).
- ▶ DOMAINORIGIN specifies the domain origin that is appended to the host name to form the fully qualified domain name for a host.
- ▶ NSINTERADDR defines the IP address of the name server in dotted decimal format.

Customizing other TCP/IP files

There are other files that influence the operation of TCP/IP. In this section we guide you through the setup of those files.

ETC.SERVICES

This file holds the information about how individual applications are assigned to the socket layer port numbers. Standard applications like TN3270 Telnet and FTP are assigned port numbers inside the well-known port number range (from 0 to 1023). The search order used to access this configuration file is as follows:

1. /etc/services that resides in the HFS.
2. userid.ETC.SERVICES, where userid is the user ID that is associated with the current security environment (address space or task/thread).
3. hlq.ETC.SERVICES, where hlq represents the value of the DATASETPREFIX keyword specified in the TCPIP.DATA configuration file (if found); otherwise, hlq is TCPIP by default.

Copy the sample member TCPIP.SEZAINST(SERVICES) into a new /etc/services HFS file. From ISPF option **6**, enter:

```
OPUT 'TCPIP.SEZAINST(SERVICES)' '/etc/services'
```

ETC.PROTO

This file contains a list of valid protocols supported by z/OS CS. The search order used to access this configuration file is as follows:

1. /etc/protocol that resides in the HFS.
2. userid.ETC.PROTO, where userid is the user ID that is associated with the current security environment (address space or task/thread).
3. hlq.ETC.PROTO, where hlq represents the value of the DATASETPREFIX keyword specified in the TCPIP.DATA configuration file (if found); otherwise, hlq is TCPIP by default.

Copy the sample member TCPIP.SEZAINST(PROTO) into a new /etc/protocol HFS file. From ISPF option **6**, enter:

```
OPUT 'TCPIP.SEZAINST(PROTO)' '/etc/protocol'
```

Customizing /etc/hosts file

The /etc/hosts file associates IP addresses with hostnames. Most systems have a small hosts file containing name and IP address information for important hosts on their own network. When applications do a hostname lookup, TCP/IP tries to resolve the hostname by looking in this file first, before going to the domain name server.

Create the /etc/hosts file by using the **ISHELL** or the **OEDIT** command. Add the IP addresses associated with hostnames to /etc/hosts; refer to Example B-4:

Example: B-4 /etc/hosts file

```
9.12.3.66    tc1.itso.ibm.com tc1
```

Setting up the resolver address space

The resolver acts on behalf of application programs as a client that accesses name servers for name-to-address or address-to-name resolution. If a name server is not available, the resolver will use local definitions (for example, `etc/hosts`, `HOSTS.SITEINFO` or `HOSTS.ADDRINFO`) to resolve the query for the requesting program. How (and if) the resolver uses name servers is controlled by `TCPIP.DATA` statements (resolver directives).

The resolver address space must be started before any application resolver calls occur. When the resolver address space starts, it reads an optional resolver setup data set pointed to by the `SETUP` DD card in the resolver JCL procedure. This resolver setup data set enables two key capabilities:

- ▶ Specification of a `TCPIP.DATA` file that contains global settings for the MVS image. The `GLOBALTCPIPDATA` setup statement identifies the file.
- ▶ Support for a user-specified default `TCPIP.DATA` file. The `DEFAULTTCPIP` setup statement identifies the file.

Starting the resolver address space

There are two ways to start the resolver address space:

- ▶ **z/OS UNIX initialization** will attempt to start the resolver unless explicitly instructed not to. Using **z/OS UNIX** is the recommended method, since it will ensure that the resolver is available before any applications can make a resolution request.

A `BPXPRMxx` statement, `RESOLVER_PROC`, is used to specify the procedure name, if any, to be used to start the resolver address space:

```
RESOLVER_PROC(procname|DEFAULT|NONE)
```

- ▶ An installation can use its automation tools to start the resolver by use of the **MVS START** operator command. If this approach to starting the resolver is used, care should be taken to ensure that no applications that need resolver services (for example, `INETD`) are started before the resolver address space is initialized.

Resolver customization

If an installation wants to make use of the new global `TCPIP.DATA` file, or to change the default `TCPIP.DATA` file name (`TCPIP.TCPIP.DATA`), the following steps will be required.

(If the new facilities are not required, no customization is required and the search order for `TCPIP.DATA` will be determined by the API being used.)

1. Create a resolver start procedure.

Copy the sample procedure member EZBREPRC(alias RESOPROC) in TCPIP.SEZAINST to your proclib, and modify the //SETUP DD JCL statement to point to a resolver setup file, as follows:

```
//RESOLVER PROC PARMS='CTRACE(CTIRES00) '  
//*  
//*   IBM Communications Server for OS/390  
//*  
//EZBREINI EXEC PGM=EZBREINI,REGION=OM,TIME=1440,PARM=&PARMS  
//*  
//*   SETUP contains Resolver setup parameters.  
//*   See the section on "Understanding Resolvers" in the  
//*   IP Configuration Guide for more information. A sample of  
//*   Resolver setup parameters is included in member RESSETUP  
//*   of the SEZAINST data set.  
//*  
//SETUP   DD   DSN=SYS1.TCPPARMS(RESSETUP),DISP=SHR,FREE=CLOSE  
//*SETUP   DD   DSN=TCPIP.SETUP.RESOLVER,DISP=SHR,FREE=CLOSE  
//*SETUP   DD   PATH='/etc/setup.resolver',PATHOPTS=(ORDONLY)
```

2. Create a resolver setup file (MVS data set or HFS file).

The setup file defines the location of the global TCPIP.DATA file (MVS data set or HFS file) and the default TCPIP.DATA name (MVS data set or HFS file).

The following statements are supported:

- DEFAULTTCPIPDATA
- GLOBALTCPIPDATA
- comments (; or #)

The z/OS CS-provided sample setup file can be found as member EZBRECNF(alias RESSETUP) in SEZAINST, as shown:

```
;
;   IBM Communications Server for OS/390
;   Function: Sample Resolver setup file
;
;   The following statement defines the final search location for
;   TCPIP.DATA statements. It will replace TCPIP.TCPIP.DATA
;   It may be an MVS data set or HFS file.
;
DEFAULTTCPIPDATA('SYS1.TCPPARMS(TCPDATA)')
;
#   The following statement defines the first search location for
#   TCPIP.DATA statements. It may be an MVS data set or HFS file.
;
;   Update with the correct data set or HFS file name
;
; GLOBALTCPIPDATA('TCPCS.SYS.TCPPARMS(GLOBAL)')
;
; GLOBALTCPIPDATA(/etc/tcpipglobal.data)
;
```

If the resolver setup file is an MVS data set, it must be: either sequential (PS) or partitioned (PO) organization; fixed (F) or fixed block format (FB); a logical record length (LRECL) between 80 and 256; and have any valid blocksize (BLKSIZE) for fixed block. If the setup file needs to be modified, a member of an MVS partitioned data set is recommended.

If the file is an HFS file, it can reside in any directory. The maximum length of line supported is 256 characters. If the line is greater than 256, it will be truncated to 256 and processed.

3. Define the RESOLVER started task:

```
ADDUSER RESOLVER DFLTGRP(OMVSGRP) OMVS(UID(42) HOME('/'))
RDEFINE STARTED RESOLVER.* STDATA(USER(RESOLVER))
SETROPTS GENERIC(STARTED) RACLIST(STARTED) REFRESH
```

The userid assigned to the resolver address space needs read access to the resolver setup file, the global TCPIP.DATA file and the default TCPIP.DATA file. Likewise, any userids or jobs using TCP/IP facilities will need read access to the global TCPIP.DATA file and default TCPIP.DATA file.

For the HFS file, permission bits of 644 (Owner can read and write, Group can read, Other can read) could be used. For an HFS file, an OMVS segment or the default OMVS segment must be configured for the resolver userid and any userids or jobs using TCP/IP facilities.

4. Update the z/OS UNIX BPXPRMxx parmlib member.

The resolver start procedure name should be specified as the *procname* in the BPXPRMxx Parmlib member's RESOLVER_PROC(*Procname*) statement.

If the recommended method of using z/OS UNIX to start the resolver is not desired for some reason, use the **MVS START** command to start the resolver address space; see the following example:

```
RESOLVER_PROC(RESOLVER)
```

Customizing the TCP/IP started task procedure

Copy the sample provided in TCPIP.SEZAINST(TCPIPPROC) to your PROCLIB and name it TCPIP. Update the following DD statements.

1. Update the PROFILE DD statement to point to your configuration file PROFILE.TCPIP, which is SYS1.TCPPARMS(PROFILE) in our example.
2. Update the SYSTCPD DD statement to point to your TCPIP.DATA configuration file, which is SYS1.TCPPARMS(TCPDATA) in our example.

Customizing the TCP/IP subsystem interface procedure

Virtual Machine Communication Facility (VMCF) and Terminal Notification Facility (TNF) are two MVS address spaces that manage interaddress space communication for TCP/IP. These address spaces contain the names of the VMCF and TNF users.

The EZAZSSI procedure is used to start VMCF/TNF as a started task. Without EZAZSSI, it would be necessary to IPL MVS in order to recover from certain VMCF/TNF problems.

Add the TCP/IP subsystem interface procedure EZAZSSI to your system PROCLIB. A sample of this procedure is located in the data set *hlq*.SEZAINST.

```
//EZAZSSI PROC P=&SYSNAME.  
//STARTVT EXEC PGM=EZAZSSI,PARM=&P,TIME=1440
```

Customizing SYS1.PARMLIB members

In this section we describe the required updates to PARMLIB members.

- Update IEFSSNxx with the TNF and VMCF subsystem statements required by TCP/IP:

```
SUBSYS SUBNAME(TNF)          /* TCP/IP */  
SUBSYS SUBNAME(VMCF)         /* TCP/IP */
```

- Update IECIOS00 to disable MIH requests for communication devices used by TCP/IP (in our example, DEV is 3002-3003):

```
MIH TIME=00:00,DEV=(3002-3003)
```

- Make sure that the following TCP/IP data sets are in PROGxx:

```
LNKLST Add Name(LNKLST00)
        Dsname(TCPIP.SEZALINK)
LNKLST Add Name(LNKLST00)
        Dsname(TCPIP.SEZATCP)
```

```
APF ADD
  DSNAM(TCPIP.SEZATCP)
  VOLUME(*****)
```

```
APF ADD
  DSNAM(TCPIP.SEZADSIL)
  VOLUME(*****)
```

```
APF ADD
  DSNAM(TCPIP.SEZALNK2)
  VOLUME(*****)
```

```
APF ADD
  DSNAM(TCPIP.SEZALINK)
  VOLUME(*****)
```

```
APF ADD
  DSNAM(TCPIP.SEZALPA)
  VOLUME(*****)
```

```
APF ADD
  DSNAM(TCPIP.SEZAMIG)
  VOLUME(*****)
```

- Make sure that the following TCP/IP data set is in LPALSTxx:

```
TCPIP.SEZALPA
```

- Update COMMNDxx to start the TCP/IP and TCP/IP subsystem interfaces:

```
COM='START EZAZSSI,P=SYSNAME,SUB=MSTR'
COM='START TCPIP'
```

Note: Start TCP/IP *after* VTAM, and start EZAZSSI *before* starting TCP/IP. Replace SYSNAME with the NJE nodename of your MVS system.

- SCHED00 defines the running and key modes for TCP/IP programs, as well as VMCF and TNF subsystems on OS/390 V2R6 and earlier. As of OS/390 V2R7, this no longer needed.

Customizing the BPXPRMxx PARMLIB member

BPXPRMxx defines the transport socket types for z/OS UNIX System Services applications. Depending on the number of stacks you want to run on your system, you can use the integrated sockets file system type (INET) or the Common INET file system type (CINET). The INET supports one TCP/IP stack at a time. It is used when applications communicate through a single stack.

Common INET is used when applications communicate through multiple stacks. It is possible that an installation would like to have more than one socket provider active at the same time; for example, an installation starts multiple instances of TCP/IP, with each stack having a unique TCP/IP identity in terms of network interfaces, IP addresses, host name and socket applications.

Recent enhancements to TCP/IP have reduced the need for multiple TCP/IP stacks. CINET may still be a viable choice if you are running Anynet and TCP/IP concurrently on a z/OS system, or isolating access from different networks to the same z/OS system (internal company networks versus Internet access is an example of this situation).

Use INET unless you have a special reason to use CINET.

Single TCP/IP stack definitions

For one TCP/IP stack, modify the definitions for AF_INET in SYS1.PARMLIB(BPXPRMxx) according to Example B-5:

Example: B-5 Definitions for single stack

```
FILESYSTYPE TYPE(UDS) ENTRYPOINT(BPXTUINT)
NETWORK DOMAINNAME(AF_UNIX)
                DOMAINNUMBER(1)
                MAXSOCKETS(10000)
                TYPE(UDS)

FILESYSTYPE TYPE(INET) ENTRYPOINT(EZBPFINI)
NETWORK DOMAINNAME(AF_INET)
                DOMAINNUMBER(2)
                MAXSOCKETS(60000)
                TYPE(INET)
```

- ▶ The FILESYSTYPE statement defines a physical file system to start, in the previous example:
 - UDS and INET are types of physical file systems.

- ▶ The NETWORK statement defines which domain the specified file system supports and some socket and port limits in that domain by specifying:
 - The address family type AF_UNIX is for socket communications that are between processes on the same system. AF_INET is for socket communication between processes on different systems or “over the wire”.
 - Its associated domain number.
 - The MAXSOCKETS parameter specifies the maximum number of sockets that can be active at any one time.

If you want to use the single transport driver support with both AF_INET and AF_INET6 address families, Example B-6 shows the BPXPRMxx parmlib member definitions:

Example: B-6 BPXPRMxx parmlib member definitions

```

FILESYSTYPE TYPE(UDS) ENTRYPOINT(BPXTUINT)
NETWORK DOMAINNAME(AF_UNIX)
          DOMAINNUMBER(1)
          MAXSOCKETS(10000)
          TYPE(UDS)

FILESYSTYPE TYPE(INET)
          ENTRYPOINT(EZBPFINI)
NETWORK DOMAINNAME(AF_INET)
          DOMAINNUMBER(2)
          MAXSOCKETS(2000)
          TYPE(INET)
NETWORK DOMAINNAME(AF_INET6)
          DOMAINNUMBER(19)
          MAXSOCKETS(2000)
          TYPE(INET)

```

Multiple TCP/IP stacks definitions

For multiple stacks, modify the definitions in BPXPRMxx according to Example B-7; you must use the Common INET (CINET).

Example: B-7 Definitions for multiple IP stacks

```

FILESYSTYPE TYPE(UDS) ENTRYPOINT(BPXTUINT)
NETWORK DOMAINNAME(AF_UNIX)
          DOMAINNUMBER(1)
          MAXSOCKETS(64)
          TYPE(UDS)

FILESYSTYPE TYPE(CINET)
          ENTRYPOINT(BPXTCINT)
NETWORK DOMAINNAME(AF_INET)
          DOMAINNUMBER(2)

```

```

MAXSOCKETS(10000)
TYPE(CINET)
INADDRANYPORT(10000)
INADDRANYCOUNT(2000)
SUBFILESYSTYPE NAME(TCPIP1) /* First TCP/IP */
TYPE(CINET)
ENTRYPOINT(EZBPFINI)
DEFAULT

SUBFILESYSTYPE NAME(TCPIP2) /* Second TCP/IP */
TYPE(CINET)
ENTRYPOINT(EZBPFINI)

SUBFILESYSTYPE NAME(TCPIP3) /* Third TCP/IP */
TYPE(CINET)
ENTRYPOINT(EZBPFINI)

```

Notes:

1. The names TCP/IP1, TCP/IP2, and TCP/IP3 are the names of the TCP/IP started tasks. The names must match the jobnames that are associated with the TCP/IP started task procedure.
2. The first TCP/IP has been designated as the default (DEFAULT) transport driver.
3. The value specified for the TYPE operand can be any 8-character value, but that value must match on the FILESYSTYPE statement for CINET, and on the SUBFILESYSTYPE statements for the transport drivers, and on the NETWORK statement for CINET.
4. Ensure that the INADDRANYPORT assignment does not conflict with PORT assignments in PROFILE.TCPIP data set.

Requesting transport affinity

A program can associate a socket with a specifically-named transport in different ways. In this section, we explain two of these ways:

- The `_BPXK_SETIBMOPT_TRANSPORT` environment variable can be set to the name of the desired transport before starting the program.

This variable can also be set in the `PARM=` parameter of an MVS started procedure to have the Language Environment run-time initialization issue a `setibmopt()` call on behalf of the program being started. This variable can also be included in the `_CEE_ENV` file.

- You can include a job step that invokes BPXTCAFF.

BPXTCAFF is invoked as a job step in front of an existing program in a started procedure or submitted job stream. For example:

```
//STEP0 EXEC,PGM=BPXTCAFF,PARM=TPNAME
//REALSTEP EXEC,PGM=MYPGM,PARM='MyParms'
```

The desired transport is specified with the PARM= keyword, and it must be 1 to 8 uppercase characters in length. This is the same value that would be specified for _BPXK_SETIBMOPT_TRANSPORT.

If PARM= is not supplied, or is blank, then the address space's transport affinity will be reset to: no transport selected. This can also be specified as PARM=&VAR, where VAR is a PROC keyword that is passed in from the Start command or is a static system symbol.

For more information, see “Setting up for Sockets” in *z/OS UNIX System Services Planning*, GA22-7800.

Security definitions for TCP/IP

To set up the security environment for TCP/IP, perform the following tasks:

TCP/IP and EZAZSSI address space security

1. Define a RACF user ID for the TCP/IP address space with UID=0:

```
ADDUSER TCPIP DFLTGRP(OMVSGRP) OMVS(UID(0) HOME('/') PROGRAM('/bin/sh')) +
NOPASSWORD
```

2. Define a RACF user ID for the EZAZSSI procedure:

```
ADDUSER EZAZSSI DFLTGRP(STCGRP) NOPASSWORD
```

3. Define a RACF STARTED class definition for TCP/IP started task:

```
RDEF STARTED TCPIP.* STDATA(USER(TCPIP) GROUP(OMVSGRP))
SETROPTS RACLIST(STARTED) REFRESH
```

4. Define a RACF STARTED class definition for EZAZSSI started task:

```
RDEF STARTED EZAZSSI.* STDATA(USER(EZAZSSI) GROUP(STCGRP))
SETROPTS RACLIST(STARTED) REFRESH
```

Secure the TCPIP VARY command

You can restrict access to the **VARY TCPIP** command by defining RACF profiles under the OPERCMDS class and specifying the list of users that are authorized to issue this command. CONTROL access is required for a user to be able to issue the **VARY TCPIP** command.

For example:

- ▶ **profile MVS.VARY.TCPIP.DROP** - for terminating a connection
- ▶ **MVS.VARY.TCPIP.OBEYFILE** - for dynamic configuration
- ▶ **MVS.VARY.TCPIP.PKTTRACE** - for setting up tracing
- ▶ **MVS.VARY.TCPIP.STRTSTOP** - to START or STOP a device

Examples

We control the use of the **OBEYFILE** and **DROP** commands as follows:

1. Activate the RACF OPERCMDS class:

```
SETROPTS CLASSACT(OPERCMDS) GENERIC(OPERCMDS)
SETROPTS GENCMD(OPERCMDS)
SETROPTS RACLIST(OPERCMDS)
```

2. Control the **VARY TCPIP OBEYFILE** and **DROP** commands:

```
RDEFINE OPERCMDS (MVS.VARY.TCPIP.DROP) UACC(NONE)
RDEFINE OPERCMDS (MVS.VARY.TCPIP.OBEYFILE) UACC(NONE)
PERMIT MVS.VARY.TCPIP.DROP ACCESS(CONTROL) CLASS(OPERCMDS) ID(USER1)
PERMIT MVS.VARY.TCPIP.OBEYFILE ACCESS(CONTROL) CLASS(OPERCMDS) ID(USER1)
SETROPTS GENERIC(OPERCMDS) REFRESH
SETROPTS RACLIST(OPERCMDS) REFRESH
```

To allow user ID AMR to issue all **MVS.VARY.TCPIP** commands:

```
RDEFINE OPERCMDS (MVS.VARY.TCPIP.***) UACC(NONE)
PERMIT MVS.VARY.TCPIP.** ACCESS(CONTROL) CLASS(OPERCMDS) ID(AMR)
```

Protect TCP/IP resources

The Communications Server uses the **SERVAUTH** RACF class to protect TCP/IP resources from unauthorized access. The use of **SERVAUTH** is optional.

In addition to the use of **SERVAUTH** protection, other functions provide further resource protection such as Intrusion Detection Services (IDS), syslogd isolation, and IP filtering. For more information, refer to *z/OS V1R2.0 CS: IP Configuration Guide*.

Stack Access Control

The Stack Access Control function allows control of access to a TCP/IP stack using RACF. It provides a way to generally allow or disallow users (or groups of user) access to a TCP/IP stack.

The function controls the ability of a user to open an **AF_INET** socket. The TCP/IP stack to be protected is represented with a RACF **SERVAUTH** profile name *EZB.STACKACCESS.sysname.tcpname*.

Access to the stack is allowed if the user is permitted to this RACF resource. There are no new TCP definitions required. The function is enabled if the RACF resource is defined. If it is not defined, a RACF check is not made.

To allow userid AMR to access TCP/IP stack TCP1 on system MVSA:

```
RDEFINE SERVAUTH (EZB.STACKACCESS.MVSA.TCP1) UACC(NONE)
PERMIT (EZB.STACKACCESS.MVSA.TCP1) ACCESS(READ) CLASS(SERVAUTH) ID(AMR)
```

Port Access Control

Port Access Control uses the PORT and PORTRANGE statements to protect against unauthorized use of non-ephemeral ports. It allows control of an application's ability to bind to specific TCP and UDP ports or port ranges using RACF.

The RACF support is used if a new keyword, SAF, is specified on the PORT or PORTRANGE statement. The SAF keyword value specifies a portion of a RACF resource name that represents the port.

The user ID associated with the application at the time of the bind to the port must be permitted to the RACF resource before the application can bind to the port. The port is represented to RACF with a RACF SERVAUTH profile name of *EZB.PORTACCESS.sysname.tcpname.SAFkeyword*. *SAFkeyword* is the value specified on the new SAF keyword on the PORT and PORTRANGE statement.

The following example provides an overview of Port Access Control. z/OS user WEBSERV is permitted to bind to port 80.

1. In the TCP/IP Profile definitions, on the port reservation statement, the SAF keyword ties an SAF resource to the reserved port number:

```
PORT 80 TCP * SAF WEBSRV
```

2. A SERVAUTH resource is created:

```
EZB.PORTACCESS.sysname.stackname.WEBSERV
```

When Universal access is set to NONE, and the started task user ID of the WEB server task is permitted READ access to the resource. Only this user ID can bind to the specified port number.

Network Access Control

The Network Access Control function enables system administrators to assign permission for z/OS users to access certain networks and hosts. With this function, the ability of users to send data from z/OS to certain networks can be controlled at the z/OS.

Network Access Control provides an additional layer of security to any authentication and authorization security that is used at the target system by disallowing the unauthorized user to send to the target network resource.

Example:

1. The TCP/IP Profile definitions:

```
NETACCESS
  9.67.40.0 255.255.248.0 ZONEB
  9.67.0.0 255.255.0.0 ZONEA
  Default WORLD
ENDNETACCESS
```

2. The SERVAUTH resources:

```
EZB.NETACCESS.sysname.stackname.ZONEA
EZB.NETACCESS.sysname.stackname.ZONEB
EZB.NETACCESS.sysname.stackname.WORLD
```

Netstat Access Control

Netstat Access Control allows control of access to Netstat command output from the TSO or UNIX System Services Shell environments using RACF.

The Netstat command output is considered the resource to be protected, and is represented with a RACF SERVAUTH profile name *EZB.NETSTAT.sysname.tcpname.netstatoption*.

Access to the Netstat output is allowed if the user is permitted to this RACF resource. There are no new TCP definitions required.

For example, if you want to permit AMR to have access to the Netstat CONN/-c option for TCP/IP stack TCP1 on system MVSA, you could use the following definitions:

```
RDEFINE SERVAUTH (EZB.NETSTAT.MVSA.TCP1.CONN) UACC(NONE)
PERMIT (EZB.NETSTAT.MVSA.TCP1.CONN) ACCESS(READ) CLASS(SERVAUTH) ID(AMR)
```

Note: A template of these commands and all other SAF commands appears in TCPIP.SEZAINST(EZARACF).

Starting TCP/IP

To start TCP/IP, issue the MVS command:

```
S TCPIP
```

Adding the new host to your DNS server

To enable the use of host names in a network, the Domain Name System (DNS) translates host names to IP addresses. DNS provides the host name-to-IP address mapping through network server hosts called Domain Name Servers.

DNS can also provide other information about server hosts and networks, such as the TCP/IP services available at a server host and the location of domain name servers in a network. Add the new host to your DNS server.

For more information about DNS, refer to *z/OS V1R2.0 CS: IP Configuration Guide*, SC31-8775.

You can also add static route definitions in the hosts file on your PC, as shown in Example B-8:

Example: B-8 Sample PC hosts file

9.12.3.66	tc1
-----------	-----

Testing your TCP/IP setup and connection

You can test your TCP/IP setup and connection as follows:

- ▶ The TCP/IP configuration program `tcpcfg` is a simple tool that provides information about the TCP/IP configuration on z/OS. It can diagnose some problems and make suggestions. This tool can be downloaded from the following Web site:

<http://www.ibm.com/servers/eserver/zseries/zos/unix/bpxalty2.html>

- ▶ To test your setup, go to ISPF option **6** and type: `HOMETEST`.
- ▶ From your workstation, issue: `ping <hostname>`. You should receive the following messages:

```
C:\>ping TC1
```

```
Pinging tc1.itso.ibm.com [9.12.3.66] with 32 bytes of data:
```

```
Reply from 9.12.3.66: bytes=32 time=10ms TTL=63
```

```
Reply from 9.12.3.66: bytes=32 time=10ms TTL=63
```

```
Reply from 9.12.3.66: bytes=32 time=10ms TTL=63
```

```
Reply from 9.12.3.66: bytes=32 time=10ms TTL=63
```

Customizing IBM-supplied daemons

z/OS UNIX and TCP/IP network services are supported by a number of daemons. As mentioned, daemons are programs that run in the background and either provide services directly or maintain logs that are used by other daemon programs.

The Internet services daemon (inetd) provides service management for a network. This daemon reduces system load by invoking other daemons only when they are needed, and by providing several simple Internet services internally without invoking other daemons.

Once started, the inetd daemon listens for connections on certain network sockets in /etc/inetd.conf. The /etc/inetd.conf file tells the inetd daemon how to handle service requests on network sockets. When inetd receives a request on one of these sockets, it determines which service corresponds to that socket. Then it either handles the service request itself or invokes the appropriate server.

The following servers are started and managed by inetd:

- **telnetd**

The Telnet protocol provides a standardized interface that allows terminal devices and terminal-oriented processes on hosts that support TCP/IP to communicate with each other. z/OS supports a separate Telnet daemon for TN3270 access (TSO access), and a separate Telnet daemon for z/OS UNIX access. The Telnet daemon listens on well-known port 23.

Note: TN3270 (Telnet 3270) runs in the TCPIP address space. It is a part of TCP/IP. The z/OS UNIX Telnet server is started and managed by inetd.

In our examples, we defined port 623 for the TN3270 Telnet daemon and port 23 for the z/OS UNIX Telnet daemon.

- **rshd (remote shell daemon)**

A remote shell client passes a command to a remote host for execution. This command is interpreted and run by the remote shell daemon. Standard output and standard errors from the remote execution are returned to the local host. The remote shell daemon uses the TCP protocol and listens on well-known port 514.

- **rlogind (remote login daemon)**

This daemon provides interactive access to remote hosts. Its function is similar to that of Telnet. It is most widely used between UNIX hosts, so there are very few rlogin clients found for the PC environment. The rlogin daemon uses the TCP protocol and listens on well-known port 513.

- ▶ **rexecd (remote exec daemon)**

Remote exec works similar to remote shell. A remote exec client passes a command to be run on a remote host. This command is interpreted and run by the remote exec daemon. The remote exec daemon uses the TCP protocol and listens on well-known port 512.

The following z/OS UNIX daemons are started and controlled separately from inetd:

- ▶ **syslogd (syslog daemon)**

The syslog daemon is a message logging server. It listens for messages. When it receives them, it forwards them to other processes, files, and devices using the specification in the /etc/syslog.conf file. The syslog daemon uses the UDP protocol and listens on well-known port 514.

- ▶ **FTPd (File Transfer Protocol daemon)**

FTPd allows you to transfer files between hosts. On z/OS, the same FTP daemon can transfer both MVS data sets, as well as HFS files. FTPd uses the TCP protocol and listens on well-known port 21.

Figure B-1 on page 350 shows an overview of logging in to z/OS UNIX.

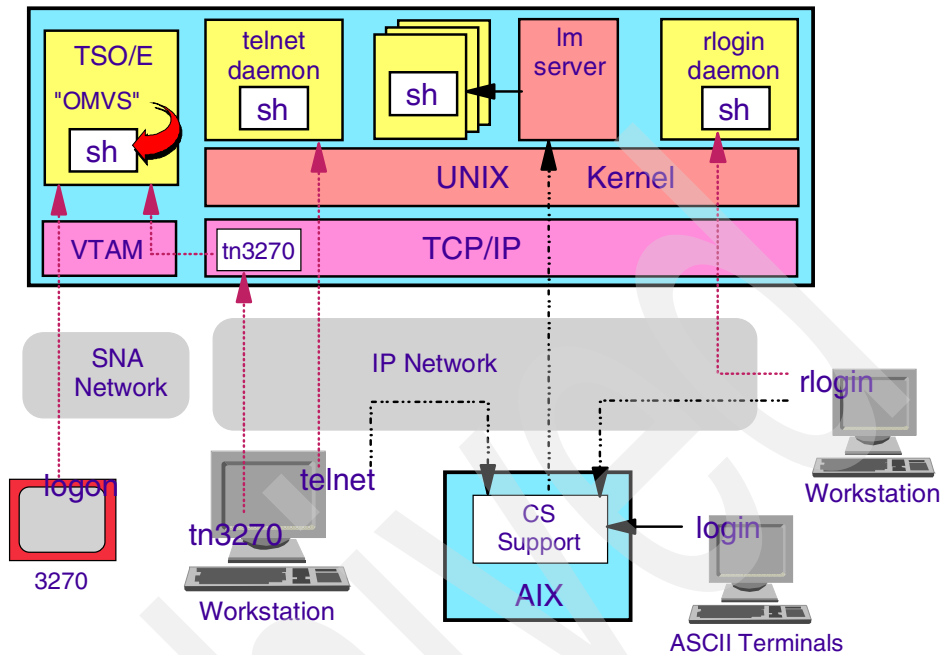


Figure B-1 Logging in to the z/OS UNIX shell

Customizing inetd

The function of the inet daemon is to listen on certain “well-known” network ports for a request to run one of a number of daemons. When a request is received, inetd creates a new socket for remote connection, then forks a new address space and uses `exec()` to start the requested daemon program. Figure B-2 on page 351 illustrates the customization of inetd.

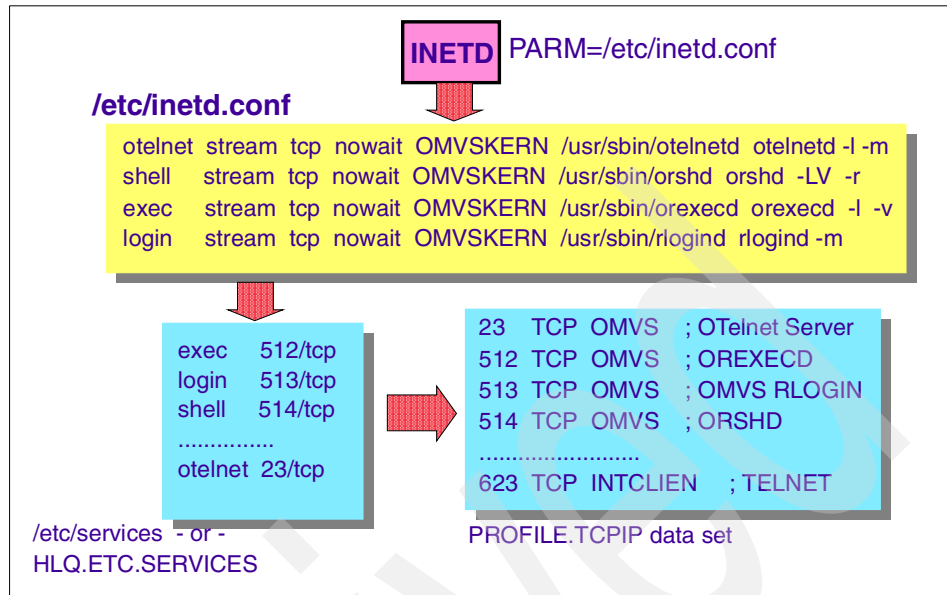


Figure B-2 Customization of inetd

To set up inetd, you must do the following:

- ▶ Set up inetd files
- ▶ Customize inetd startup
- ▶ Create security definitions

Set up inetd files

As shown in Figure B-2, there are three configuration files that need to be updated for the inetd support: /etc/inetd.conf, /etc/services, and PROFILE.TCPIP.

- ▶ /etc/inetd.conf file

The inetd daemon uses the configuration file /etc/inetd.conf to determine what ports to listen on and what processes to start for requests on those ports. There is one line (entry) for each daemon controlled by inetd.

Copy the /samples/inetd.conf to /etc/inetd.conf. An sample of this file is shown in Example B-9:

Example: B-9 Sample of the /etc/inetd.conf file

```
# /etc/inetd.conf
#
#       Internet server configuration database
#
# Services can be added and deleted by deleting or inserting a
# comment character (ie. #) at the beginning of a line
```

```
#
#=====
# service | socket | protocol | wait/ | user | server | server program
# name    | type  |         | nowait|      | program | arguments
#=====
#
otelnet  stream tcp nowait OMVSKERN /usr/sbin/otelnetd otelnetd -l
shell    stream tcp nowait OMVSKERN /usr/sbin/orshd orshd -LV
login    stream tcp nowait OMVSKERN /usr/sbin/rlogind rlogind -m
exec     stream tcp nowait OMVSKERN /usr/sbin/orexecd orexecd -LV
```

These statements tell the inetd daemon that when a request is received for Telnet, remote shell, remote login or remote exec, the system starts a named process to service that request. OMVSKERN is the initial user ID that this named process runs. The daemon process executable resides in /usr/sbin.

► /etc/services file

The /etc/services file has information about how individual applications are assigned to socket layer port numbers. Servers started by inetd use the service name that you specify in /etc/inetd.conf to match a server port number in /etc/services. Change /etc/services as shown in Example B-10:

Example: B-10 Example of the /etc/services file

Otelnet	23/tcp	
telnet	623/tcp	
exec	512/tcp	
biff	512/udp	comsat
login	513/tcp	
who	513/udp	whod
shell	514/tcp	cmd
syslog	514/udp	

► PROFILE.TCPIP

In the PROFILE.TCPIP, add the following ports in the PORT statement. We use the OMVS keyword to indicate that these ports are reserved for z/OS UNIX.

If we explicitly define the started task or the job name, it can create a problem, as we are not sure what the new forked name will be. For example, when you start FTPd, you will see FTPD start and end; another address space FTPD1 (it can be FTPDx, where x can be any number) will start up in its place.

```
PORT
    23 TCP OMVS                ; OMVS Telnet Server
    623 TCP INTCLIEN           ; TN3270 Server
```

Note: To activate the changes that you made to PROFILE.TCPIP, you can either **STOP** and then **START** the TCP/IP procedure, or issue the following command to activate the changes dynamically:

```
VARY TCPIP,,0,DSN=SYS1.TCPPARMS(PROFILE)
```

Customizing inetd startup

Add the INETD started task procedure to your proclib, as shown in Example B-11:

Example: B-11 Example of a started task starting inetd

```
//INETD  PROC
//INETD  EXEC  PGM=BPXBATCH,REGION=30M,TIME=NOLIMIT,
//          PARM='PGM /usr/sbin/inetd /etc/inetd.conf'
//STDOUT  DD  PATH='/tmp/inetd-stdout',
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//          PATHMODE=SIRWXU
//STDERR  DD  PATH='/tmp/inetd-stderr',
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//          PATHMODE=SIRWXU
// *STDENV DD  DSN=TCPIPOE.&SYSNAME..TCPPARMS(INETDENV),DISP=SHR
```

The inetd daemon can also be started automatically at OMVS startup by specifying it in the /etc/rc file.

Create security definitions for inetd

1. Define the inetd user ID as UID = 0.

In our example, inetd will run with the kernel (OMVSKERN) user ID, which has already been defined as UID(0). OMVSKERN is the user ID that is defined in the /etc/inetd.conf file to start the inetd-controlled daemons.

2. Authorize inetd to BPX.DAEMON.
3. Make sure all the required modules are in controlled libraries:

```
RDEFINE PROGRAM * ADDMEM('CEE.SCEERUN'//NOPADCHK +
                          'SYS1.LINKLIB'//NOPADCHK +
                          'TCPIP.SEZALINK'//NOPADCHK +
                          'TCPIP.SEZATCP'//NOPADCHK) UACC(READ)
```

Refresh the profile in memory:

```
SETROPTS WHEN(PROGRAM) REFRESH
```

4. Define a RACF STARTED class definition for INETD, as follows:

```
RDEF STARTED INETD.* STDATA(USER(OMVSKERN) GROUP(OMVSGRP))
SETROPTS RACLIST(STARTED) REFRESH
```

Starting INETD

From the MVS console, issue:

```
S INETD
```

You'll see that the parent INETD process (task) ends, but if you issue: D J,L from the MVS console, you'll notice another process called INETD1, which is the forked process.

Customizing the syslog daemon

The syslog daemon is used to log system messages to the MVS console, log files, other machines, or users. To customize the syslog daemon, do the following:

1. Create and update the /etc/syslog.conf configuration file.

A sample is found in /usr/lpp/tcpip/samples/syslog.conf; copy the sample to /etc/syslog.conf.

Example B-12 shows an example of the /etc/syslog.conf. The /dev/console statement was uncommented to write all messages to MVS syslog.

Example: B-12 Example of the /etc/syslog.conf file

```
#
# The following has been uncommented and can be used in an
# actual installation. The files named must exist before
# the syslog daemon is started.
# all messages will be written to the MVS syslog
*.* /dev/console
```

2. Update the PORT statement in the PROFILE.TCPIP; syslogd uses port 514:

```
514 TCP OMVS ; Remote Shell Server
514 UDP OMVS ; SyslogD Server
```

3. Update the /etc/rc file to start the syslog daemon.

The syslog daemon should be started as one of the first processes in your z/OS UNIX System Services environment. The shell script **sleep** command can be used to set up a delay so the syslog daemon has enough time to start before the other servers, as shown in the following:

```
# Start the SYSLOG daemon
export _BPX_JOBNAME='syslogd'
/usr/sbin/syslogd -f /etc/syslog.conf &

sleep 5
```

Customizing FTP

The File Transfer Protocol (FTP) allows a user to copy files from one machine to another. The protocol allows for data transfer between the client (the end user) and the server in either direction. This data can be MVS data sets, as well as HFS files.

In addition to copying files, the client can issue FTP commands to the server to manipulate the underlying file system of the server (for example, to create or delete directories, delete files, rename existing files, and so on.) FTP is the most frequently used TCP/IP application for moving files between computers.

Configuring PROFILE.TCPIP

- Update the PORT statement.

FTP uses ports 20 and 21. Port 21 is the control port, and port 20 is the data port.

```
PORT
    20 TCP OMVS      NOAUTOLOG ; FTP Server
    21 TCP OMVS      ; FTP Server
```

- Add the FTPD started task name to AUTOLOG, so that the FTP daemon will start automatically when TCP/IP starts:

```
AUTOLOG 5
    FTPD ; FTP Server
ENDAUTOLOG
```

Configuring ETC.SERVICES

In /etc/services, verify that the following entry exists:

```
ftp      21/tcp
```

Configuring /etc/syslog.conf

For ftpd syslog, you should consider the fact that ftpd writes log messages to the system console if syslogd is not running. If you enable ftp server traces without syslogd active, large amounts of data might be written to the system console.

The daemon.priority entries in /etc/syslog.conf determine where FTP messages and trace entries are written. The FTP server issues informational, warning, and error messages. All trace entries are written with debug priority. To direct trace entries (and all messages) to /tmp/daemon.trace, include the following in /etc/syslog.conf:

```
*.*.daemon.debug    /tmp/daemon.trace
```

Log messages can be isolated within syslogd. For FTP, an installation might want FTP log messages to be written to different files depending on the user ID, or separately for the FTP daemon.

If FTP messages are to be isolated for user1, use this statement:

```
user1.*.daemon.debug /tmp/daemon.trace
```

If FTP messages are to be logged for all FTP applications, use this statement:

```
*.FTPD*.daemon.debug /tmp/daemon.trace
```

In these statements, it is assumed that `_BPX_JOBNAME` is set to `FTPD`.

Creating the FTP server started task procedure

Copy the sample in `TCPIP.SEZAINST(FTPD)` to your `PROCLIB`. Update the `SYSTCPD` DD card to point to your `TCPIP.DATA`.

If you are running multiple TCP/IP stacks and you want to ensure that FTP has an affinity to a TCP/IP stack, use the `_BPXK_SETIBMOPT_TRANSPORT` keyword.

This example sets the FTP server to have an affinity to `TCPIEOE`:

```
//FTPD  PROC  MODULE='FTPD',PARMS=' '  
//FTPD  EXEC  PGM=&MODULE,REGION=4096K,TIME=NOLIMIT,  
//      PARM=('POSIX(ON) ALL31(ON)',  
//          'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIEOE"',  
//          '"TZ=EST")/&PARMS')
```

Configuring FTP.DATA

The `FTP.DATA` data set is optional. It contains FTP daemon startup parameters. The FTP daemon looks for this data set during initialization, following this sequence:

1. A data set specified by the `//SYSFTPD` DD statement
2. `ftpserve_job_name.FTP.DATA`
3. `/etc/ftp.data`
4. `SYS1.TCPPARMS(FTPDATA)`
5. `hlq.FTP.DATA` data set

In our example, we used `/etc/ftp.data`. There is a sample in `TCPIP.SEZAINST(FTPDATA)`. Use `TSO OPUT` to copy it to `/etc/ftp.data`, as follows:

```
OPUT 'TCPIP.SEZAINST(FTPDATA)' '/etc/ftp.data'
```

Example B-13 on page 357 shows an example of a modified `/etc/ftp.data` file.

Example: B-13 Example of the /etc/ftp.data file

```
StartDir      HFS
umask         022
;
; File and disk parameters
;
Primary       5           ; Primary allocation is 5 tracks
Secondary     2           ; Secondary allocation is 2 tracks
Directory     15          ; PDS allocated with 15 directory blocks
Lrecl         128         ; Logical record length is 128 bytes
BlockSize     6144        ; Block size is 6144 bytes
AutoRecall    true        ; Migrated HSM files recalled automatically
AutoMount     true        ; Nonmounted volumes mounted automatically
DirectoryMode  false       ; Use all qualifiers (Datasetmode)
Volume        TC1SY1      ; Volume serial number for allocation
SpaceType     TRACK       ; Data sets allocated in tracks
Recfm         FB          ; Fixed blocked record format
UnitName      3390        ; Unit name used for allocation
Filetype      SEQ         ; File Type = SEQ (default)
RDW           false       ; Do not retain RDWs as data
```

Security definitions for FTP daemon

To create the RACF definitions for the FTP daemon, complete the following tasks.

1. Define user ID for FTP daemon with UID(0).

In our example, FTP will run with the TCPIP user ID.

2. Grant DAEMON authority to the FTP user ID, as follows:

```
PERMIT BPX.DAEMON CLASS(FACILITY) ID(TCPIP) ACCESS(READ)
SETROPTS GLOBAL(FACILITY) RACLIST(FACILITY) GENERIC(FACILITY) REFRESH
```

3. Define the FTPD started task.

```
RDEFINE STARTED FTPD.* STDATA(USER(TCPIP) GROUP(OMVSGRP))
SETROPTS GENERIC(STARTED) RACLIST(STARTED) REFRESH
```

Security considerations for the FTP server

Consider the following for security:

- User IDs

To log into the FTP server, a user ID must either have an z/OS UNIX UID, or it can use the default UNIX UID.

- MVS Network Access Controls

In the RACF SERVAUTH class, PORTACCESS or NETACCESS resources could be defined to secure TCP ports or networks. For more information, see “Protect TCP/IP resources” on page 344. Also see the NETACCESS statement in *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776.

Starting FTPD

To start FTPD from the MVS console, issue:

```
S FTPD
```

You will see FTPD start and end, but another address space, FTPD1, will start up in its place. For more information about FTP, refer to *z/OS V1R2.0 CS: IP Configuration Guide*, SC31-8775.

DB2 quick setup

This appendix describes how to perform a quick setup of a DB2 subsystem within a data sharing group for test and development purposes. While it does not claim completeness or offer detailed explanations, it did prove to be helpful when we needed additional DB2 systems.

WebSphere Application Server uses DB2 7.1 to store configuration information. In this appendix we give a brief description of DB2 and how to install and configure it to work with WebSphere. Further information can be found in *DB2 Universal Database for OS/390 and z/OS Version 7 Installation Guide*, GC26-9936, and *DB2 Universal Database for OS/390 and z/OS: Program Directory Version 7*, GI10-8216.

Overview

DB2 has a comprehensive infrastructure that enables it to provide data integrity, performance, and the ability to recover user data. DB2 controls and accesses system structures. These system structures include the Catalog, Active and Archive logs, as well as the Bootstrap data set and Buffer pools.

- ▶ The DB2 catalog

This set of tables contains information about the data that is under DB2 control.

- ▶ Archive logs

DB2 records all data changes and other significant events in archive logs.

- ▶ Bootstrap data set (BSDS)

This data set contains information that is critical to DB2, such as the names of the logs.

- ▶ Buffer pools

Also known as *virtual buffer pools*, these are areas of virtual storage in which DB2 temporarily stores pages of table spaces or indexes.

WebSphere Application Server use of DB2

WebSphere uses shared DB2 tables to hold its configuration. These must be accessible from every system in the sysplex. Applications can run on any system in the sysplex—and their data must also be accessible.

Note: In this appendix we describe how to install and configure DB2 to be used by WebSphere in a sysplex. Notice that the configuration is very simple; more complex DB2 environments, such as those required by production applications, are not investigated. We do not discuss the use of DB2 by applications; this subject is covered in Chapter 4, “Java-based access to DB2 for z/OS” on page 117.

DB2 pre-installation considerations

DB2 *sysadm* authority is required. You can run the DB2 configuration jobs yourself, or you can get a DB2 administrator to run them. The user ID must be able to issue UNIX Services commands to install the JDBC HFS data set.

DB2 installation in a sysplex

A *data sharing group* is a collection of one or more DB2 subsystems that access shared DB2 data. The data sharing function of the licensed program DB2 for OS/390 and z/OS enables applications that run on more than one DB2 subsystem to read from and write to the same set of data concurrently.

DB2 subsystems that share data must belong to a DB2 data sharing group, which runs on a Parallel Sysplex. A Parallel Sysplex is a collection of MVS systems that communicate and exchange data with each other.

Each DB2 subsystem that belongs to a particular data sharing group is a member of that group. All members of a data sharing group use the same shared DB2 catalog and directory. Currently, the maximum number of members in a data sharing group is 32. For more information about running DB2 in data sharing mode, see: *Data Sharing: Planning and Administration Version 7*, SC26-9935.

The following assumes you already have a Parallel Sysplex configured and DB2 V7 is SMP/E installed.

The DB2 product supplies a set of installation ISPF dialogs that are used to customize a DB2 subsystem. The dialogs will prompt you to supply parameters for your DB2 environment, or allow you to accept defaults.

Run the dialogs by issuing DSNTINST from ISPF option 6

To use the ISPF panels, you must first make the DB2 ISPF libraries available and invoke the installation CLIST in ISPF mode. A sample CLIST to allocate the necessary data sets is shown here:

```
PROC 0 DEBUG          CONTROL(NONE)
/*
  ALTLIB ACTIVATE APPLICATION(CLIST)  +
  DATASET('DSN710.SDSNCLST')
  ISPEXEC LIBDEF ISPLLIB DATASET +
  ID('DSN710.SDSNSPF' 'DSN710.SDSNPFPE')
  ISPEXEC LIBDEF ISPLLIB DATASET ID('DSN710.SDSNSPFS')
  ISPEXEC LIBDEF ISPTLIB DATASET ID('DSN710.SDSNSPFT')
  ISPEXEC LIBDEF ISPMLIB DATASET ID('DSN710.SDSNSPFM')
  ISPEXEC SELECT CMD(DSNTINST)
  ISPEXEC LIBDEF ISPLLIB
  ISPEXEC LIBDEF ISPTLIB
  ISPEXEC LIBDEF ISPMLIB
  ISPEXEC LIBDEF ISPLLIB
  ALTLIB DEACTIVATE APPLICATION(CLIST)
```

Press Enter to advance from one panel to the next. At the end, a number of JCL members will be generated in DB7FU.D7F1.NEW.SDSNSAMP and DB7FU.D7F1.NEW.SDSNTEMP. Fill in the panels with values appropriate to your installation.

For the purposes of this example, we are building the first member of a data sharing group with the following values:

- High level qualifier for DB2 target libraries: DSN710.**
- High level qualifier for DB2 user databases: DB7FU.**
- Subsystem name: D7F1
- Trigger character: "-"

Data parameters

When building the first DB2 data sharing system for the residency, we used the following unique data parameters:

DATA SHARING => YES

If you're setting up a data sharing group, when you run the CLIST for the initial build, specify YES for data sharing, and on the next screen, select **Group**. For subsequent members of the group, you select **Member** rather than group.

DATA SHARING FUNCTION => GROUP

INPUT MEMBER NAME => DSNTIDXA

OUTPUT MEMBER NAME => D7F1

When filling in the dialog panels, all entries are saved in this output member in DSN710.SDSNSAMP.

GROUP NAME => DB7FU

MEMBER NAME => D7F1

WORK FILE DB => D7F1

GROUP ATTACH NAME => D7FG

TEMP CLIST LIBRARY => DB7FU.D7F1.NEW.SDSNTEMP

SAMPLE LIBRARY => DB7FU.D7F1.NEW.SDSNSAMP

DB2 SUBSYSTEM NAME => D7F1

MAX USERS => 700

MAX BATCH CONNECT => 500

BP0 => 2000

BP1 => 5000

BP2 => 8000

BP7 => 3000

BP32K => 100

BP32K1 => 500

PARAMETER MODULE => DSNZPAF1

DB2 data sharing systems will share the SDSNEXIT data set, but each data sharing member must have a unique ZPARM member. When updating the installation dialogs, enter a unique ZPARM name for each data sharing member.

CACHE DYNAMIC SQL => YES

IRLM SUBSYSTEM NAME => I7F1

IRLM PROC NAME => D7F1IRLM

When running in data sharing mode, you will have a unique IRLM procedure running on each system in the data sharing group in the sysplex.

IRLM MEMBER IDENTIFIER => 1

Make sure you specify a unique IRLMID number for each IRLM procedure. Usually member 1 will have an IRLMID of 1, member 2 will have an IRLMID of 2, and so on.

IRLM XCF GROUP NAME => DB7FGRP

Each IRLM procedure should point to the same IRLM XCF GRP value.

SYSTEM ADMIN 1 => your TSO userid

SUBSYSTEM NAME => D7F1

COMMAND PREFIX => -D7F1

Dialog parameters

The following dialog parameters build the DRDA component of DB2. While required to be configured for WebSphere, it is also helpful during application development to be able to connect to the DB2 database from a workstation.

DB2 STARTUP OPTION => AUTO

DB2 LOCATION NAME => DB7F

This is the name used by other DB2 systems to refer to all the DB2 subsystems in this data sharing group. Each data sharing member should specify the same DB2 location name.

Do not make the DB2 location name the same as the DB2 subsystem name. New DB2 subsystems in this data sharing group will use this value.

DB2 NETWORK LUNAME => SCPD7F1

DRDA PORT => 33730

RESYNC PORT => 33731

The dialogs will generate customized JCL that you can use to build the first data sharing DB2 member.

Building and activating structures in the Coupling Facility

Remember to build and activate structures in the Coupling Facility for the DB2 you are building. Here are some example definitions:

```
/*-----*/  
/* DB2 DATA SHARING GROUP: DB7FU    / LOCK STRUCTURE */  
/*-----*/
```

```
        STRUCTURE NAME(DB7FU_LOCK1)  
        INITSIZE(32768)  
        SIZE(65536)  
        PREFLIST(CF01,CF02)  
        REBUILDPERCENT(5)
```

```
/*-----*/  
/* DB2 DATA SHARING GROUP: DB7FU    / LIST STRUCTURE */  
/*-----*/
```

```
        STRUCTURE NAME(DB7FU_SCA)  
        INITSIZE(16384)  
        SIZE(32768)  
        PREFLIST(CF01,CF02)  
        REBUILDPERCENT(5)
```

```
/*-----*/  
/* DB2 DATA SHARING GROUP: DB7FU   / CACHE STRUCTURES */  
/*-----*/
```

```
STRUCTURE NAME(DB7FU_GBP0)  
  INITSIZE(16000)  
  SIZE(32000)  
  DUPLEX(ENABLED)  
  PREFLIST(CF02,CF01)  
  REBUILDPERCENT(5)
```

```
STRUCTURE NAME(DB7FU_GBP1)  
  INITSIZE(8192)  
  SIZE(16384)  
  PREFLIST(CF02,CF01)  
  REBUILDPERCENT(5)
```

```
STRUCTURE NAME(DB7FU_GBP2)  
  INITSIZE(48000)  
  SIZE(64000)  
  DUPLEX(ENABLED)  
  PREFLIST(CF02,CF01)  
  REBUILDPERCENT(5)
```

```
STRUCTURE NAME(DB7FU_GBP3)  
  INITSIZE(48000)  
  SIZE(64000)  
  DUPLEX(ENABLED)  
  PREFLIST(CF02,CF01)  
  REBUILDPERCENT(5)
```

```
STRUCTURE NAME(DB7FU_GBP8K0)  
  INITSIZE(16000)  
  SIZE(32000)  
  DUPLEX(ENABLED)  
  PREFLIST(CF02,CF01)  
  REBUILDPERCENT(5)
```

```
STRUCTURE NAME(DB7FU_GBP32K)  
  INITSIZE(16000)  
  SIZE(32000)  
  DUPLEX(ENABLED)  
  PREFLIST(CF02,CF01)  
  REBUILDPERCENT(5)
```

Editing and running the customized JCL

Edit and run the customized JCL to perform the following tasks. You'll need to carefully review the JCL, as it may need to be modified to match your installation standards. It is beyond the scope of this redbook to cover this consideration in detail.

Define DB2 to MVS and build catalog procedures

This job (DB7FU.D7F1.NEW.SDSNSAMP(DSNTIJMV)) will update the LINKLIST. We recommend that you make these changes manually and delete the relevant steps from the job.

Note: One step in this job will add the following data sets to APF and/or the LINKLIST. Delete this step from the JCL if you choose to do this manually.

- ▶ DSN710.SDSNLINK - This contains modules that you must place in the LINKLIST because they are loaded at subsystem initialization during IPL. After adding SDSNLINK to the LINKLIST, an IPL is required to update the MVS Subsystem Vector Table (SSVT). This data set must be APF-authorized.
- ▶ DSN710.SDSNLOAD - This contains all the main DB2 modules. You can either add this library to linklist, or use a STEPLIB or JOBLIB to access. If you are running many DB2 subsystems on the same system concurrently, it is best to use JOBLIBs or STEPLIBs to access SDSNLOAD. This data set must be APF-authorized.
- ▶ DSN710.SDSNEXIT - This data set must be APF-authorized.
- ▶ DSN710.SDXRRESL - This data set must be APF-authorized.
- ▶ DB7FU.RUNLIB.LOAD - This data set must be APF-authorized.

1. After updating PARMLIB with APF and/or linklist changes, you must activate the changes. You can do this dynamically by issuing the SETPROG APF and SETPROG LNKLIST MVS commands.

2. Update IEFSSNxx for IRLM and DB2 - we recommend you update this member manually. Delete the DSNTIMP step from the JCL. Parmlib updates:

```
SUBSYS SUBNAME(I7F1)
SUBSYS SUBNAME(D7F1) INITRTN(DSN3INI) INITPARM('DSN3EPX,-D7F1,S,D7FG')
```

3. In addition, you can issue the following MVS commands to add the DB2 subsystem definitions dynamically:

```
SETSSI ADD,SUBNAME=I7F1
SETSSI ADD,SUBNAME=D7F1,INITRTN=DSN3INI,INITPARM='DSN3EPX,-D7F1,S,D7FG'
```

4. Run the DSNTIPM step to add the procedures to PROCLIB; expect RC=0.

5. If you did not activate the APF and/or LINKLIST data sets dynamically, or if you did not add the subsystem names dynamically with the SETSSI command, you must IPL the system now to pick up the changes.

Define DB2 bootstrap and log data sets

Submit the JCL DB7FU.D7F1.NEW.SDSNSAMP(DSNTIJIN); expect RC=0.

Initialize DB2 catalog and directory data sets

Submit the JCL DB7FU.D7F1.NEW.SDSNSAMP(DSNTIJID); expect RC=0.

Define DB2 initialization parameters

Job: DB7FU.D7F1.NEW.SDSNSAMP(DSNTIJUZ).

Submit the JCL; expect RC=0.

Establish the DB2 - TSO environment

Job: DB7FU.D7F1.NEW.SDSNSAMP(DSNTIJVC).

- Find the first SYSIN, and change it to DD DUMMY.
- Find DSNTIVB.SYSUT2, and change it to RECFM=FB, LRECL=80, BLKSIZE=0.
- Submit the JCL; expect RC=0.

Update VTAM

Add a VTAM applid for each DB2 subsystem and vary it active. The following example is for D7F1:

```
VBUILD TYPE=APPL
*
SCPD7F1  APPL ACBNAME=SCPD7F1,
          APPC=YES,
          ATNLOSS=ALL,
          AUTH=(ACQ),
          AUTOS=10,
          DMINWNL=25,
          DMINWNR=25,
          DSESLIM=50,
          EAS=509,
          ENCR=NONE,
          MODETAB=AGWTAB,
          PARSESS=YES,
          SECACPT=ALREADYV,
          SONSCIP=NO,
          SRBEXIT=YES,
          SYNCLVL=SYNCPT,
```

X
X
X
X
X
X
X
X
X
X
X
X
X
X
X
X
X

VERIFY=NONE,	X
VPACING=2,	X
VTAMFRR=NO	

Add a RACF user id and RACF Started Class Profile

In the following example, SYS1 is a valid RACF group and has a valid GID assigned.

```
ADDUSER DB2STC DFLTGRP(SYS1) NOPASSWORD OWNER(SYS1)
ALTUSER DB2STC OMVS(UID(0) HOME('/') PROGRAM('/bin/sh'))
SETROPTS GENERIC(STARTED)
RDEFINE STARTED D7F1*.* OWNER(SYS1) STDATA(USER(DB2STC) +
GROUP(SYS1) TRUSTED(NO) TRACE(YES))
SETROPTS CLASSACT(STARTED) RACLIST(STARTED)
SETROPTS RACLIST(STARTED) REFRESH
```

Start DB2 Version 7

You are now ready to start your DB2 system. Go into SDSF and start your DB2 subsystem: /-D7F1 STA DB2. The following procedures should start D7F1MSTR, D7F1IRLM, D7F1DBM1, D7F1DIST and D7F1SPAS.

Create temporary files for DB2

Job: DB7FU.D7F1.NEW.SDSNSAMP(DSNTIJTM). Submit the JCL; expect RC=0 for all steps (however, for step DSNTWR, PROCSTEP DSNTIAS you should expect RC=12).

Create user-maintained tables

Job: DB7FU.D7F1.NEW.SDSNSAMP(DSNTIJSG). Submit the JCL; expect RC=0.

Bind CLI default packages and plan

Job: DB7FU.D7F1.NEW.SDSNSAMP(DSNTIJCL). This job is not in your data set. It must be copied from DSN710.SDSNSAMP(DSNTIJCL). You have to do a certain amount of tailoring here:

- ▶ Check the JOB card and JOBPARM.
- ▶ Change every DSN SYSTEM(xxxx) to DSN SYSTEM(D7F1).
- ▶ Change every DSN!!0.SDSNLOAD to DSN710.SDSNLOAD.
- ▶ Change every DSN!!0.SDSNDBRM to DSN710.SDSNDBRM.
- ▶ Add the following step to the end of the job to grant PUBLIC access to this plan:

```
//GRANT EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT DD SYSOUT=*
```

```
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSIN DD *
  DSN SYSTEM(D7F1)
  RUN PROGRAM(DSNTIAD) PLAN(DSNTIA71) -
  LIBRARY('DB7FU.RUNLIB.LOAD')
  END
//SYSIN DD *
  GRANT EXECUTE ON PLAN DSNACLI TO PUBLIC;
/*
//
```

Submit the JCL; expect RC=4 on step BINDCLI.

Bind JDBC default packages and plan

Job: DB7FU.D7F1.NEW.SDSNSAMP(DSNTJJCL). This job is not in your data set. It must be copied from DSN710.SDSNSAMP(DSNTJJCL). You have to do a certain amount of tailoring here:

- ▶ Check the JOB card and JOBPARM.
- ▶ Change every DSN SYSTEM(xxxx) to DSN SYSTEM(D7F1).
- ▶ Change every DSN!!0.SDSNLOAD to DSN710.SDSNLOAD.
- ▶ Change every DSN!!0.SDSNDBRM to DSN710.SDSNDBRM.
- ▶ Add the following step to the end of the job to grant PUBLIC access to the plan:

```
//GRANT EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSIN DD *
  DSN SYSTEM(D7F1)
  RUN PROGRAM(DSNTIAD) PLAN(DSNTIA71) -
  LIBRARY('DB7FU.RUNLIB.LOAD')
  END
//SYSIN DD *
  GRANT EXECUTE ON PLAN DSNJDBC TO PUBLIC;
/*
//
```

Submit the JCL; expect RC=0.

Copy DB2/JDBC HFS

As part of the DB2 V7 SMP/E installation, an HFS data set containing the JDBC/SQLJ classes was created. You now need to clone this HFS data set to make it available in the sysplex.

If you are running with shared HFS support in your sysplex, you only need to copy the DB2 HFS data set once. Refer to Appendix D “Using an alternate HFS structure for product upgrades” in *WebSphere Application Server V4.01 for z/OS and OS/390 Installation and Customization*, GA22-7834 for more information.

If you are not running with shared HFS support, then you will have to copy the DB2 HFS data set to each system in the sysplex where you are running WebSphere.

1. First, dump the SMP/E copy of the DB2 HFS data set:

```
//HFSDUMP JOB (999,POK),'DB2 HFS BACKUP',CLASS=A,MSGCLASS=T,
// NOTIFY=&SYSUID,TIME=1440,REGION=OM
//DUMP EXEC PGM=ADRSSU
//SYSPRINT DD SYSOUT=*
//HFSOUT1 DD DSN=DSN710.BUILD.DB2.SEQ,
// DISP=(NEW,CATLG,DELETE),SPACE=(TRK,(1200,120),RLSE),
// UNIT=3390,VOL=SER=DSN710
//SU.SYSIN DD *
DUMP DATASET(INCLUDE(DSN710.BUILD.DB2.HFS)) -
COMPRESS TOL(ENQF) -
OUTDDNAME(HFSOUT1) ALLDATA(*) ALLEXCP
/*
```

2. Restore the dumped DB2 HFS data set to the target system to be used by D7F1:

```
//HFSREST JOB (999,POK),'DB2 HFS RESTORE',CLASS=A,MSGCLASS=T,
// NOTIFY=&SYSUID,TIME=1440,REGION=OM
//RESTORE EXEC PGM=ADRSSU
//SYSPRINT DD SYSOUT=*
//HFSIN1 DD DSN=DSN710.BUILD.DB2.SEQ,
// DISP=SHR,UNIT=3390,VOL=SER=DSN710
//SU.SYSIN DD *
RESTORE DATASET(INCLUDE(**)) -
RENAMEU(DSN710.BUILD.DB2.HFS, -
DSN710.D7F1.DB2.HFS) -
TOL(ENQF) CANCELERROR -
INDDNAME(HFSIN1)
/*
```

3. Mount the HFS data set at the /usr/lpp/db2 mount point on the target system (D7F1, in this example). From TSO, issue the following commands from a user ID that has superuser authority on that system:

```
MKDIR '/usr/lpp/db2' MODE(7,5,5)
MOUNT FILESYSTEM('DSN710.D7F1.DB2.HFS') MOUNTPoint('/usr/lpp/db2')
TYPE(HFS) MODE(RDWR)
```

Building the second, and subsequent, data sharing systems

When building the second DB2 data sharing system for the residency, we used the following unique data parameters:

DATA SHARING => YES

DATA SHARING FUNCTION => MEMBER

INPUT MEMBER NAME => D7F1

Use the output member that you created when building the first data sharing system as input to building the second system. Many common parameters will already be filled in.

OUTPUT MEMBER NAME => D7F2

When filling in the dialog panels, all entries will be saved in this output member in DSN710.SDSNSAMP

Member Name => D7F2

Work File DB => D7F2

TEMP CLIST LIBRARY => DB7FU.D7F2.NEW.SDSNTEMP

SAMPLE LIBRARY => DB7FU.D7F2.NEW.SDSNSAMP

DB2 subsystem name => D7F2

MAX USERS => 700

MAX BATCH CONNECT => 500

BP0 => 2000

BP1 => 5000

BP2 => 8000

BP7 => 3000

BP32K => 100

BP32K1 => 500

PARAMETER MODULE => DSNZPAF2

DB2 data sharing systems will share the SDSNEXIT data set, but each data sharing member must have a unique ZPARM member. When updating the installation dialogs, enter a unique ZPARM name for each data sharing member.

CACHE DYNAMIC SQL => YES

IRLM SUBSYSTEM NAME => I7F2

IRLM PROC NAME => D7F2IRLM

When running in data sharing mode, you will have a unique IRLM procedure running on each system in the data sharing group in the sysplex.

IRLM MEMBER IDENTIFIER => 2

Make sure you specify a unique IRLMID number for each IRLM procedure. Usually member 2 will have an IRLMID of 2, and so on.

SUBSYSTEM NAME => D7F2

COMMAND PREFIX => -D7F2

Other dialog parameters

The following dialog parameters build the DRDA component of DB2. While not required to be configured for WebSphere, it's helpful during application development to be able to connect to the DB2 database from a workstation.

DB2 STARTUP OPTION => AUTO

DB2 NETWORK LUNAME => SCPD7F2

See VTAM definition below.

DRDA PORT => 33730

RESYNC PORT => 33732

Define DB2 to MVS and build catalog procedures

Job: DB7FU.D7F2.NEW.SDSNSAMP(DSNTIJMV)

- ▶ Add the subsystem information to PARMLIB manually. Delete the DSNTIMP step.
- ▶ Update IEFSSNxx for IRLM and DB2 for the second data sharing system:

SUBSYS SUBNAME(I7F2)

SUBSYS SUBNAME(D7F2) INITRTN(DSN3INI) INITPARM('DSN3EPX,-D7F2,S,D7FG')

- In addition, you can issue the following MVS commands to add the DB2 subsystem definitions dynamically:

```
SETSSI ADD,SUBNAME=I7F2
SETSSI ADD,SUBNAME=D7F2,INITRTN=DSN3INI,INITPARM='DSN3EPX,-D7F2,S,D7FG'
```

- Run the DSNTIPM step to add the procedures to PROCLIB; expect RC=0.
- When you built the first DB2 datasharing system, you added a few DB2 data sets to APF and/or linklst. You must issue the SETPROG APF and/or SETPROG LNKLIST to dynamically add these data sets to the system at this time.

If you do not activate the APF and/or linklst data sets dynamically, or if you do not add the subsystem names dynamically with the **SETSSI** command, you must IPL the system now to pick up the changes.

Define DB2 bootstrap and log data sets

Job: DB7FU.D7F2.NEW.SDSNSAMP(DSNTIJIN). Submit the JCL; expect RC=0.

Initialize DB2 catalog and directory data sets

Job: DB7FU.D7F2.NEW.SDSNSAMP(DSNTIJID). Submit the JCL; expect RC=0.

Define DB2 initialization parameters

Job: DB7FU.D7F2.NEW.SDSNSAMP(DSNTIJUZ). Submit the JCL; expect RC=0.

Update VTAM

Add a VTAM applid for each DB2 subsystem and vary it active. The following example is for D7F2:

```

                                VBUILD TYPE=APPL
*
SCPD7F2  APPL  ACBNAME=SCPD7F2,                                X
                                APPC=YES,                        X
                                ATNLOSS=ALL,                     X
                                AUTH=(ACQ),                      X
                                AUTOSSES=10,                     X
                                DMINWNL=25,                      X
                                DMINWNR=25,                      X
                                DSESLIM=50,                      X
                                EAS=509,                         X
                                ENCR=NONE,                       X
                                MODETAB=AGWTAB,                  X
                                PARSESS=YES,                     X
                                SECACPT=ALREADYV,                X
                                SONSCIP=NO,                      X
                                SRBEXIT=YES,                      X

```

SYNCLVL=SYNCPT,	X
VERIFY=NONE,	X
VPACING=2,	X
VTAMFRR=NO	

Add a RACF Started Class Profile

```
RDEFINE STARTED D7F2*.* OWNER(SYS1) STDATA(USER(DB2STC) +
GROUP(SYS1) TRUSTED(NO) TRACE(YES))
SETROPTS CLASSACT(STARTED) RACLIST(STARTED)
SETROPTS RACLIST(STARTED) REFRESH
```

Start DB2 Version 7

Go into SDSF and start your DB2 subsystem on the system where you want to bring up the second DB2 data sharing system: /-D7F2 STA DB2. The following procedures should start: D7F2MSTR, D7F2IRLM, D7F2DBM1, D7F2DIST and D7F2SPAS.

Create temporary files for DB2

Job: DB7FU.D7F2.NEW.SDSNSAMP(DSNTIJTM). Submit the JCL; expect RC=0 for all steps (however, for step DSNTWR, PROCSTEP DSNTIAS you should expect RC=12).

Copy DB2/JDBC HFS

Restore the dumped DB2 HFS data set to the target system to be used by D7F2. Provided that you already copied the DB2 JDBC HFS data set for the first data sharing member, and provided that you're running shared HFS support, there's no need to copy the JDBC HFS data set again.

```
//HFSREST JOB (999,P0K),'DB2 HFS RESTORE',CLASS=A,MSGCLASS=T,
// NOTIFY=&SYSUID,TIME=1440,REGION=OM
//RESTORE EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//HFSIN1 DD DSN=DSN710.BUILD.DB2.SEQ,
// DISP=SHR,UNIT=3390,VOL=SER=DSN710
//SU.SYSIN DD *
RESTORE DATASET(INCLUDE(**)) -
RENAMEU(DSN710.BUILD.DB2.HFS, -
DSN710.D7F2.DB2.HFS) -
TOL(ENQF) CANCELERROR -
INDDNAME(HFSIN1)
/*
```


Next, mount the HFS data set at the /usr/lpp/db2 mount point on the target system (D7F2, in this example). From TSO, issue the following commands from a user ID on that system which has superuser authority:

```
MKDIR '/usr/lpp/db2' MODE(7,5,5)
MOUNT FILESYSTEM('DSN710.D7F2.DB2.HFS') MOUNTPoint('/usr/lpp/db2')
TYPE(HFS) MODE(RDWR)
```




Sample files

This appendix provides a quick reference to the sample files.

simple.sh

The following sections show the simple.sh template, as well as simple.sh for JDBC and simple.sh for SQLJ.

simple.sh template

```
#!/bin/sh
#-----
# Program Name: simple.sh
#
# Description Name: Shell script used to customize the environment and
#                  to compile and run java samples
#-----
#
# 1. If the DB2 V7 libraries are not in the linklist, export the STEPLIB
#
export STEPLIB=<DB2_HLQ>.SDSNEXIT:<DB2_HLQ>.SDSNLOAD:<DB2_HLQ>.SDSNLOD2
#-----
# 2. Include JAVA_HOME directory.
#    Replace /usr/lpp/java/IBM/J1.3 with your installation directory for Java
#    Replace /usr/lpp/db2/db2710 with your DB2 installation directory
#
export JAVA_HOME=/usr/lpp/java/IBM/J1.3
export DB2_HOME=/usr/lpp/db2/db2710
#-----
# 3. Include the Java bin and the DB2 bin in the PATH statement,
#
PATH=$JAVA_HOME/bin:$DB2_HOME/bin:$PATH
export PATH=$JAVA_HOME/bin/classic:$PATH
#-----
# 4. Include the DB2 library directory
#
export LIBPATH=$DB2_HOME/lib:$LIBPATH
#-----
# 5. Include the DB2 library path in the load library statement
#
export LD_LIBRARY_PATH=$DB2_HOME/lib:$LD_LIBRARY
#-----
# 6. Add additional class (zip, jar) files in CLASSPATH
#
export CLASSPATH=.:$DB2_HOME/classes/db2j2classes.zip:$CLASSPATH
#-----
# 7. Define the DB2SQLJDBC properties file, default is: db2sqljdbc.properties
#
export DB2SQLJPROPERTIES=$DB2_HOME/classes/db2sqljdbc.properties
#-----
if [ "$1" = "c" ]
```

```

then
    echo "$2" " will be compiled."
    javac "$2"
else
    if [ "$1" = "r" ]
    then
        echo "$2" "will be executed."
        java "$2" "$3"
    else
        echo
        echo "To compile the file : simple.sh c <java_file>"
        echo "To run the file      : simple.sh r <class_file> <db2_subsystem_name>"
        echo
    fi
fi
#

```

simple.sh for JDBC

```

#!/bin/sh
export STEPLIB=DB7A7.SDSNEXIT:DB7A7.SDSNLOAD:DB7A7.SDSNLOAD2
export JAVA_HOME=/usr/lpp/java/IBM/J1.3
export DB2_HOME=/usr/lpp/db2/db2710
PATH=$JAVA_HOME/bin:$DB2_HOME/bin:$PATH
export PATH=$JAVA_HOME/bin/classic:$PATH
export LIBPATH=$DB2_HOME/lib:$LIBPATH
export LD_LIBRARY_PATH=$DB2_HOME/lib:$LD_LIBRARY
export CLASSPATH=.:$DB2_HOME/classes/db2j2classes.zip:$CLASSPATH
export DB2SQLJPROPERTIES=$DB2_HOME/classes/db2jdbc.properties
#
if [ "$1" = "c" ];
then
    echo "$2" " will be compiled."
    javac "$2"
else
    if [ "$1" = "r" ];
    then
        echo "$2" "will be executed."
        java "$2" "$3"
    else
        echo
        echo "To compile the file : simple.sh c <java_file>"
        echo "To run the file      : simple.sh r <class_file> <db2_subsystem_name>"
        echo
    fi
fi

```

simple.sh for SQLJ

```
#!/bin/sh
export STEPLIB=DB7A7.SDSNEXIT:DB7A7.SDSNLOAD:DB7A7.SDSNLOAD2
export JAVA_HOME=/usr/lpp/java/IBM/J1.3
export DB2_HOME=/usr/lpp/db2/db2710
PATH=$JAVA_HOME/bin:$DB2_HOME/bin:$PATH
export PATH=$JAVA_HOME/bin/classic:$PATH
export LIBPATH=$DB2_HOME/lib:$LIBPATH
export LD_LIBRARY_PATH=$DB2_HOME/lib:$LD_LIBRARY
export CLASSPATH=.:$DB2_HOME/classes/db2j2classes.zip:$CLASSPATH
export DB2SQLJPROPERTIES=$DB2_HOME/classes/db2sqlj.properties
#
if [ "$1" = "c" ];
then
    echo "$2" " will be compiled."
    javac "$2"
else
    if [ "$1" = "r" ];
    then
        echo "$2" "will be executed."
        java "$2" "$3"
    else
        echo
        echo "To compile the file : simple.sh c <java_file>"
        echo "To run the file      : simple.sh r <class_file> <db2_subsystem_name>"
        echo
    fi
fi
#
```

db2jdbcsqlj.properties

The following sections show db2jdbc.properties and db2sqlj.properties.

db2jdbc.properties

```
# DB2SQLJDBRMLIB=COOK4.DBRMLIB.DATA
# DB2SQLJJDBCPROGRAM=DSNJDBC
# DB2SQLJPLANNAME=DSNJDBC
DB2SQLJSSID=DB7A
DB2SQLJMULTICONTEXT=YES
DB2CURSORHOLD=YES
DB2SQLJATTACHTYPE=RRSAF
# DB2SQLJ_TRACE_FILENAME=/tmp/jdbctrace
# DB2SQLJ_TRACE_BUFFERSIZE=1024
# DB2SQLJ_TRACE_WRAP=0
```

db2sqlj.properties

```
# DB2SQLJDBRMLIB=COOK4.DBRMLIB.DATA
# DB2SQLJJDBCPROGRAM=DSNJDBC
DB2SQLJPLANNAME=SQLBIND
DB2SQLJSSID=DB7A
DB2SQLJMULTICONTXT=YES
DB2CURSORHOLD=YES
DB2SQLJATTACHTYPE=RRSAF
# DB2SQLJ_TRACE_FILENAME=/tmp/jdbctrace
# DB2SQLJ_TRACE_BUFFERSIZE=1024
# DB2SQLJ_TRACE_WRAP=0
```

BIND and INSERT jobs

Here we show various BIND and INSERT jobs.

Sample JDBC BIND job for a local DB2

```
//JDBCBIND JOB <JOBCARD>
//JOB LIB DD DISP=SHR,
//          DSN=DB7A7.SDSNLOAD
//BINDJDBC EXEC PGM=IKJEFT01,DYNAMNBR=20
//DBRMLIB DD DISP=SHR,
//          DSN=DB7A7.SDSNDBRM
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(DB7A)
BIND PACKAGE (DSNJDBC) MEMBER(DSNJDBC1) ISOLATION(UR)
BIND PACKAGE (DSNJDBC) MEMBER(DSNJDBC2) ISOLATION(CS)
BIND PACKAGE (DSNJDBC) MEMBER(DSNJDBC3) ISOLATION(RS)
BIND PACKAGE (DSNJDBC) MEMBER(DSNJDBC4) ISOLATION(RR)
BIND PLAN(DSNJDBC)
          PKLIST(DSNJDBC.DSNJDBC1, -
                  DSNJDBC.DSNJDBC2, -
                  DSNJDBC.DSNJDBC3, -
                  DSNJDBC.DSNJDBC4)
END
/*
```

Sample JDBC BIND job for a remote DB2

```
//DBV6BIND JOB <JOBCARD>
//JOB LIB DD DISP=SHR,
```

```

//          DSN=<hlq_db2_v7>.SDSNLOAD
//BINDJDBC EXEC PGM=IKJEFT01,DYNAMNBR=20
//DBRMLIB DD DISP=SHR,
//          DSN=<hlq_db2_v7>.SDSNDBRM
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(<db2_subsystem_id>)
  BIND PACKAGE (DBRAFJ.DSNJDBC) MEMBER(DSNJDBC1) ISOLATION(UR)
  BIND PACKAGE (DBRAFJ.DSNJDBC) MEMBER(DSNJDBC2) ISOLATION(CS)
  BIND PACKAGE (DBRAFJ.DSNJDBC) MEMBER(DSNJDBC3) ISOLATION(RS)
  BIND PACKAGE (DBRAFJ.DSNJDBC) MEMBER(DSNJDBC4) ISOLATION(RR)
  BIND PLAN(DSNJDBC) ACTION(REPLACE) RETAIN -
    PKLIST(DSNJDBC.DSNJDBC1, -
            DSNJDBC.DSNJDBC2, -
            DSNJDBC.DSNJDBC3, -
            DSNJDBC.DSNJDBC4, -
            DBRAFJ.DSNJDBC.DSNJDBC1, -
            DBRAFJ.DSNJDBC.DSNJDBC2, -
            DBRAFJ.DSNJDBC.DSNJDBC3, -
            DBRAFJ.DSNJDBC.DSNJDBC4)
END
/*

```

Sample SQLJ BIND job for a local DB2

```

//SQLJBIND JOB <JOB CARD>
//JOB LIB DD DISP=SHR,
//          DSN=<HLQ_DB2>.SDSNLOAD
//BINDJDBC EXEC PGM=IKJEFT01,DYNAMNBR=20
//DBRMLIB DD DISP=SHR,
//          DSN=<user-ID>.DBRMLIB.DATA
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(<DB2_subsystem>)
  BIND PACKAGE (SQLJBIND) MEMBER(SQLJBIND1) ISOLATION(UR)
  BIND PACKAGE (SQLJBIND) MEMBER(SQLJBIND2) ISOLATION(CS)
  BIND PACKAGE (SQLJBIND) MEMBER(SQLJBIND3) ISOLATION(RS)
  BIND PACKAGE (SQLJBIND) MEMBER(SQLJBIND4) ISOLATION(RR)
  BIND PLAN(SQLJBIND) DYNAMICRULES(BIND) -
    PKLIST(DSNJDBC.DSNJDBC1, -
            DSNJDBC.DSNJDBC2, -
            DSNJDBC.DSNJDBC3, -
            DSNJDBC.DSNJDBC4, -
            SQLJBIND.SQLJBIND1, -

```



```

                                SQLJBIND.SQLJBIND2,    -
                                SQLJBIND.SQLJBIND3,    -
                                SQLJBIND.SQLJBIND4)
END
//SYSIN      DD      *
GRANT EXECUTE ON PLAN SQLJBIND TO PUBLIC;
//
/*

```

Sample SQLJ BIND job for a remote DB2

```

//DBV6SQLJ JOB <JOB CARD>
//JOB LIB DD DISP=SHR,
//          DSN=<hlq_db2_v7>.SDSNLOAD
//BINDJDBC EXEC PGM=IKJEFT01,DYNAMNBR=20
//DBRMLIB DD DISP=SHR,
//          DSN=<user_id>.DBRMLIB.DATA
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSIN DD      *
DSN SYSTEM(<db2_subsystem_id>)
BIND PACKAGE (DBRAFJ.DSNJDBC) MEMBER(DSNJDBC1) ISOLATION(UR)
BIND PACKAGE (DBRAFJ.DSNJDBC) MEMBER(DSNJDBC2) ISOLATION(CS)
BIND PACKAGE (DBRAFJ.DSNJDBC) MEMBER(DSNJDBC3) ISOLATION(RS)
BIND PACKAGE (DBRAFJ.DSNJDBC) MEMBER(DSNJDBC4) ISOLATION(RR)
BIND PLAN(SQLBIND) ACTION(REPLACE) RETAIN -
          PKLIST(DSNJDBC.DSNJDBC1,    -
                  DSNJDBC.DSNJDBC2,    -
                  DSNJDBC.DSNJDBC3,    -
                  DSNJDBC.DSNJDBC4,    -
                  SQLBIND.SQLBIND1,    -
                  SQLBIND.SQLBIND2,    -
                  SQLBIND.SQLBIND3,    -
                  SQLBIND.SQLBIND4,    -
                  DBRAFJ.DSNJDBC.DSNJDBC1, -
                  DBRAFJ.DSNJDBC.DSNJDBC2, -
                  DBRAFJ.DSNJDBC.DSNJDBC3, -
                  DBRAFJ.DSNJDBC.DSNJDBC4)
END
/*

```

Sample INSERT job

```

//DB2INSERT JOB <jobcard>
//JOB LIB DD DSN=<hlq_db2_v7>.SDSNEXIT,DISP=SHR
//          DD DSN=<hlq_db2_v7>.SDSNLOAD,DISP=SHR
//          DD DSN=<hlq_db2_v7>.SDSNLOD2,DISP=SHR

```

```
//STEP1 EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTPRT DD SYSOUT=*
//SYSTIN DD *
      DSN SYSTEM(DB7A)
      RUN PROGRAM(DSNTEP2) PLAN(DSNTEP71) -
        LIB('<h1q_db2_v7>.RUNLIB.LOAD')
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
INSERT INTO SYSIBM.LOCATIONS (LOCATION, LINKNAME, PORT)
      VALUES ('DBRAFJ','DBRAFJ','33380');
INSERT INTO SYSIBM.IPNAMES (LINKNAME, SECURITY_OUT, USERNAMES, IPADDR)
      VALUES ('DBRAFJ','R',' ','dbrafj.pokplex.poughkeepsie.ibm.com');
SELECT * FROM SYSIBM.LOCATIONS;
SELECT * FROM SYSIBM.IPNAMES;
/*
```

Output from the INSERT job

```
SELECT * FROM SYSIBM.LOCATIONS;
LOCATION LINKNAME IBMREQD PORT TPN
-----
DBRAFJ DBRAFJ N 33380
DSNE610I NUMBER OF ROWS DISPLAYED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100

SELECT * FROM SYSIBM.IPNAMES;
LINKNAME SECURITY_OUT USERNAMES IBMREQD IPADDR
-----
DBRAFJ R N dbrafj.pokplex.poughkeepsie.ibm.com
DSNE610I NUMBER OF ROWS DISPLAYED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
```

Java and SQLJ samples

The following sections show Java and SQLJ sample files.

newsample01.java

```
// NAME = newsample01.java
//
// DESCRIPTIVE NAME = JDBC sample01 application
//
// LICENSED MATERIALS - PROPERTY OF IBM
// 5655-DB2
// (C) COPYRIGHT 1982, 1997 IBM CORP. ALL RIGHTS RESERVED.
//
```

```

//
// DB2 JDBC newsample01.java application:
//
// (a) Create the DB2 for OS/390 JDBC DataSource
// (b) Create Connection instance
// (c) Create a Statement instance
// (d) Execute a Query and generate a ResultSet instance
// (e) Print column 1 (table name) to system.out
// (f) Close the ResultSet
// (g) Close the Statement
// (h) Close the Connection
//

import java.sql.*;          // JDBC base
import javax.naming.*;      // JNDI Naming Services
import javax.sql.*;         // JDBC 2.0 standard ext. APIs
import com.ibm.db2.jcc.*;    // standard extension APIs
// Optional DB2-proprietary imports for error processing
// See "catch" block for processing SQLException and method
"processDB2Diagnosable"
// import com.ibm.db2.jcc.DB2Diagnosable;
// import com.ibm.db2.jcc.DB2Sqlca;

public class newsample01 {

    public static void main(String args[]) {
        try {
            // Create the connection
            DB2DataSource db2ds = new com.ibm.db2.jcc.DB2DataSource();
            db2ds.setDatabaseName(args[0]);
            db2ds.setDescription("Our Sample Database");
            Connection con=db2ds.getConnection();
            System.out.println("***** JDBC Connection to DB2 for OS/390");

            // Create the Statement
            Statement stmt = con.createStatement();
            System.out.println("***** JDBC Statement Created");

            // Execute a Query and generate a ResultSet instance
            // The Query is a Select from SYSIBM.SYSROUTINES
            ResultSet rs = stmt.executeQuery("SELECT NAME FROM
SYSIBM.SYSROUTINES");
            System.out.println("***** JDBC Result Set Created");

            // Print all of the table names to sysout
            while (rs.next()) {
                String s = rs.getString(1);
                System.out.println("Table NAME = " + s);
            }
        }
    }
}

```



```
**** JDBC Disconnect from DB2 for OS/390.  
**** JDBC Exit from class newsample01.
```

```
COOK4:/u/cook4 $
```

newsample02.sqlj

```
// NAME = newsample02.sqlj  
//  
// DESCRIPTIVE NAME = SQLJ newsample02 application  
//  
// LICENSED MATERIALS - PROPERTY OF IBM  
// 5655-DB2  
// (C) COPYRIGHT 1998 IBM CORP. ALL RIGHTS RESERVED.  
//  
//  
// DB2 for OS/390 SQLJ newsample02.sqlj application:  
//  
// (a) Create the DB2 for OS/390 JDBC DataSource  
// (b) Create JDBC Connection instance  
// (c) Create SQLJ Connection Context from JDBC Connection  
// (d) Execute a query, returning an SQLJ result set iterator object  
// (e) Fetch column 1 (table name) from the iterator, and print results to  
to System.out  
// (f) Close the ResultSet  
// (g) Close the Connection  
//  
  
import java.sql.*; //JDBC base  
import javax.naming.*; //JNDI Naming Services  
import javax.sql.*; //JDBC 2.0 standard ext. APIs  
import com.ibm.db2.jcc.*; //standard ext. APIs  
  
import sqlj.runtime.*;  
// import java.io.*;  
  
// connection context declaration  
#sql context newsample02ctx; // generate connection context class  
  
// iterator for the select  
#sql iterator newsample02Iter (String NAME);  
  
public class newsample02 {  
public static void main(String args[])  
throws SQLException {  
newsample02ctx conCtx = null;  
try {  
// Create the connection
```

```

DB2DataSource db2ds = new com.ibm.db2.jcc.DB2DataSource();
db2ds.setDatabaseName(args[0]);
db2ds.setDescription("Our Sample Database");
Connection con=db2ds.getConnection();
System.out.println("**** JDBC Connection to DB2 for OS/390");
con.setAutoCommit(false);          // set Auto Commit OFF
try {
    conCtx = new newsample02ctx(con); // Create an SQLJ ConnectionContext
    System.out.println("**** SQLJ Connection to DB2 for OS/390");
    try {
        newsample02Iter iter;
        int count=0;
        // Issue SELECT
        #sql [conCtx] iter = { SELECT NAME FROM SYSIBM.SYSROUTINES };
        // Retrieve and print resultset
        while (iter.next()) {
            System.out.println(iter.NAME());
            count++;
        }
        System.out.println("Retrieved " + count + " rows of data");
        System.out.println("**** SQLJ Result Set output completed");
    }
    catch( SQLException e ) {
        System.out.println ("**** SELECT SQLException...");
        System.out.println ("Error msg: " + e + ".  SQLSTATE=" +
e.getSQLState() + " SQLCODE=" + e.getErrorCode());
        e.printStackTrace();
    }
    catch( Exception e ) {
        System.out.println("**** NON-SQL exception  = " + e);
        e.printStackTrace();
    }
    // Close the SQLJ connection
    con.commit();
    conCtx.close();
}
catch( SQLException e ) {
    System.out.println ("**** SQLException ...");
    System.out.println ("Error msg: " + e + ".  SQLSTATE=" + e.getSQLState()
+ " SQLCODE=" + e.getErrorCode());
    e.printStackTrace();
}
catch( Exception e ) {
    System.out.println ("**** NON-SQL exception = " + e);
    e.printStackTrace();
}
// Close the connection
con.close();
System.out.println ("**** JDBC Disconnect from DB2 for OS/390.");

```

```

        System.out.println ("**** JDBC exit - NO errors.");
    }
    catch( Exception e ) {
        System.out.println("Exception: " + e.toString() );
        e.printStackTrace();
    }
}
}
}

```

Output from newsample02 with local DB2

```

COOK4:/u/cook4 $ simpleA.sh r newsample02 db7a
newsample02 will be executed.
**** JDBC Connection to DB2 for OS/390
**** SQLJ Connection to DB2 for OS/390
JXFNTK
INSTALL_JAR
REMOVE_JAR
REPLACE_JAR
DSNACCAV
DSNACCDD
DSNACCDE
DSNACCDL
DSNACCDR
DSNACCD S
DSNACCDS
DSNACCMD
DSNACCMG
DSNACCQC
DSNTPSMP
DSNUTILS
DSNWZP
WLM_REFRESH
Retrieved 17 rows of data
**** SQLJ Result Set output completed
**** JDBC Disconnect from DB2 for OS/390.
**** JDBC exit - NO errors.
COOK4:/u/cook4 $

```


Command reference

This appendix provides a quick reference to commands that may be used in setting up and running the environment for WebSphere Application Server.

Sample start and stop commands for WebSphere instances are also provided.

E.1 General commands

Commands prefixed with a forward slash (/) are issued from the SDSF command line.

E.1.1 z/OS

In this section, xx identifies the parmlib member.

- ▶ Add new libraries to the APF authorizationlist and add new libraries to the linklist:

```
/SET PROG=xx
```

- ▶ Modify program properties table information. xx identifies the SCHEDxx member.

```
/SET SCH=xx
```

- ▶ Update the linklist dynamically. (Note that this is a multi-step process. First you need to define a new linklist, based on the current active linklist.)

```
/SETPROG LNK,DEFINE,NAME=wasadd,COPYFROM=CURRENT
```

Change your copy of the linklist by adding and deleting libraries:

```
/SETPROG LNK,ADD,NAME=wasadd,DSNAME=h1q.SBB0LOAD
```

```
/SETPROG LNK,DELETE,NAME=wasadd,DSNAME=BB0.SBB0LD2
```

Activate your linklist, which then becomes the active linklist:

```
/SETPROG LNK,ACTIVATE,NAME=wasadd,
```

- ▶ Add a library to LPA dynamically. If the library is a PDSE, it must be added this way, and not at IPL time:

```
/SETPROG LPA,ADD,MASK=*,DSNAME=h1q.SBB0nnnn
```

- ▶ Remove modules from LPA:

```
/SETPROG LPA,DELETE,MODNAME=BBODASR,FORCE=YES
```

Modules need to be removed one at a time, which can be very cumbersome. Example E-1 shows sample jcl that demonstrates how to do this in a batch job.

Example: E-1 Sample jcl for removing modules in a batch job

```
//BBLDAPDE JOB (999,POK),'LDAP DEL',CLASS=A,REGION=4096K,  
//          MSGCLASS=T,TIME=10,MSGLEVEL=(1,1)  
/*JOBPARM L=999,SYSAFF=SC64  
//  COMMAND 'SETPROG LPA,DELETE,MODNAME=BBODASR,FORCE=YES'  
//  COMMAND 'SETPROG LPA,DELETE,MODNAME=BBODASRP,FORCE=YES'  
//STEPO  EXEC PGM=IEFBR14
```

- ▶ Check on the amount of space left on spool:
/\$DSP00L
- ▶ Display the coupling facility structure that the system logger writes to:
/D LOGGER,STRUCTURE
- ▶ Start the trace writer:
/TRACE CT,WTRSTART=BBOWTR
- ▶ Display the active XCF groups and the members connected to that group:
/DISPLAY XCF

E.1.2 RRS commands

- ▶ Start Resource Recovery Services:
/START RRS,SUB=MSTR
- ▶ Stop Resource Recovery Services:
/STOP RRS

E.1.3 WLM commands

- ▶ Display Workload Management information:
/DISPLAY WLM,APPLENV=*
/DISPLAY WLM,SYSTEMS
- ▶ Control an application environment:
/VARY WLM,APPLENV=entryName,RESUME

E.1.4 UNIX System Services commands

From within the OMVS shell:

- ▶ Display complete file system (space, mounted file systems):
df - display complete file system
- ▶ Display only /WebSphere390 file system
df | grep /WebSphere390

E.1.5 TCP/IP commands

- ▶ Activate new TCP/IP parameters:
`/VARY TCPIP,tcpipoe,0,DSN=tcpipoe.sc64.tcparms(profile)`
- ▶ Display the TCP/IP connections on your system. (**Note:** tcpipf is the stack name.)
`netstat -p tcpipf`

E.2 WebSphere commands

- ▶ Start the LDAP server:
`/S BBOLDAP`
- ▶ Start the WebSphere daemon. (**Note:** It will start the other daemons INTFRP01, NAMING01, and SYSMGT01.)
`/S BBODMN.DAEMON01`
- ▶ Start the WebSphere daemon on the second system. (**Note:** You need to override the daemon name specified in the porc.)
`/S BBODMN.DAEMON02,SRVNAME='DAEMON02'`
- ▶ Stop the WebSphere daemon. (**Note:** It will stop the other WebSphere daemons, including any J2EE servers that are running.)
`/STOP DAEMON01`
- ▶ Start the J2EE (application) servers:
`/S BBOASR2.BBOASR2A`
`/S BBOASR2.BBOASR2B,SRVNAME='BBOASR2B'`
- ▶ Stop the J2EE (application) servers:
`/STOP BBOASR2A`
`/STOP BBOASR2B`

E.3 DB2 commands

- ▶ To start DB2 with subsystem ID D7F1 (- is the DB2 subsystem id character):
`/-D7F1 START DB2`
- ▶ To stop DB2 with subsystem ID D7F2:
`/-D7F2 STOP DB2`

The following commands are issued from a DB2 command prompt.

- ▶ Display the current status for the buffer pool named BP32K:
-DIS BUFFERPOOL(BP32K)
- ▶ Alter the attribute “virtual buffer pool size” to 100 for buffer pool named BP32K:
-ALTER BUFFERPOOL(BP32K) VPSIZE(100)
- ▶ Display current status information about DB2 threads (allied threads, database access threads, parallel task threads). Threads can be active, inactive, indoubt, or postponed.
-DIS THREAD(*)
- ▶ Display information about the status of DB2 databases, table spaces, tables in segmented table spaces, LOB table spaces, index spaces with a database, indexes on auxiliary tables, and partitions of partitioned table spaces and index spaces. LIMIT(nn) limits the number nn of messages displayed by the command.
-DIS DATABASE(*) LIMIT(*)
- ▶ Same as the previous command for the specific database BBOLDAP:
-DIS DATABASE(BBOLDAP) LIMIT(*)
- ▶ Same as -DIS DATABASE(*) LIMIT(*) for the database BBOMDB01, with information about locks:
-DIS DATABASE(BBOMDB01) LIMIT(*) LOCKS
- ▶ Same as -DIS DATABASE(*) LIMIT(*) for the database BBOMDB01, with information about claimed table spaces:
-DIS DATABASE(BBOMDB01) LIMIT(*) CLAIMERS
- ▶ Same as -DIS DATABASE(*) LIMIT(*) for the database BBOMDB01, with information about what is in use, databases and table spaces that have internal DB2 resources allocated:
-DIS DATABASE(BBOMDB01) LIMIT(*) USE

E.4 Commands related to IMS

In this section we discuss various IMS Connect commands.

E.4.1 IMS Connect commands

The following IMS Connect commands can be entered from the z/OS and OS/390 console by specifying the outstanding IMS Connect reply message followed by the command, as follows (where nn is the reply number):

/R nn,<COMMAND>

CLOSEHWS	Terminates IMS Connect (HWS).
OPENDS, <datastore>	Starts a communication session between IMS Connect and the specified datastore (IMS).
OPENPORT <port>	Re-establishes the communication session between IMS Connect and the TCP/IP network through the specified port address.
RECORDER open/close	Opens and closes the line trace dataset that is used to store the line trace information.
SETRACF on/off	Turns the RACF flag on and off.
STOPCLNT <port> <client>	Immediately terminates the communication session between IMS Connect and the specified client using the specified port address.
STOPDS <datastore>	Immediately terminates the communication session between the HWS and the specified datastore (IMS).
STOPPORT <port>	Immediately terminates the communication session between IMS Connect and the TCP/IP network that uses the specified port address.
VIEWDS <datastore>	Displays the current status of the specified datastore (IMS).
VIEWHWS	Displays the current status of IMS Connect.
VIEWPORT <port>	Displays the current status of the communication session between IMS Connect and the specified port.

E.4.2 IMS OTMA commands

This section describes how to use IMS commands to determine the status of the TCP/IP network and IMS Connect.

In the following examples, the following DATASTORE definition is being used:

DATASTORE (ID=IMSB, GROUP=IMS71XCF, MEMBER=HWS710A, TMEMBER=SCSIMS7A)

/DISPLAY OTMA Display XCF information, status and security level about the OTMA server and clients. If the IMS TCP/IP OTMA Connection is ready for use, the output of this command appears as shown in Example E-2:

Example: E-2 Sample output of /DIS OTMA command

GROUP/MEMBER GRPNAME	XCF-STATUS	USER-STATUS	SECURITY
-HWS710A	ACTIVE	SERVER	FULL
-IMS71XCF	ACTIVE	ACCEPT TRAFFIC	

/DISPLAY TMEMBER <name>

Display information about an OTMA client.

/DISPLAY TMEMBER <name> TPIPE <ID>

Display the status of the current transaction member status for OTMA clients and servers. **/DIS TMEMBER ALL** simply shows all the members in the XCF group, similar to the **/DIS OTMA** command output.

/DISPLAY TRACE TMEMBER <name>

Display information about what is being traced.

/DEQUEUE TMEMBER <name> TPIPE <ID>

Remove messages from a Tpipe; specify **PURGE** to remove all messages, or **PURGE1** to remove the first message only.

/START OTMA

Enable communications through OTMA.

/START TMEMBER <name> TPIPE <ID>

Start the named Tpipe.

/STOP OTMA

Stop communications through OTMA.

/STOP TMEMBER <name> TPIPE <ID>

Stop the named Tpipe.

/TRACE

Control the IMS trace.

/SECURE OTMA

Set security options.

/SECURE OTMA NONE

No security checks are made for the transaction.

/SECURE OTMA CHECK

The `UserIdentifier` field of the **MQMD** structure is passed to IMS for transaction or command authority checking. An Accessor Environment Element (ACEE) is built in the IMS control region.

/SECURE OTMA FULL

The `UserIdentifier` field of the **MQMD** structure is passed to IMS for transaction or command authority checking. An ACEE is built in the IMS dependent region, as well as the IMS control region.

/SECURE OTMA PROFILE

The `UserIdentifier` field of the **MQMD** structure is passed to IMS for transaction or command authority checking. The `SecurityScope` field in the **MQI IH** structure is used to determine whether to build an ACEE in the IMS dependent region as well as the IMS control region.

E.4.3 MQSeries commands

RESET TPIPE

Connections between MQSeries and IMS (OTMA) that are out of sync can be resynchronized, whereby the operator has to state whether units of recovery that are associated with the name tpipes are to be committed or backed out.

WLM quick setup

These are the steps for a quick WLM setup.

1. Allocate the WLM couple data set:

Define the WLM couple data set to store the service definition information. If you are running a sysplex with mixed release levels, format the WLM couple data set from the highest level system. The following is a sample JCL to allocate a WLM couple data set.

```
//DEFLOWLM JOB (999,P0K),'AMR',CLASS=A,REGION=4096K,
//          MSGCLASS=T,TIME=10,MSGLEVEL=(1,1),NOTIFY=&SYSUID
//*****
//*  SAMPLE JCL TO FORMAT THE PRIMARY AND ALTERNATE WLM CDS      *
//*****
//STEP1    EXEC PGM=IXCL1DSU
//SYSPRINT DD  SYSOUT=*
//SYSIN     DD  *
           DEFINEDS SYSPLEX(TC1PLEX)
                   MAXSYSTEM(2)
                   DSN(SYS1.XCF.WLM00) VOLSER(TC1CKT)
                   CATALOG
                   DATA TYPE(WLM)
                   ITEM NAME(POLICY) NUMBER(10)
                   ITEM NAME(WORKLOAD) NUMBER(35)
                   ITEM NAME(SRVCLASS) NUMBER(30)
                   ITEM NAME(SVDEFEXT) NUMBER(5)
                   ITEM NAME(SVDCREXT) NUMBER(5)
```

```

ITEM NAME(APPLENV) NUMBER(100)
ITEM NAME(SVAEAEXT) NUMBER(5)
ITEM NAME(SCHENV) NUMBER(100)
ITEM NAME(SVSEAEXT) NUMBER(5)
DEFINEDS SYSPLEX(TC1PLEX)
MAXSYSTEM(2)
DSN(SYS1.XCF.WLM01) VOLSER(TC2CKT)
CATALOG
DATA TYPE(WLM)
ITEM NAME(POLICY) NUMBER(10)
ITEM NAME(WORKLOAD) NUMBER(35)
ITEM NAME(SRVCLASS) NUMBER(30)
ITEM NAME(SVDEFEXT) NUMBER(5)
ITEM NAME(SVDCREXT) NUMBER(5)
ITEM NAME(APPLENV) NUMBER(100)
ITEM NAME(SVAEAEXT) NUMBER(5)
ITEM NAME(SCHENV) NUMBER(100)
ITEM NAME(SVSEAEXT) NUMBER(5)

```

Where:

SYSPLEX

The name of your sysplex as it appears in your COUPLExx parmlib member.

DSN

The name of your WLM couple data set.

VOLSER

A volume that you have access to. If you are using DFSMS, you do not need to specify a VOLSER.

TYPE

The type of function for which this data set is allocated. For a service definition, the type is WLM.

ITEM NAME(POLICY) NUMBER()

Specifies that an increment of space large enough to accommodate the specified number of policies be allocated in the WLM couple data set (Default=5, Minimum=1, Maximum=99).

ITEM NAME(WORKLOAD) NUMBER()

Specifies that an increment of space large enough to accommodate the specified number of workloads be allocated in the WLM couple data set (Default=32, Minimum=1, Maximum=999).

ITEM NAME(SRVCLASS) NUMBER()

Specifies that an increment of space large enough to accommodate the specified number of service classes be allocated in the WLM couple data set (Default=128, Minimum=1, Maximum=999).

ITEM NAME(SVDEFEXT) NUMBER()

Specifies that an exact amount of space (in K bytes) for extension areas to the WLM Service Definition (IWMSVDEF) be allocated in the WLM couple data set (Default=0, Minimum=0, Maximum=8092).

ITEM NAME(SVDCREXT) NUMBER()

Specifies that an exact amount of space (in K bytes) for extension areas to the WLM Service Definition Classification Rules (IWMSVDCR) be allocated in the WLM couple data set (Default=0, Minimum=0, Maximum=8092).

ITEM NAME(APPLENV) NUMBER()

Specifies that an increment of space large enough to accommodate the specified number of application environments be allocated in the WLM couple data set (Default=100, Minimum=1, Maximum=999).

ITEM NAME(SVAEAEXT) NUMBER()

Specifies that an exact amount of space (in K bytes) for extension areas to the WLM Service Definition Application Environment Area (IWMSVAEA) be allocated in the WLM couple data set (Default=0, Minimum=0, Maximum=8092).

ITEM NAME(SCHENV) NUMBER()

Specifies that an increment of space large enough to accommodate the specified number of scheduling environments be allocated in the WLM couple data set (Default=100, Minimum=1, Maximum=999).

ITEM NAME(SVSEAEXT) NUMBER()

Specifies that an exact amount of space (in K bytes) for extension areas to the WLM Service Definition Scheduling Environment Area (IWMSVSEA) be allocated in the WLM couple data set (Default=0, Minimum=0, Maximum=8092).

2. Make WLM couple data set available.

To make the WLM couple data set available to the sysplex, do the following:

a. Use the SETXCF command.

- To make a primary WLM couple data set called SYS1.XCF.WLM00 available to the sysplex, enter the following command:

```
SETXCF COUPLE,TYPE=WLM,PCOUPLE=SYS1.XCF.WLM00
```

- To make an alternate WLM couple data set called SYS1.XCF.WLM01 available to the sysplex, enter this command:

```
SETXCF COUPLE,TYPE=WLM,ACOUPLE=SYS1.XCF.WLM01
```

b. Update your COUPLExx for any subsequent IPLs:

```
DATA TYPE(WLM)
PCOUPLE(SYS1.XCF.WLM00)
ACOUPLE(SYS1.XCF.WLM01)
```

3. Control access to the WLM couple data set.

To determine who needs access to the WLM couple data set itself, define the kind of access authority required, as follows:

– READ

With READ access, the user can extract a service definition from the WLM couple data set.

– UPDATE

With UPDATE access, the user can:

- Do all the functions available for READ access.
- Install a service definition to a WLM couple data set.
- Activate a service policy.

Example:

```
RDEFINE FACILITY MVSADMIN.WLM.POLICY UACC(NONE)
PERMIT MVSADMIN.WLM.POLICY CLASS(FACILITY) ID(user1) ACCESS(READ)
PERMIT MVSADMIN.WLM.POLICY CLASS(FACILITY) ID(user2) ACCESS(UPDATE)
```

4. Install the service definition on the WLM couple data set.

Before you can activate a service policy, you need to install the service definition on the WLM couple data set.

There is a sample service definition, provided by the Washington Systems Center, that is intended to give examples of the various WLM constructs.

Go to the URL:

<http://www.ibm.com/servers/eserver/zseries/zos/wlm/>

Select Sample Service Definition, follow the steps to download the PDS, then invoke the WLM ISPF Application IWMARIN0 to view and make changes to the service definition. Then Activate the new service policy.

Also, there is a Goal Mode Migration Aid, which is a semi-automated tool that assists you in migrating your system performance goals from compatibility mode to a service definition for goal mode. It helps you to reduce the editing effort of the migration process and assists you in mapping your existing performance group definitions to service classes and to find service goals for a service policy.

For more information about WLM, see *z/OS V1R2.0 MVS Planning: Workload Management*, SA22-7602.

Example of adding Web Server definitions to WLM

This is an example of how to add the Web server definitions for scaling mode in WLM.

If there is no installation-defined ISPF option to provide access to WLB, enter the following TSO command:

```
ex 'sys1.sblscli0(iwmarin0)'
```

The WLM welcome screen will pop up.

Press Enter, and the pop-up menu shown in Example F-1 is displayed.

Example: F-1 WLM: Choose the service definition

Choose Service Definition

Select one of the following options.

- 2 1. Read saved definition
2. Extract definition from WLM couple data set
3. Create new definition

Enter 2 as shown in Example F-1 to select the Extract definition from the WLM couple data set.

Example: F-2 WLM: Definition menu

File	Utilities	Notes	Options	Help
------	-----------	-------	---------	------

Functionality	LEVEL004	Definition Menu	WLM App1	LEVEL008
---------------	----------	-----------------	----------	----------

Command ==> _____

Definition data set . . . : none

Definition name DB2JSP2 (Required)
Description Stored procedure policy/DB2 Util

Select one of the
following options. ____ 1. Policies

- [illegible]

Defining the workload

Example: F-3 WLM: Workload selection list

Place your cursor on the Workload menu bar option and press Enter. This displays the pull-down menu shown in Example F-4 on page 405.

Example: F-4 WLM: Workload pull-down menu

- 1. Create
- 2. Copy
- 3. Modify
- 4. Browse
- 5. Print
- 6. Delete
- 7. Exit

Select option 1, Create. The screen shown in Example F-5 is displayed. Enter the data that is in bold type.

Example: F-5 WLM: Create a workload

Workload Notes Options Help

Create a Workload

Command ==> _____

Enter or change the following information:

Workload Name **WEBWORK** (Required)

Description **Web Workload**_____

Press PF3 (not Enter) to save this definition. You are returned to the Workload Selection List where your new definition is displayed, as shown in Example F-6.

Example: F-6 WLM: Workload Selection List showing new workload

Workload View Notes Options Help

Workload Selection ListRow 1 to 9 of 9

Command ==> _____

Action Codes: 1=Create, 2=Copy, 3=Modify, 4=Browse, 5=Print, 6=Delete,
 /=Menu Bar

		----Last Change----	
Action	Name	Description	User Date
___	BATCH	Batch work	JOB 1993/10/19
___	CHAT	SG24-5660 test	SJTRES1 1999/12/16
___	CICS	CICS work	JOB 1993/10/19
___	DB2RES	DB2 stored procedures	DB2RES2 2000/03/08
___	IMS	IMS work	JOB 1993/10/19
___	OMVS	UNIX Service Workload	HERZOG 1998/07/22
___	OTHER	Other work	JOB 1993/10/19
___	TSO	TSO work	JOB 1993/10/19
___	WEBWORK	Web Workload	MCNAB 2000/05/16
***** Bottom of data *****			

Press PF3 again to return to the Definition Menu.

Defining the service class

From the Definition Menu, select option 4, Service Classes. This displays the screen shown in Example F-7.

Example: F-7 WLM: Service Class Selection List

Service-Class View Notes Options Help			

Service Class Selection List			Row 1 to 22 of 46
Command ==> _____			
Action Codes: 1=Create, 2=Copy, 3=Modify, 4=Browse, 5=Print, 6=Delete, /=Menu Bar			
Action	Class	Description	Workload
___	BATCHHI	Importance 4 batch for cbw2	BATCH
___	BATCHLOW	Importance 5 batch for cbw2	BATCH
___	BATCHSPL	Batch high priority	BATCH
___	BATCHTR	Batch Thrashers	BATCH
___	CHAT	SG24-5660 USS WLM test	CHAT
___	CICSDFLT	CICS default service class	CICS
___	CICSRGN	CICS regions vel70	CICS
___	CICSWORK	CICS classified work	CICS
___	CICS12	CICS trans from LU 1 and 2	CICS
___	CICS345	CICS trans from LU 3, 4, 5	CICS
___	CICS678	CICS trans from LUs 6, 7, 8	CICS
___	CICS90	CICS trans from LUs 9, and 0	CICS
___	CPSMLONG	Long running CPSM transactions	CICS
___	CPSMTRAN	CPSM transactions	CICS
___	DISC	Discretionary class	BATCH
___	HIGHPRT	test sched + 1 ad spc	DB2RES
___	IMS	IMS transactions	IMS
___	IMSCTL	IMS control regions vel70	OTHER
___	IMSMP	IMS MPRs/MPPs vel50	OTHER
___	IMS1	IMS hr* transactions	IMS
___	OMVS	UNIX Services	OMVS
___	OURBATHI	Importance 3 batch for ls2106	BATCH

Place your cursor on the Service-Class menu bar option and press Enter. From the pull-down menu, select option 1, Create. This displays the screen shown in Example F-8. Enter the text that is in bold type.

Example: F-8 WLM: Create a Service Class

Service-Class Notes Options Help			

Service Class Name **WEBSRVC** (Required)

Description **WEB Service Class**

Workload Name **WEBWORK** (name or ?)

Base Resource Group _____ (name or ?)

Specify BASE GOAL information. Action Codes: I=Insert new period,
E=Edit period, D=Delete period.

---Period--- -----Goal-----

Action # Duration Imp. Description

***** Bottom of data *****

Press Enter and you are prompted as follows:

A goal is required. Use 'I' to insert the first period goal. (IWMAM414)

Put an I in the Action field and press Enter to bring up the pop-up menu.

Enter 1 to select Average response time. On the screen that is displayed, enter the text that is in bold type, as shown in Example F-9.

Example: F-9 WLM: Average response time goal

Average response time goal

Enter a response time of up to 24 hours for period 1

Hours (0-24)

Minutes (0-99)

Seconds **0.3** (0-9999)

Importance . . **2** (1=highest, 5=lowest)

Duration . . . _____ (1-999,999,999, or
none for last period)

Press Enter and you are returned to the Create a Service Class screen, with the goal information filled in, as shown in Example F-10.

Example: F-10 WLM: Create a Service Class completed

Service-Class Notes Options Help

Create a Service Class

Row 1 to 2 of 2

Command ==>

Service Class Name WEBSRVC (Required)
Description Web Service Class
Workload Name WEBWORK (name or ?)
Base Resource Group _____ (name or ?)

Specify BASE GOAL information. Action Codes: I=Insert new period,
E=Edit period, D=Delete period.

---Period---		-----Goal-----	
Action #	Duration	Imp.	Description
1	2		Average response time of 00:00:00.300
***** Bottom of data *****			

Press PF3 to save the service class and return to the Service Class Selection List. Press PF3 again to return to the Definition Menu.

Defining/modifying the subsystem type

The IWEB subsystem type is used by WLM to classify the workload for the HTTP server. It may already be defined, or you may need to add it. Either way, make sure that the definitions are as specified here.

From the Definition Menu, enter 6 to select Classification Rules. This displays the screen shown in Example F-11.

Example: F-11 WLM: Subsystem Type Selection List for Rules

Subsystem-Type View Notes Options Help			

Subsystem Type Selection List for Rules			Row 1 to 14 of 14
Command ==> _____			
Action Codes: 1=Create, 2=Copy, 3=Modify, 4=Browse, 5=Print, 6=Delete, / =Menu Bar			
-----Class-----			
Action	Type	Description	Service Report
___	ASCH	Use Modify to enter YOUR rules	
___	CHAT	WLM USS SG24-5660 test	CHAT
___	CICS	Use Modify to enter YOUR rules	
___	DB2	Use Modify to enter YOUR rules	SCDB2STP
___	DDF	DRDA Stored Procedures	SCDB2STP
___	IMS	Use Modify to enter YOUR rules	
___	JES	Use Modify to enter YOUR rules	OURBATMD
___	KAAB	WEB Scalable Subsystem ***kaa***	TSO0DD
___	LSFM	Use Modify to enter YOUR rules	
___	OMVS	Use Modify to enter YOUR rules	OMVS
___	SAP	WLM definition for SAP R/3 4.5B	SAPAS DBK1
___	SOM	Use Modify to enter YOUR rules	

—	CHATWLM	WLM ChatServer test SG24-5660
—	DBSGWLM1	SP for DB2 Residency SG24-5486
—	DB2JODBA	SP for DB2J for PeggyR for ODBA
—	DB2JREXX	SP for DB2J for PeggyR for REXX
—	DB2JSP1	SP for DB2J for PeggyR for SPB
—	DB2JSP2	SP for DB2J for Java
—	DB2JUTIL	SP for DSNUTILS for PeggyR
—	KAAWEBQ	HTTP Queue Server
—	KAAWEBQ1	HTTP Queue Server
—	WLM_ENVIRONMENT_03	Underscore, but in quotes
—	WLM_ENV4	Quotes around underscore
—	WLM_JAVA_DB2V	SG24-5485 DB2V WLM addr sp JAVA
—	WLM_JAVA_DB21	SG24-5485 DB21 WLM addr sp JAVA
—	WLMAUTO	WLMENV2 with &IWMSSNM
—	WLMDDB2V2	SG24-5485 DB2V WLM addr sp
—	WLMENVI	SP for DB2J for PeggyR for ODBA
—	WLMENVM	Multi-Env for REXX + ODBC/CLI

Place your cursor on the Application-Environment menu bar option and press Enter. From the pop-up menu, select option 1, Create. This displays the screen shown in Example F-14. Enter the text that is in bold type.

Example: F-14 WLM: Create an Application Environment

```

Application-Environment  Notes  Options  Help
-----
                                Create an Application Environment
Command ==> _____

Application Environment . . . WEBHTML                                Required
Description . . . . . WebServer Appl Environment
Subsystem Type . . . . . IWEB Required
Procedure Name . . . . . IWMIWM
Start Parameters . . . . . IWMSSN=&IWMSSNM,IWMAE=WEBHTML_____
                                _____
                                _____

Limit on starting server address spaces for a subsystem instance:
1  1. No limit
   2. Single address space per system
   3. Single address space per sysplex

```

Press PF3 to return to the Application Environment Selection List with your new definition listed,

You should receive the following confirmation message:

Application environment WEBHTML was created. (IWMAM604)

Press PF3 again to return to the Definition Menu.

Installing the definition

The changes made in the previous sections need to be written back to the couple data sets. From the Definition Menu, place your cursor on the Utilities menu bar option and press Enter. Select option 1 to install the definition and press Enter.

You are returned to the Definition Menu with the following confirmation message:

Service definition was installed. (IWMAM038)

Activating the policy

From the Utilities pull-down menu, select option 3, activate the service policy, and press Enter. This displays the Policy Selection List, as shown in Example F-15.

Example: F-15 WLM: Policy Selection List

Policy Selection List	Row 1 to 6 of 18
Command ==> _____	

The following is the current Service Definition installed on the WLM couple data set.

Name : DB2JSP2

Installed by : AMR from system SC61

Installed on : 2001/11/16 at 15:04:46

Select the policy to be activated with "/"

Sel	Name	Description
-	ALLCENT	Policy for all central runs
-	CAPNONE	Policy for capping run
-	CAP1000	Policy for capping run
-	CAP1500	Policy for capping run
-	CAP2000	Policy for capping run
-	CHAT	SG24-5660 test

Page down to the required policy, if necessary, and type a slash (/) next to it to select it. Press Enter to activate it. You are returned to the Definition Menu with the following confirmation message:

Service policy POLTEST was activated. (IWMAM060)

Exit WLM.

RACF REXX sample

```

/* REXX ----- */
/*
/* Basic Assumptions -
/*
/* The proc name for the control region and server region is the
/* same as the userid associated with the region.
/*
/* Here is the naming scheme:
/*
/* CBserver name           = BBXXXX
/* - APPLENV name          = BBXXXX
/* CBserver instance name  = BBXXXXAY
/* - ORBSrvname default value = BBXXXXAY
/* Userid for control region = BBXXXXC
/* - PROC for control region = BBXXXXC
/* Group id for control region = BBXXXXG
/* Userid for server region = BBXXXXS
/* - PROC for server region = BBXXXXS
/* Group id for server region = BBXXXXR
/* Default remote userid    = BBXXXXI
/* Default local  userid    = BBXXXXD
/* Group id for default ids  = BBXXXXP
/*
/* So basically, the application server is distinguished from all
/* other servers by characters 2-6 (i.e., 'XXXX'). The server
/* instance, one or more on each system, is distinguished by

```

```

/* the 8th character (i.e., 'Y') */
/* */
/* Here are the userids/UIDs, change as desired. */
/* BBXXXXC 0 - do not change. */
/* BBXXXXS 1100 */
/* BBXXXXD 1101 */
/* BBXXXXI 1102 */
/* HerE are the groups/GIDS, change as desired. */
/* BBXXXXG 1000 */
/* BBXXXXR 1001 */
/* BBXXXXP 1002 */
/* */
/* */
/* Customization: */
/* */
/* Set the appl_id to the 4-character project identifier */
/* the appl_instance to the correct value. */
/* */
/* If desired, set the UID and GID values to something meeting */
/* your installation standards. */
/* */
/* ----- */

```

trace r

```

default_logstream_name           = "WAS.ERROR.LOG"

appl_id                          = "PROD"
appl_instance                    = "1"

default_appl_CR_proc_name        = "BB" || appl_id || "C"
say default_appl_CR_proc_name
default_appl_CR_userid           = default_appl_CR_proc_name
default_appl_CR_group            = "BB" || appl_id || "G"
default_appl_CR_UID              = "2170"
default_appl_CR_GID              = "2270"

default_appl_SR_proc_name        = "BB" || appl_id || "S"
default_appl_SR_userid           = default_appl_SR_proc_name
default_appl_SR_group            = "BB" || appl_id || "R"
default_appl_SR_UID              = "2171"
default_appl_SR_GID              = "2271"

default_appl_generic_server_name = "BB" || appl_id
default_appl_server_instance_name = "BB" || appl_id || "A" || ,
                                   appl_instance

default_appl_unauth_local_userid = "BB" || appl_id || "D"
default_appl_unauth_local_uid    = "2172"

```



```

default_appl_unauth_remote_userid      = "BB" || appl_id || "I"
default_appl_unauth_remote_uid        = "2173"
default_appl_unauth_group              = "BB" || appl_id || "P"
default_appl_unauth_GID                = "2272"

default_CB_admin_group                 = "CBADMGP"
default_CB_CFG_group                   = "CBCFG1"
default_CB_CR_group                    = "CBCTL1"

```

ADDRESS TSO

```

/* ----- */
/*
/* Create RACF groups for the Application control and server
/* regions and for the default client ids.
/*
/* ----- */
if appl_id = 'XXXX' then do
  say 'Error... configure the appl_id value and re-try.'
  exit(-1)
end
if appl_instance = 'Y' then do
  say 'Error... configure the appl_instance value and re-try.'
  exit(-1)
end
if default_logstream_name = '' then do
  say 'Error... configure the default_logstream_name and re-try.'
  exit(-1)
end
say 'Adding groups for Application control and server regions'
"ADDGROUP " || default_appl_CR_group || ,
  " OMVS(GID(" || default_appl_CR_GID || "))"

"ADDGROUP " || default_appl_SR_group || ,
  " OMVS(GID(" || default_appl_SR_GID || "))"

"ADDGROUP " || default_appl_unauth_group || ,
  " OMVS(GID(" || default_appl_unauth_GID || "))"

/* ----- */
/*
/* Define the user ids for Application Control and server regions.
/* Connect these ids to default_CB_CFG_group so HFS can be read.
/*
/* ----- */

say 'Adding users for IVP Control and Server Regions'

"ADDUSER " || default_appl_CR_userid || ,

```

```

" DFLTGRP(" || default_appl_CR_group || ,
" ) OMVS(UID(" || default_appl_CR_UID || ,
" ) HOME(/tmp) PROGRAM(/bin/sh)) NAME('CB390 " || appl_id || " CR') "

"ADDUSER " || default_appl_SR_userid || ,
" DFLTGRP(" || default_appl_SR_group || ,
" ) OMVS(UID(" || default_appl_SR_UID || ,
" ) HOME(/tmp) PROGRAM(/bin/sh)) NAME('CB390 " || appl_id || " SR') "

say 'Connecting IVP CR and SR userids to the CB configuration group.'

"CONNECT " || default_appl_CR_userid || ,
" group(" || default_CB_CFG_group || ")"

"CONNECT " || default_appl_CR_userid || ,
" GROUP(" || default_CB_CR_group || ")"

"CONNECT " || default_appl_SR_userid || ,
" group(" || default_CB_CFG_group || ")"

/* ----- */
/*
/* Define the default userids for local and remote non-
/* authenticated users
/*
/* ----- */

say 'Adding default users for installation'

"ADDUSER " || default_appl_unauth_local_userid || ,
" DFLTGRP(" || default_appl_unauth_group || ,
" ) OMVS(UID(" || default_appl_unauth_local_UID || ,
" ) HOME(/tmp) PROGRAM(/bin/sh)) NAME('CB390 " || appl_id || ,
" LOCAL USER') "

"ADDUSER " || default_appl_unauth_remote_userid || ,
" DFLTGRP(" || default_appl_unauth_group || ,
" ) OMVS(UID(" || default_appl_unauth_remote_UID || ,
" ) HOME(/tmp) PROGRAM(/bin/sh)) NAME('CB390 " || appl_id || ,
" REMOTE USER') "

/* ----- */
/*
/* Assigning userids to X.SERVER to control region
/*
/* ----- */

say 'Assigning userids to started tasks'

```

```

"RDEF STARTED " || default_appl_CR_proc_name || ,
  ".* STDATA(USER(" || default_appl_CR_userid || ,
    ") GROUP(" || default_appl_CR_group || ,
    ") TRACE(YES))"

"RDEF STARTED " || default_appl_SR_proc_name || ,
  ".* STDATA(USER(" || default_appl_SR_userid || ,
    ") GROUP(" || default_appl_SR_group || ,
    ") TRACE(YES))"

/* ----- */
/*
/* Allowing access to CBIND, SERVER and SOMDOBJs profiles
/*
/* ----- */

say 'Defining CBIND CB.BIND.generic_server with control access.'

"RDEFINE CBIND CB.BIND." || default_appl_generic_server_name || ,
  " UACC(READ)"

"Permit CB.BIND." || default_appl_generic_server_name || ,
  " CLASS(CBIND) ID(" || default_CB_CR_group || ,
  ") ACC(CONTROL)"

say 'Defining CBIND CB.generic_server with read access.'

"RDEFINE CBIND CB." || default_appl_generic_server_name || ,
  " UACC(READ)"

say 'Defining SOMDOBJs servername.*.* with alter access.'

"RDEFINE SOMDOBJs " || default_appl_generic_server_name || ,
  ".*.* " || ,
  " UACC(ALTER) "

say 'Defining SERVER CB.server_instance.generic_server w/ read access.'

"RDEFINE SERVER CB.*." || ,
  default_appl_generic_server_name || ,
  " UACC(NONE) "
"PERMIT CB.*." || ,
  default_appl_generic_server_name || ,
  " CLASS(SERVER) ID(" || default_appl_SR_userid || ,
  ") ACC(READ)"

/* ----- */
/*

```

```

/* Allowing CR and SR to write to the logstream */
/*
/* ----- */

say 'Defining BPX.SERVER with update access.'

"PERMIT" default_logstream_name "CLASS(LOGSTRM)" || ,
    "ID(" || default_appl_CR_group || ") ACCESS(UPDATE)"
"PERMIT" default_logstream_name "CLASS(LOGSTRM)" || ,
    "ID(" || default_appl_SR_group || ") ACCESS(UPDATE)"

/* ----- */
/* Optional Step: */
/* Allowing our Web server to bind to the J2EE Server */
/*
/* ----- */
/* "PE CB.BIND.BB" || appl_id, */
/* " CLASS(CBIND) ", */
/* " ID(WEBSRV) ", */
/* " ACCESS(CONTROL) " */
/*
/* ----- */
/*
/* Wrap it up. */
/*
/* ----- */

say 'Refreshing the world '

"SETR RACLIST(CBIND) GENERIC(CBIND) REFRESH"
"SETR RACLIST(FACILITY) GENERIC(FACILITY) REFRESH"
"SETR RACLIST(SERVER) GENERIC(SERVER) REFRESH"
"SETR RACLIST(SOMDOBJ)S GENERIC(SOMDOBJ)S REFRESH"
"SETR RACLIST(STARTED) GENERIC(STARTED) REFRESH"

```

Additional material

This redbook refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

<ftp://www.redbooks.ibm.com/redbooks/SG245981>

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the redbook form number, SG24-5981.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

<i>File name</i>	<i>Description</i>
sample_files.zip	Code samples in downloadable form from Appendix D, “Sample files” on page 377 and Appendix G, “RACF REXX sample” on page 413.

System requirements for downloading the Web material

Not applicable.

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 424.

- ▶ *e-business Cookbook for z/OS Volume II: Infrastructure*, SG24-5981
- ▶ *e-business Cookbook for z/OS Volume III: Java Development*, SG24-5980
- ▶ *Enterprise JavaBeans for z/OS and OS/390 WebSphere Application Server V4.0*, SG24-6283
- ▶ *EJB Development with VisualAge for Java for WebSphere Application Server*, SG24-6144
- ▶ *Design and Implement Servlets, JSPs, and EJBs for IBM WebSphere Application Server*, SG24-5754
- ▶ *Enterprise JavaBeans for z/OS and OS/390 CICS Transaction Server V2.1*, SG24-6284
- ▶ *Revealed! Architecting Web Access to CICS*, SG24-5466
- ▶ *Migrating WebSphere Applications to z/OS*, SG24-6521
- ▶ *IMS e-business Connectors: A Guide to IMS Connectivity*, SG24-6514
- ▶ *IMS V7 and Java Application Programming*, SG24-6123
- ▶ *Java Connectors for CICS: Featuring the J2EE Connector Architecture*, SG24-6401
- ▶ *CICS Transaction Gateway V3.1 The WebSphere Connector for CICS*, SG24-6133
- ▶ *WebSphere Version 4 Application Development Handbook*, SG24-6134
- ▶ *IBM Web-to-Host Integration Solutions*, SG24-5237
- ▶ *Enterprise Integration with IBM Connectors and Adapters*, SG24-6122
- ▶ *From code to deployment: Connecting to CICS from WebSphere V4.01 for z/OS*, REDP0206

Other resources

These publications are also relevant as further information sources:

- ▶ *Configuring Web Applications* white paper, WP100238, available at:
www.ibm.com/support/techdocs
- ▶ *WebSphere Application Server V4.0.1 for z/OS and OS/390: Installation and Customization*, GA22-7834
- ▶ *WebSphere Application Server V4.0.1 for z/OS and OS/390: Messages and Diagnosis*, GA22-7837
- ▶ *WebSphere Application Server V4.0.1 for z/OS and OS/390: Operations and Administration*, SA22-7835
- ▶ *WebSphere Application Server V4.0.1 for z/OS and OS/390: Assembling Java 2 Platform, Enterprise Edition (J2EE) Applications*, SA22-7836
- ▶ *WebSphere Application Server V4.0.1 for z/OS and OS/390: Assembling CORBA Applications*, SA22-7848
- ▶ *WebSphere Application Server V4.0.1 for z/OS and OS/390: SystemManagement User Interface*, SA22-7838
- ▶ *WebSphere Application Server V4.0.1 for z/OS and OS/390: Migration*, GA22-7860
- ▶ *WebSphere Application Server V4.0.1 for z/OS and OS/390: System Management Scripting Application Programming Interface*, SA22-7839
- ▶ *Data Sharing: Planning and Administration Version 7*, SC26-9935
- ▶ *DB2 Universal Database for OS/390 and z/OS Version 7 Installation Guide*, GC26-9936
- ▶ *DB2 Universal Database for OS/390 and z/OS: Program Directory Version 7*, GI10-8216
- ▶ *DB2 Universal Database for OS/390 and z/OS: Application Programming Guide and SQL Guide Version 7*, SC26-9933
- ▶ *MQSeries System Setup Guide*, SC34-5651
- ▶ *MQSeries Command Reference*, SC33-1369
- ▶ *IMS Connector for Java User's Guide and Reference*, SC27-1559
- ▶ *IMS Connect Guide and Reference*, SC27-0946
- ▶ *IMS/ESA Open Transaction Manager Access Guide and Reference*, SC26-8743
- ▶ *z/OS V1R2.0 CS: IP Configuration Guide*, SC31-8775
- ▶ *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776

Referenced Web sites

These Web sites are also relevant as further information sources:

- ▶ Java on OS/390 and z/OS
<http://www-1.ibm.com/servers/eserver/zseries/software/java/>
- ▶ VisualAge Developer Domain
<http://www7.software.ibm.com/vad.nsf>
- ▶ WebSphere Application Server for z/OS and OS/390
http://www-3.ibm.com/software/webervers/appserv/zos_os390/index.html
- ▶ AlphaWorks
<http://www.alphaworks.ibm.com/>
- ▶ WebSphere Application Server for z/OS and OS/390 support page
http://www-4.ibm.com/software/webervers/appserv/zos_os390/support.html
- ▶ DeveloperWorks
<http://www-106.ibm.com/developerworks/>
- ▶ WebSphere Application Server for OS/390 and z/OS Library
http://www-3.ibm.com/software/webervers/appserv/zos_os390/library.html
- ▶ z/OS Internet Library
<http://www-1.ibm.com/servers/eserver/zseries/zos/bkserv/>
- ▶ Connector Architecture for WebSphere Application Server
<http://www7b.boulder.ibm.com/wsdd/downloads/jca.html>
- ▶ Using J2EE Resource Adapters in a Non-Managed Environment
http://www7b.boulder.ibm.com/wsdd/library/techarticles/0109_kelle/0109_kelle.html
- ▶ WebSphere InfoCenter
<http://www-4.ibm.com/software/webervers/appserv/doc/v40/aee/index.html>
- ▶ Technical Support Technical Information Site
<http://www-1.ibm.com/support/techdocs/atmastr.nsf>
- ▶ Java Servlet site
<http://java.sun.com/products/servlet/index.html>
- ▶ Java Connector Architecture
<http://java.sun.com/j2ee/connector/>
- ▶ WebSphere Application Server for z/OS Tools

<http://www6.software.ibm.com/dl/websphere20/zosos390-p>

- ▶ Documentation site for HTTP server

<http://www.ibm.com/software/websphere/httpservers/doc53.html>

- ▶ WebSphere Troubleshooter

<http://www.ibm.com/software/webservers/httpservers/troubleshooter.html>

- ▶ SSL security setup

<http://www-3.ibm.com/software/webservers/httpservers/gskkyman.htm>

How to get IBM Redbooks

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following Web site:

ibm.com/redbooks

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.

Index

A

ARM (Automatic Restart Manager) 14

C

Call Attach Facility (CAF) 133
CICS 5
CICS Connector for CICS TS 188
CICS ECI resource adapter
 installation 191
 preparing 190
CICSConnectionSpec 188
Common Connector Framework (CCF) 187
Configuration Option
 enabling 28
container 5
CORBA (MOFW) 6
coupling facility 3
Coupling Facility (CF) 6
ctgclient.jar 188

D

DB2
 BBOJDB01 5
 BBOMDB01 5
DB2 UDB for OS/390 and z/OS 5
Domain Name Server (DNS) 347

E

ECIInteractionSpec 188
ECIRequest class 188
Enabling CICS connector support 190
eNetwork Communication Server 5
Enterprise Archive file (EAR) 34
Enterprise Information Systems 189
Enterprise Java Beans (EJBs) 5
EPIInteractionSpec 188
EZA ZSSI procedure 338

H

Hierarchical 5
Hierarchical File System (HFS) 5

contents 284

I

IBM HTTP Server for OS/390
 workload scaling
 activating the Policy in WLM 411
 defining an Application Environment in WLM 409
 defining the Service Class in WLM 406
 defining the Workload in WLM 404
 defining/modifying the Subsystem Type in WLM 408
 installing the WLM Definition 411
IBM WebSphere Application Server for OS/390
 basic setup
 httpd.conf directives, example 26
 information resources 10
 product information, Web site 11
 related documentation 11
 WebSphere Troubleshooter for OS/390 11
IMS 5
IMS Connect (V7)
 commands 396
 OPENPORT 396
 RECORDER 396
 SETRACF 396
 STOPCLNT 396
 STOPDS 396
 STOPPORT 396
 VIEWDS 396
 VIEWHWS 396
 VIEWPORT 396
 configuration statement parameters 240
 installing 236
 user exits
 customizing 237
 HWSJAVA0 245
 HWSSMPL0 245
 HWSUINIT 245
 HWSYDRU0 245
IMS Connect / IMS TOC
 configuration member 240
 documentation 258

- helpful commands
 - /DIS OTMA 397
- Installation Verification Program (IVP) 250
- MVS PPT
 - entry 239
 - updates 239
- overview 234
- user exits, installing 244
- IMS Connector for Java 258
 - prerequisites 262
- IR (Interface Repository) 6

J

- Java Database Connectivity (JDBC)
 - advantages 119
 - CLASSPATH 128
 - db2sqljjdbc.properties file 130
 - installation
 - DB2SQLJATTACHTYPE, in db2sqljjdbc.properties 132–134
 - DB2SQLJPLANNAME, in db2sqljjdbc.properties 131
 - DB2SQLJPROPERTIES 128
 - LD_LIBRARY_PATH 128
 - LIBPATH 128
 - STEPLIB 127
 - PATH 128
 - software prerequisites 124
- Java Naming and Directory Interface (JNDI) 6
- JavaGateway class 188
- JCICS, link() method 188
- JNDI
 - tracing 193

L

- LDAP 5
- Lightweight Directory Access Protocol (LDAP) 6

M

- MAXPROCSYS 290
- MAXPROCUSER 290
- MAXPTYS 290
- MAXUIDS 290
- monplex 7
- MQSeries classes for Java
 - downloading 180
 - installation 179

- environment variables 180
- IVP 181
 - overview 179
 - software prerequisites 178
- MQSeries for OS/390 174
 - Distributed Queue Management (DQM) 176
 - MQSeries-CICS bridge 175
 - MQSeries-IMS bridge 176

N

- Network File System (NFS) 5

O

- Object Management Group (OMG) 6
- Object Transaction Service (OTS) 2

R

- RACF
 - CBIND class 4
 - DSNR class 4
 - LOGSTREAM class 4
 - SERVER class 5
 - STARTED class 4
- Recoverable Resource Services Attach Facility (RRSAF) 133
- Redbooks Web site 424
 - Contact us xv
- Resource Access Control Facility (RACF) 4
- resource adapter 189
- Resource Managers 3
- Resource Recovery Services 2
- RRS 192
- RRS Attachment Facility (RRSAF) 2

S

- SecureWay Security Server 6
- servlets 186
- shared HFS 5
- SMS (Systems Management Server) 6
- Structured Query Language Java (SQLJ)
 - differences with JDBC 121
 - executing the sample SQLJ application program 152
- Sysplex Distributor 8
- System Authorization Facility (SAF) 4
- System Logger 3

T

TCP/IP 6

- FTPD (File Transfer Protocol daemon) 349
- logging in to OS/390, overview 350
- rexecd (remote exec daemon) 349
- rlogind (remote login daemon) 348
- rshd (remote shell daemon) 348
- syslogd (syslog daemon) 349
- telnetd daemon 348

TCP/IP configuration

- /etc/hosts file 334
- /etc/services file, copying the sample 333
- adding the new host to your DNS server 347
- BPXPRMxx PARMLIB updates 340
- CINET 340

- COMMNDxx PARMLIB updates 339

customizing

- other TCP/IP files 333
- PROFILE.TCPIP 329
- SYS1.PARMLIB members 338
- TCPIP.DATA 333

- EZAZSS1 procedure, sample 338

- IECIOS00 PARMLIB updates 339

- IEFSSNxx PARMLIB updates 338

- INET 340

- LPALSTxx PARMLIB updates 339

- multiple TCP/IP stacks definitions, in BPX-PRMxx 341

- PROFILE file, groups of configuration parameters 329

- PROFILE.TCPIP file

- BEGINVTAM statement 331

- PROGxx PARMLIB updates 339

- SCHED00 PARMLIB 339

security definitions 343

- defining a RACF STARTED class definition for EZAZSSI started task 343

- defining a RACF STARTED class definition for TCP/IP started task 343

- defining a RACF user ID for TCP/IP address space 343

- defining user ID for EZAZSSI procedure 343

- TCPIP VARY command 343

- single TCP/IP stack definitions, in BPXPRMxx 340

- TCPIP.DATA file

- DATASETPREFIX statement 333

- finding 328

- sample 333

- TCPIPJOBNAME statement 333

- testing the setup and connection 347

- values 329

- Terminal Notification Facility (TNF) 338

U

Unit Of Work (UOW) 2

UNIX System Services

- \$HOME/.profile 298

- /etc/profile 298

- BPXOINIT 287

- child processes, creating 297

- copying files 297

customization

- _BPX_SHAREAS environment variable 290

- adding LE data set to LNKST in PROGxx 294

- adding LE data set to LPALST in LPALSTxx 294

- adding LE data sets to APF list in PROGxx 294

- adding OS/390 UNIX ISPF data sets to TSO/E IDs 297

- allocating HFS data sets 304

- allocating other file systems 291

- ALLOCxx PARMLIB member 295

- BPXPRM00 PARMLIB member 290

- BPXPRMxx PARMLIB member 288

- BPXPRMxx, example 290

- caching of the OS/390 UNIX System Services UID and GID information 294

- COFVLFXxx PARMLIB member 294

- creating HFSs for users 304

- CTnBPXxx member, where specified 295

- CTnBPXxx PARMLIB member 295

- CTnBPXxx PARMLIB member, example 295

- CTRACE parameter 295

- customizing the Shell 297

- customizing the system set up 289

- defining /dev and /var file systems 292

- defining a Temporary File System (TFS) for /tmp 293

- environment variables 297

- file permission bits, changing 292

- file system definition in BPXPRMxx, example

- 289
- FILESYSTYPE 293
- forked address spaces behavior 295
- HFS data sets, making available 305
- IEADMR00 PARMLIB member 296
- IEASYSxx PARMLIB member 293
- LPALSTxx PARMLIB member 294
- maximum processors 290
- mounting file systems, at IPL 292
- number of address spaces per user 290
- number of concurrent processes per user 290
- number of daemons 290
- number of PTYs and RTYs per user 290
- other PARMLIB members 293
- preparing PROCLIB members 296
- PROCUSERMAX keyword, in OMVS segment 290
- PROGxx PARMLIB member 294
- rules of thumb 290
- SMFPRMxx PARMLIB member 296
- system resources per user 290
- using the default values in BPXPRM00 290
- customization tasks 285
- daemon 319
 - activating RACF program control 320
 - DAEMON authority 319
 - defining BPX.DAEMON FACILITY class 319
 - defining BPXROOT user ID 320
 - start options 322
 - supplied by UNIX 319
 - with user ID other than kernel 322
- defining BPXOINIT cataloged procedure 287
- defining OMVS cataloged procedure 287
- kernel services
 - full-function mode setup 283
- OMVS started procedure 287
- security
 - assigning a user ID with a UID of 0 309
 - BPX.DEFAULT.USER FACILITY class profile 316
 - BPXISEC1 308
 - cataloged procedures for superusers 287
 - creating a RACF profile for HFS data sets 288
 - creating a RACF profile for the Root HFS 288
 - defining a Superuser 308

- defining BPX.SUPERUSER profile 309
- defining regular OS/390 UNIX System Services users 314
- defining the kernel group 286
- file access 307
- file security packet 307
- Group Identifier (GID) 305
- HFS 306
- minimizing users with superuser authority 310
- permitting a user for BPX.SUPERUSER 309
- protecting HFS data sets 288
- RACF OMVS segments 306
- RACF UID/GID values 312
- refreshing FACILITY class in memory 309
- setting up a default OMVS segment 314
- UID/GID value restriction with pax and tar 312
- UID/GID values 306
- UID/GID, where specified 306
- User Identifier (UID) 305
- using BPX.SUPERUSER FACILITY class profile 309
- verifying SYSPROG user ID in OMVS definitions 313
- setting up DFSMS/MVS 285
- UNIX Systems Services
 - Temporary File System (TFS) 293
- User Identifier (UID) 305

V

- Virtual Machine Communication Facility (VMCF) 338
- VSAM linear data set 3

W

- WLM
 - Application Environments 3
 - goal mode 4
 - WLM services 3
- Workload Manager (WLM) 3
- WTRSTART 296

X

- X.500 6



e-business Cookbook for z/OS Volume II: Infrastructure

(0.5" spine)
0.475" <-> 0.875"
250 <-> 459 pages



Redbooks

e-business Cookbook for z/OS Volume II: Infrastructure

How to set up WebSphere on z/OS

Configuration and use of e-business connectors

WebSphere setup in a sysplex

This IBM Redbook is the second volume of a series entitled the e-business Cookbook for z/OS. It is intended for System Programmers involved in setting up and configuring an environment to run Java-based e-business applications on z/OS and OS/390.

We focus on the most popular and viable application topologies. Software products covered are z/OS UNIX System Services, TCP/IP, IBM HTTP Server for z/OS, IBM WebSphere Application Server for z/OS V4, and the Java connectors for DB2, CICS, MQSeries and IMS. We also detail how to set up and run WebSphere in a Parallel Sysplex environment.

The information in this book will help you configure an environment on z/OS that will be ready to implement end-to-end e-business solutions using Java, WebSphere and traditional back-end systems. We also provide information on security aspects and performance optimization regarding many of the products.

The other volumes in this series are *e-business Cookbook for z/OS Volume I: Technology Introduction*, SG24-5664, and *e-business Cookbook Volume III: Java Development*, SG24-5980.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks