An Automation and High Availability Solution for WebSphere Application Server for OS/390 and z/OS Based on System Automation for OS/390

Contents

Chapter 1. Introduction 6
  1.1 WebSphere Application Server: a High-level Overview 6
  1.2 System Automation for OS/390: a High-level Overview 8
  1.3 WebSphere Application Automation and High Availability 8
Chapter 2. The Environment for WebSphere Application Server for OS/390 and z/OS 11
  2.1 Software products 11
    2.1.1 Prerequisite Products 11
      Mandatory Requisites 11
      Functional Requisites 12
  2.2 Basic Dependencies 12
    2.2.1 How WebSphere Application Server Uses z/OS Components 13
      z/OS Unix Systems Services (USS) 13
      Workload Manager (WLM) 13
      RACF 14
      LDAP 14
      DB2 15
      Resource Recovery Services (RRS) 15
      TCP/IP 15
      HTTP Server 17
    2.2.2 The Structure of WebSphere Application Server for OS/390 and z/OS 17
      The Base Environment 18
      Application Server Environment 19
  2.3 WebSphere Application Server in a Sysplex Environment 23
  2.4 Options for WebSphere Application Server for OS/390 and z/OS 24
    2.4.1 HTTP Sessions 24
      Session Affinity 24
      Persistent Sessions 24
    2.4.2 Using the IBM HTTP Server for z/OS 25
    2.4.3 Using an HTTP Server for Distributed Platforms 26
Chapter 3. High Availability for WebSphere Application Server for OS/390 and z/OS 27
  3.1 WebSphere Application Server for OS/390 and z/OS Administration 27
  3.2 WebSphere Application Server for OS/390 and z/OS Operations 28
  3.3 Changing Configurations for WebSphere Application Server for OS/390 and z/OS 28
    3.3.1 Starting, Stopping, Monitoring Address Spaces 29
      Using Identifiers, not Jobname 29
      Using Jobname, no Identifiers 30
    3.3.2 Conversation Activation and System Automation 30
      Disable Automation and let Activation Restart Servers 31
      Stop and Restart Servers using System Automation for OS/390 32
Chapter 4. System Setup 33
  4.1 Sysplex 35
  4.2 TCP/IP 35
  4.3 Base WebSphere Application Server for OS/390 and z/OS Setup 36
    4.3.1 TCP Related Definitions 36

4.3.2 HTTP Session Related Definitions 37
4.4 HTTP server 37
Chapter 5. Setup for System Automation for OS/390 39
5.1 Sysplex Group 41
5.2 KEYA, KEYB, KEYC 42
5.3 Application Groups 43
5.3.1 BBO_PLEX 43
5.3.2 BBO_DAEMON 44
5.3.3 BBO_J2EE 45
5.3.4 BBO_ASR2 46
5.3.5 BBO_BANK 47
5.3.6 BBO_LDAP 47
5.3.7 WWW_WEBSRV 47
5.4 Applications 48
5.4.1 BBO_CLASS 48
5.4.2 BBODMN 49
5.4.3 BBOSMS 51
5.4.4 BBONM and BBOIR 51
5.4.5 BBOWTR 52
5.4.6 BBOASR2 53
5.4.7 BBOBANK 54
5.4.8 BBOLDAP and WEBSRV 54
5.5 Relationships 54
5.5.1 LDAP 55
5.5.2 The IBM HTTP Server 56
5.5.3 The WebSphere Base Environment 56
5.5.4 J2EE Servers 57
5.6 Message Automation Table 58
5.7 Clean-up Program 58
5.8 WebSphere Maintenance 59
5.8.1 Activation using the SM Scripting API 60
Tools Samples 60

Figures

Figure 1: The WebSphere software Platform 7
Figure 2: The WebSphere Application Server for OS/390 and z/OS Environment 13
Figure 3: WebSphere Application Server for OS/390 and z/OS Environment 18
Figure 4: Components of a J2EE Server 20
Figure 5: WebSphere Application Server V4.0.1 for z/OS and OS/390 Plug-in Environment 21
Figure 6: HTTP Transport Handler 22
Figure 7: WebSphere in a Sysplex 23
Figure 8: Persistent Sessions with IBM HTTP Server 25
Figure 9: Session Affinity with a distributed HTTP Server 26
Figure 10: WebSphere on a Sysplex: Sample Configuration 34
Figure 11: Groups and Applications in a Sysplex Environment 41
Figure 12: BBO_PLEX WebSphere Base Environment 43
Figure 13: BBO_J2EE Group for J2EE Servers 45
Figure 14: Relationships 55

Chapter 1. Introduction

Running WebSphere Application Server for OS/390 and z/OS on a z/OS single system or in a sysplex environment increases the complexity of operating such a system. This is because WebSphere has a number of prerequisite products that it interfaces with, and also uses a number of z/OS address spaces itself to deliver high performance Web services.

When you run Internet applications on a system, its availability is exposed to the public. It is therefore essential to provide a scalable and highly available setup for operations providing Internet services.

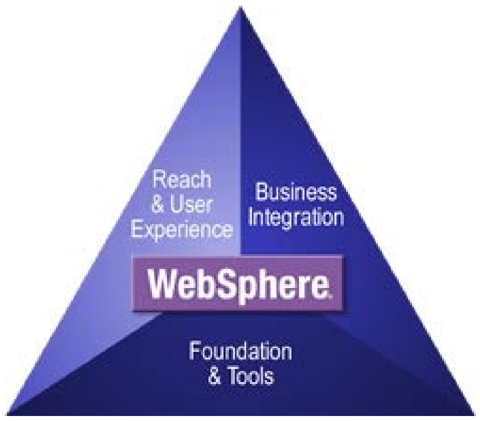
This paper presents a system automation and high availability solution. This provides a sample of how to set up a fully automated operational environment for all WebSphere Application Server V4.0.1 for z/OS and OS/390 components and prerequisites and related products running on a z/OS or OS/390 environment using System Automation for OS/390 V2.2 (SA OS/390). IBM supports this set-up by

- providing this white paper as a guideline
- supplying a sample Policy Database (PDB) over the Internet that contains the SA OS/390 definitions used to set up the environment described in the white paper. You can use this PDB as a sample to set up SA OS/390 for your own WebSphere Application Server for OS/390 and z/OS installation.
- enhancing the message table by adding messages issued by WebSphere and related products during the start up and termination processing through APAR OA02375
- shipping a common routine that can be used to clean up server address spaces that WebSphere was unable to terminate during its end processing through the same APAR

1.1 WebSphere Application Server: a High-level Overview

WebSphere is Internet infrastructure software - known as middleware. It enables companies to develop, deploy and integrate next-generation e-business applications.

Figure 1 shows the WebSphere software platform.



- Foundation and Tools
  - WebSphere Application Server
  - WebSphere Application Server - Express
  - WebSphere Studio
- Reach and User Experience
  - WebSphere Commerce
  - WebSphere Portal
  - WebSphere Portal - Express
  - WebSphere Everyplace
  - WebSphere Personalization
- Business Integration
  - WebSphere MQ
  - WebSphere Business Integration
  - IBM CrossWorlds
  - Adapters and Connectors

Figure 1: The WebSphere software Platform

The foundation of the platform is the IBM WebSphere Application Server, that provides specialized configurations designed to meet your most critical business needs.

WebSphere Application Server provides a rich, e-business application deployment environment with a complete set of application services. These include capabilities for transaction management, security, clustering, performance, availability, connectivity and scalability.

WebSphere Application Server is a Java-based Web application server, built on open standards, that helps you deploy and manage Web applications. It is J2EE-compliant and provides a portable Web deployment platform for Java components, XML and Web services, that can interact with databases and provide dynamic Web content.

It on workstation platforms, including Windows®, Linux, AIX®, UNIX and z/OS operating systems.

IBM WebSphere® Application Server V4.0.1 for z/OS® and OS/390® extends the role of S/390 and zSeries as premier servers for enterprise e-business -- offering superior qualities of service, scalability, performance, security, and availability. It offers:

- An EJB production environment with Java™ 2 Enterprise Edition (J2EE™) compliance
- Java Messaging Service (JMS), JavaMail, and Client Container
- Web services equivalent to those in the Version 4 distributed family of servers
- Web services with SOAP and UDDI open standards support

- Servlets, JavaServer Pages™ (JSPs), and Enterprise JavaBeans™ (EJBs) in compliance with J2EE specifications

For more details on WebSphere please refer to
http://www7b.boulder.ibm.com/wsdd/zones/newcomers/

1.2 System Automation for OS/390: a High-level Overview

System Automation for OS/390 plays an important role in building the end-to-end automation of the IBM autonomic computing initiative. The unique functions of SA OS/390 V2.2 can help customers with single z/OS or OS/390 systems and Parallel Sysplex clusters to ease management, may reduce costs, and increase availability. SA OS/390 is designed to automate I/O, processor, and system operations and includes "canned" automation for IMS, CICS, IBM Tivoli Workload Scheduler, and DB2.

Working examples are provided for:

- SAP R/3 High-Availability Automation
- High Availability Solution for WebSphere Application Server for z/OS and OS/390.

For Parallel Sysplex application automation SA OS/390 V2 introduced extremely powerful new concepts, including sysplex-wide grouping of resources, relationships, and goals that make Parallel Sysplex automation a reality. You can manage any number of stand-alone systems and clusters from a single point-of-control, even from an easy-to-use Java-based GUI. In addition, you can move applications and related timers inside a sysplex.

SA OS/390, a NetView application, integrates with Tivoli enterprise solutions such as Tivoli Enterprise Console (TEC) and Tivoli Business Systems Manager (TBSM). TBSM can act as status observer for SA-monitored resources.

SA OS/390 primarily deals with starting and stopping applications in accordance with their inter-relationships. These include relationships of applications to other applications, or being a component application of an application complex. SA OS/390 also supports the permanent availability of an application by moving it to another system if there is an unrecoverable failure.

All applications and systems that you may want to include in automation must be defined to SA OS/390 in an automation Policy Database. This database contains the objects to be managed by SA OS/390, and the rules according to that automation of these objects proceeds.

For more details on System Automation for OS/390 please refer to
http://www-1.ibm.com/servers/eserver/zseries/software/sa/

1.3 WebSphere Application Automation and High Availability

Running WebSphere Application Server for OS/390 and z/OS on a z/OS single system or in a sysplex environment increases the complexity of operating such a system. This is because WebSphere has a number of prerequisite products that it interfaces with, and also uses a number of z/OS address spaces itself to deliver high performance Web services.

When you run Internet applications on a system, its availability is exposed to the public. It is therefore essential to provide a highly available setup for operations of this type. The aspects of high availability in general include:

- continuous operation: this covers the ability to avoid planned outages
  - enable administrative work, and maintenance of hardware and software to be done while the application remains available to end users
  - normally accomplished by providing multiple servers, and switching end users to an available server when one server is unavailable
- high availability: this is the ability to avoid unplanned outages by eliminating single points of failure
- the ability to minimize the effect of an unplanned outage by masking the outage from end users
- continuous availability: this combines the characteristics of high availability and continuous operation to get as close as possible to 24x7x365.

In a monoplex environment your options to react to outages are limited because you do not have spare images that you could use for back up in case of failures. But you can still use SA OS/390 to automate an environment with WebSphere Application Server for OS/390 and z/OS focusing on:

- ease of operations support:
  - start all prerequisites and components of WebSphere in the right order
  - stop all components of WebSphere in the right order
  - show status of all prerequisites and components
- availability support reducing downtimes:
  - monitor prerequisites and components
  - automatically restart failing components.

In addition to the monoplex advantages, SA OS/390 can increase availability in a sysplex environment by

- ensuring that WebSphere Application Servers run on a predefined number of images in the sysplex
- moving all or parts of WebSphere Application Server for OS/390 and z/OS from a failing image to another image when restart of vital component fails
- cleaning up failed system.

It is currently necessary to shut down and restart a J2EE server when you change its configuration. WebSphere Application Server will automatically shut down and restart all running J2EE servers with a changed configuration when the conversation containing these changes is activated.

This prevents a set up that fully supports continuous operations even if you run in a sysplex environment.

With SA OS/390 enabled for WebSphere Application Server you can minimize downtimes. It offers facilities to allow that maintenance for WebSphere Application Server for OS/390 and z/OS can safely be performed by terminating WebSphere Application servers and making them available again as quickly as possible. This is discussed in more detail in 2.1 “Conversation Activation and System Automation” on page 30.

Chapter 2. The Environment for WebSphere Application Server for OS/390 and z/OS

In this chapter we give the software prerequisites for WebSphere Application Server for OS/390 and z/OS. We also give an overview of the dependencies between these z/OS components and products, and the WebSphere Application Server. We tell you how the WebSphere Application Server uses these components. The parts that make up the WebSphere Application Server for OS/390 and z/OS itself are shown, and their functions are explained.

2.1 Software products

Base products for WebSphere Application Server for OS/390 and z/OS automation are, of course, WebSphere Application Server for OS/390 and z/OS V4.0.1 and System Automation for OS/390 V2.2. In section 2.1.1 Mandatory Requisites, we list the products that are required to run the base products. In 0 Functional Requisites on page 6 we list products that are not strictly required but that we used in the set-up we describe in this paper.

2.1.1 **Prerequisite Products**

A mandatory requisite is defined as a product that is required without exception; the presented solution **will not install** or **will not function** unless these requisites are met.
A functional requisite is defined as a product that is **not** required for the successful installation of this solution, but **is** needed at run time for a specific function to work.
Note: Some PTFs have to be applied to different software components. For details please check the Program Directories of WebSphere Application Server V4.0.1 for z/OS and OS/390 and Tivoli System Automation for OS/390 Version 2 Release 2.

Mandatory Requisites

The following products are required:

- WebSphere Application Server V4.0.1 for z/OS and OS/390
- Tivoli System Automation for OS/390 Version 2 Release 2 (with APAR OA02375 for WebSphere Automation enhancements)

and their prerequisite operating system:

- OS/390 Version 2 Release 10

or

- z/OS Version 1 Release 1 or higher

In addition the following products or features are required:

- Tivoli NetView Release 4
- OS/390 R10 Security Server or z/OS R1 Secure Way Security Server
- DB2 Version 7
- IBM Developer Kit for OS/390 Java 2 Technology Edition

Functional Requisites

The list includes prerequisites for System Automation for OS/390 and for WebSphere Application Server for OS/390 and z/OS.

- IBM HTTP Server V5.3
- OS/390 V2R8 (or higher) SecureWay Communications Server IP or a functionally equivalent product for Java networking classes

2.2 Basic Dependencies

Figure 2: The WebSphere Application Server for OS/390 and z/OS Environment" on page 13 shows the main software components required to run a WebSphere Application Server for OS/390 and z/OS environment.

It also shows the sequence in that these components parts should be started in a z/OS image.

The use of the following z/OS components by the WebSphere Application Server provides a stable platform, with built-in recovery and security:

- TCP/IP as the communications vehicle
- Unix Systems Services (USS) platform support with HFS for file support
- LDAP and DB2 for directory services and database support
- WLM for work load management
- RACF for security
- RRS for recovery

Note that WebSphere Application Server for OS/390 and z/OS consists itself of a base environment that provides the infrastructure for the J2EE and MOFW servers that control and run J2EE web applications or CORBA applications.

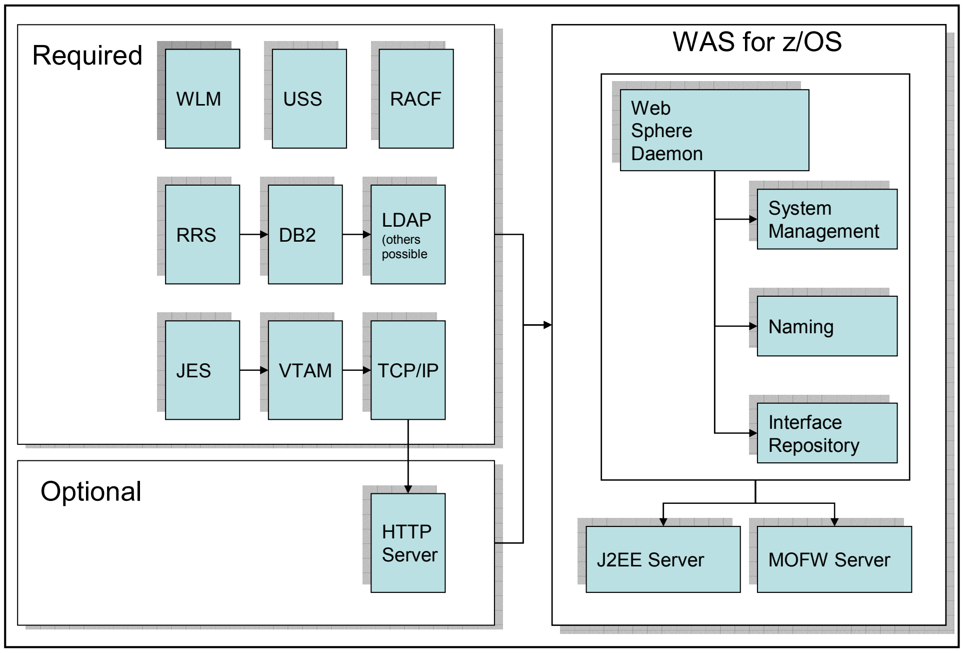


Figure 2: The WebSphere Application Server for OS/390 and z/OS Environment

2.2.1 How WebSphere Application Server Uses z/OS Components

This section provides some details of how the WebSphere Application Server for OS/390 and z/OS uses functions offered by z/OS components.

z/OS Unix Systems Services (USS)

z/OS Unix Systems Services provides services needed to run the WebSphere Application Server environment and the Java applications that run within it. It also provides the Hierarchical File System used by the WebSphere Application Server.

Hierarchical File System (HFS)

HFS is used to store Java code, static files, and the configuration files needed to run the application servers. In a sysplex configuration all the systems that run WebSphere need shared access to the WebSphere configuration HFS. The recommendation is to use shared HFS.

Workload Manager (WLM)

The Workload Manager functionality is provided by the z/OS operating system. It gives increased flexibility to WebSphere when running on the zSeries platform.

With workload management, you define performance goals and assign a business importance to each goal. The system decides how much resource, such as CPU and storage, should be allocated to meet the goal. Workload Manager will constantly monitor the system and adapt processing to meet the goals that you set.

WebSphere requires that WLM is running in goal mode and that Application Environments are defined for its servers. To balance work on the system WebSphere uses the following services of the Workload Manager:

- **Queuing and address space management:** to dispatch work requests from the WebSphere Application Server for z/OS Control Region to one or more WebSphere Application Server for z/OS
- **Prioritizing work to meet performance goals:** WebSphere delegates the responsibility for starting and stopping Server Regions to the WLM.

In addition we used WLM in connection with the Sysplex Distributor.

- **Providing utilization information:** to enable the Sysplex Distributor to route incoming service requests to the most suitable system.

RACF

WebSphere security is based on RACF (or any other external security server implementing the SAF interface).

All entities or principals, internally or externally, have to be authenticated implicitly or explicitly. This includes Control and Server Regions running under WebSphere. Therefore a number of IDs have to be defined in RACF and many authorizations in several RACF classes have to be granted (such as 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 etc.).

For WebSphere running 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.

LDAP

The z/OS Lightweight Directory Access Protocol (LDAP) server is part of the SecureWay Security Server for z/OS.

LDAP directory services provide an easy way to store, update, and retrieve directory information in a central location for the Java Naming and Directory Interface (JNDI) and CORBA naming and interface repository services. The contents of the directory are stored in a DB2 table.

LDAP is used by both the “system servers” -- Systems Management Server (SMS) and Interface Repository (IR) -- and the J2EE servers, but in different ways. SMS and 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 Java Naming Directory Interface (JNDI).

DB2

WebSphere V4 requires the use of DB2 UDB for OS/390 and z/OS V7.1 or later for the following purposes:

- WebSphere configuration and session control information is held in DB2 databases. The WebSphere control information consists of about 60 tables in two databases: BBOMDB01, BBOJDB01
- DB2 is the default database for the LDAP server
- DB2 is used to store persistent session data

In addition DB2 user data can be accessed by Java applications through JDBC.

For WebSphere running in a sysplex you should use a shared DB2 to make the data consistent across the sysplex.

Resource Recovery Services (RRS)

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

For CORBA applications the Object Transaction Service (OTS) manages transactions and provides the interface to z/OS RRS, that coordinates resource recovery across several Resource Managers. OTS requires RRS.

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.

TCP/IP

TCP/IP is part of the z/OS Communications Server.

The TCP/IP protocol is used as the basis for communication between the clients and the HTTP server. It is also used to communicate between other portions of the WebSphere Application Server environment, such the System Management Server and the System Management User Interface.

We make use of the Virtual IP Addressing (VIPA) and Sysplex Distributor functionality offered by TCP/IP to achieve high availability and to utilize z/OS workload balancing.

Virtual IP Addressing

With VIPA, you define a virtual IP address that does not correspond to any physical attachment or interface. Communication Server for z/OS IP then takes care that the VIPA address appears to the IP network as if it was a separate sub-network. A client selecting the VIPA address to contact its server will have packets routed to the VIPA via any one of the available real host interfaces.

Communication Server extends the VIPA concept to allow for the recovery of failed system images or entire TCP/IP stacks:

- The automatic VIPA takeover function allows you to define the same VIPA address on multiple TCP/IP stacks in a sysplex. One stack is defined as the primary or owning stack and the others are defined as secondary or backup stacks for the VIPA. Only the primary one is made known to the IP network. If the owning stack fails, then one of the secondary stacks takes its place and assumes ownership of the VIPA In this case, applications associated with these DVIPAs are active on the backup systems, thereby providing a *hot standby* for the services.
- Dynamic VIPA (for an application instance) allows an application to register to the TCP/IP stack with its own VIPA address. This lets the application server move around the sysplex images without affecting the clients. In this way, the application can dynamically activate the VIPA on the system image it wishes to host the application. Because the application instance is only active on one image in the sysplex at a time, the other images provide a *cold standby* of the service.

These VIPA enhancements are enabled by the use of XCF to communicate between the TCP/IP stacks.

Sysplex Distributor

Sysplex Distributor is the strategic IBM solution for connection workload balancing in the S/390 or zSeries sysplex.

The concept is that a cluster of server instances is represented by a single IP address. Because Sysplex Distributor is built on Dynamic VIPAs, this is called a Distributed VIPA (DVIPA). Basically, it is defined on a primary TCP/IP stack and on backup stacks as appropriate. In addition, the primary stack contains a configuration statement that identifies the DVIPA as Distributed, the ports that an application will use (between one and four ports per Distributed DVIPA), and the target application hosting TCP/IP stacks in the sysplex.

“All present and future stacks” may be specified as the target application hosting stacks, or targets may be limited to selected specific TCP/IP stacks. This information is distributed automatically to all candidate target TCP/IP stacks, and, on all candidate target stacks, the same IP address is activated as a “hidden” DVIPA. It is hidden in the sense that the target-application hosting stacks do not advertise its presence via their dynamic routing daemons, so its presence on those stacks is not known to the outside world.

The Sysplex Distributor routing stack advertises ownership of the DVIPA to the world, and waits for connection requests to be sent to it.

The routing stack will not send a TCP connection to a target application host until there is an actual application ready to receive work. Because the application-hosting target stacks are aware of the port numbers associated with a Distributed DVIPA, the target stack notifies the routing stack when an application is bound to one of these ports, and accepts connections to the Distributed DVIPA. The routing stack will send future work (TCP connections) for that application (port) to that target stack. If the server should close its listening socket, or end abnormally, the hosting target stack is immediately aware of this, and notifies the routing stack.

Should a target stack suffer an outage, connections to any server instances on that target stack are of course lost, but other server instances are still available for the affected clients when they attempt to reconnect to the server via the Distributed DVIPA. The routing stack will immediately send a connection reset and the reconnection request will go to one of the remaining functioning instances. New connection requests will not be routed to a failed target stack or LPAR.

To cover the case of a failure of the routing stack itself the Distributed DVIPA should be configured with backup routing stacks. If the routing stack should suffer an outage, the backup routing stack will take over global responsibility for the Distributed DVIPA, and each target stack will be aware of the takeover. This is all handled by the collaborating Sysplex Distributor TCP/IP stacks, and client and server applications themselves are entirely unaware of the failure of the former routing stack.

HTTP Server

The HTTP Server is primarily used to serve static pages. The HTTP Server allows you to provide simple Web serving and an e-business presence on the Internet, but the technology is limited.

The HTTP Server also provides an environment for the WebSphere Application Server Plugin. This plugin environment expands your ability to do business on the Internet. For details see the discussion of the WebSphere Plugin environment on page 21.

2.2.2 The Structure of WebSphere Application Server for OS/390 and z/OS

In this section, we look first at WebSphere Application Server for OS/390 and z/OS in a monoplex environment. For sysplex considerations see section 2.3 “WebSphere Application Server in a Sysplex Environment”on page 23.

The WebSphere Application Server for OS/390 and z/OS environment consists of a base environment that provides support for application servers, and the application servers themselves.

Figure 3 shows the structure of a WebSphere Application Server for OS/390 and z/OS environment:

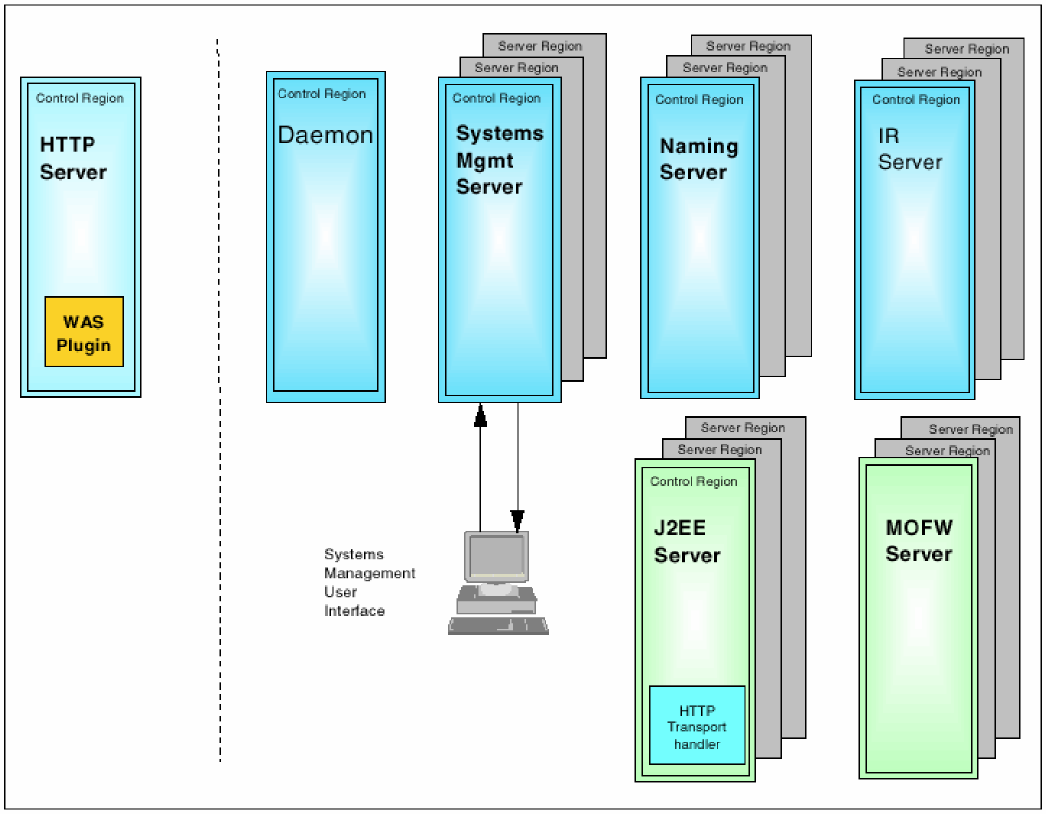


Figure 3: WebSphere Application Server for OS/390 and z/OS Environment

Each of the subcomponents in the above figure is a separate address space.

The Base Environment

The base environment consists of the Daemon, Systems Management (SMS), Naming and Interface Repository (IR) servers.

The Daemon Server

The Daemon brings up the WebSphere Application Server environment. It exists as a control region only within a single z/OS address space. It is started via a procedure. It starts the other control regions of the base environment (Systems Management, Naming and IR).

Systems Management Server

The purpose of the SMS is to allow users to add new applications and control the configuration of the application server environment. This is accomplished by connecting a Systems Management Extended User Interface (SMEUI) console to the SMS server via a TCP/IP connection.

The SMS consists of a Control Region and multiple Server Regions, each in a separate address space that can be started by WLM if needed.

Naming Server

The Naming server is used to locate Java objects, such as EJBs, and works in conjunction with the IR server. This provides the operational support for both the J2EE and CORBA (MOFW) servers. It allows applications, EJBs, and CORBA Business Objects to locate and communicate with each other.

Like the SMS, the Naming Server consists of a control region and multiple server regions that can be started by WLM if needed.

Interface Repository Server

The IR server manages the inventory of CORBA business object interfaces. The IR server also consists of a control region and multiple server regions. It is required to run on the bootstrap system in a sysplex (the first system where you install WebSphere).

Systems Management Extended User Interface (SMEUI)

During the installation of the WebSphere Application Server Environment you also install the SMEUI application on a PC. This graphical user interface allows you to view and update the current configuration of your application servers.

Application Server Environment

There are two types of application servers that run in this environment: the J2EE application server and the MOFW (CORBA) server.

J2EE application servers are based on the Java language only and allow you to run J2EE components, such as servlets or EJBs. The CORBA application server is used for serving CORBA Business Objects that can be written in a number of different languages for use on different platforms. Both of these technologies can talk to each other in a WebSphere Application Server for OS/390 and z/OS environment.

J2EE Server

The J2EE server of the WebSphere Application Server for z/OS and OS/390 provides a highly available, secure, reliable, and scalable run-time environment for Java 2 Enterprise Edition (J2EE) applications.

A J2EE server consists of one Control Region and one or more Server Regions. Additional Server Regions are started by WLM if required by the J2EE application workload.

The J2EE Server supports both Enterprise JavaBeans and Web components such as JSPs and Servlets that conform to the J2EE specifications and packaging standards published by Sun Microsystems.

These two types of J2EE application components run on a WebSphere for z/OS J2EE server, and can use

- the application programming interfaces (APIs) and services that the Java 2 Standard Edition (J2SE) Software Development Kit (SDK) V1.3 provides, and
- enterprise services such as Java Database Connectivity (JDBC), Java Naming and Directory Interface (JNDI), and the Java Transaction Service (JTS) and API (JTA).

The J2EE specifications dictate that APIs and services each type of application component may use, and the environment in that they must run. Although both Enterprise beans and Web applications may run in a single WebSphere for z/OS J2EE server, each component actually runs in a separate type of container within the J2EE server. Enterprise beans run in the EJB container, and Web applications run in a Web container. These two containers in the WebSphere for z/OS J2EE server conform to the J2EE specifications for run-time environments. Figure 4 below shows the components of a J2EE Server.

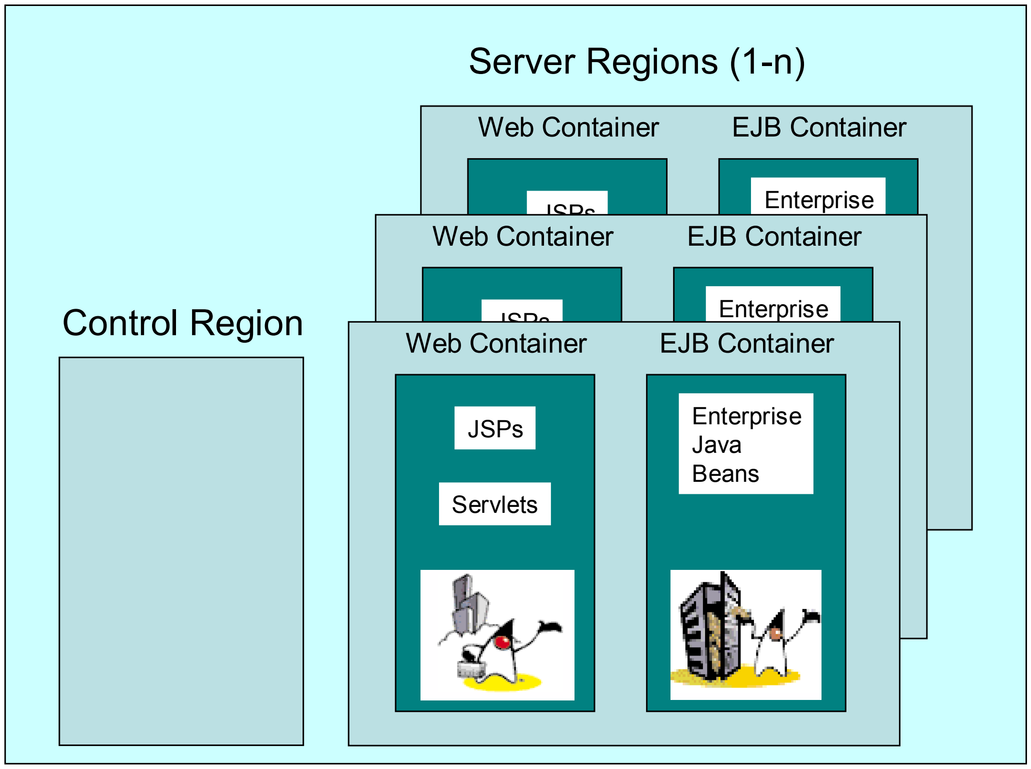


Figure 4: Components of a J2EE Server

MOFW (CORBA) Server

WebSphere for z/OS also provides a Managed Object Framework (MOFW) Server that provides a run-time environment for CORBA-compliant components.

We did not implement an MOFW Server in our installation and will therefore not discuss it further.

WebSphere Plugin environment

The WebSphere Plugin is used primarily to route servlet and JSP requests to the WebSphere Application Server. It is shipped WebSphere Application Server V4.0.1 for z/OS and OS/390, but it is code that runs within the IBM HTTP Server.

Starting with WebSphere 4.0.1 you have the choice of whether you want to run your servlets in the WebSphere Plugin (locally), or to run them remotely in a Web container residing in the same address space as your EJBs.

The ability to run servlets locally provides you with a means to migrate to WebSphere 4.0.1 using your existing environment with minimum impact. The ability to run servlets remotely in a Web container allows you to minimize the overhead when talking to EJBs. You can keep all the pieces of your application within a single z/OS address space. This is the only option we consider in the rest of this document.
Figure 5 shows the setup involved with this configuration.

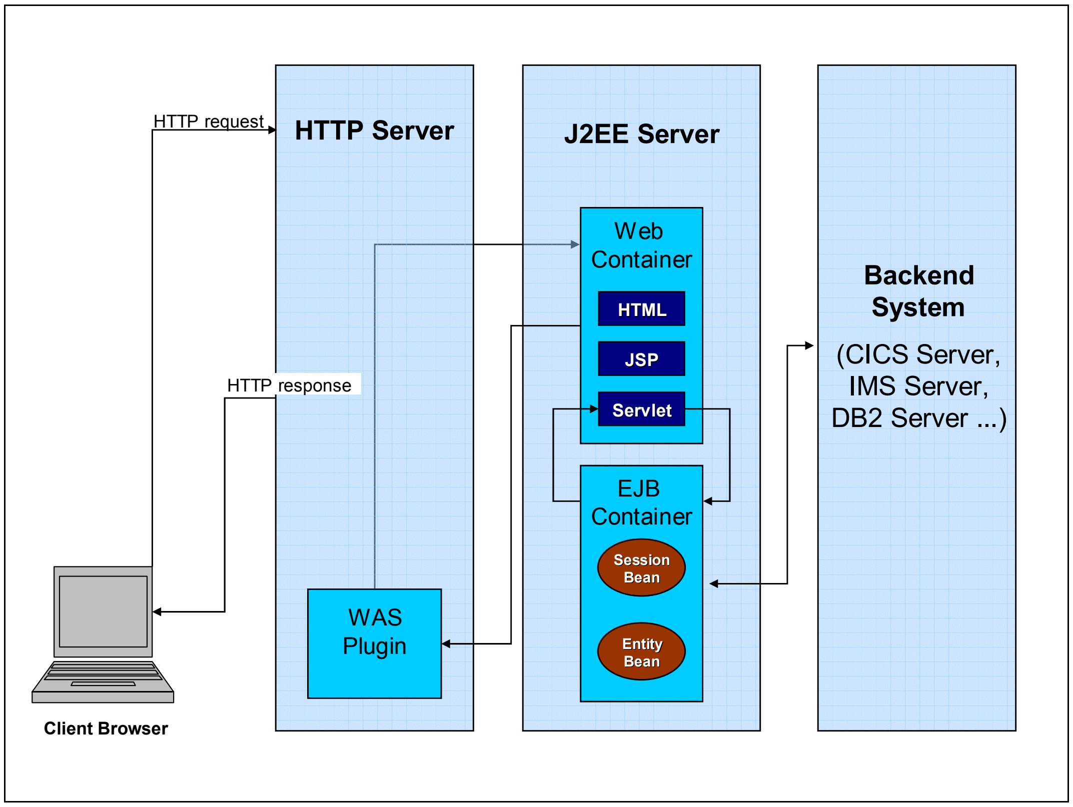


Figure 5: WebSphere Application Server V4.0.1 for z/OS and OS/390 Plug-in Environment

Based on the context root of an HTTP request coming from a browser the request is passed from the IBM HTTP server to the WebSphere Application Server V4 Plugin. The plugin routes the request to the J2EE server's Web container where static or dynamic content can be composed and passed back in HTML format to the HTTP server via the plugin. If the request invokes a servlet that in turn invokes EJBs these are running in the J2EE Servers EJB container.

HTTP Transport Handler

The HTTP Transport Handler runs as part of the J2EE Server's Control region. This component can handle HTTP(S) requests from clients.

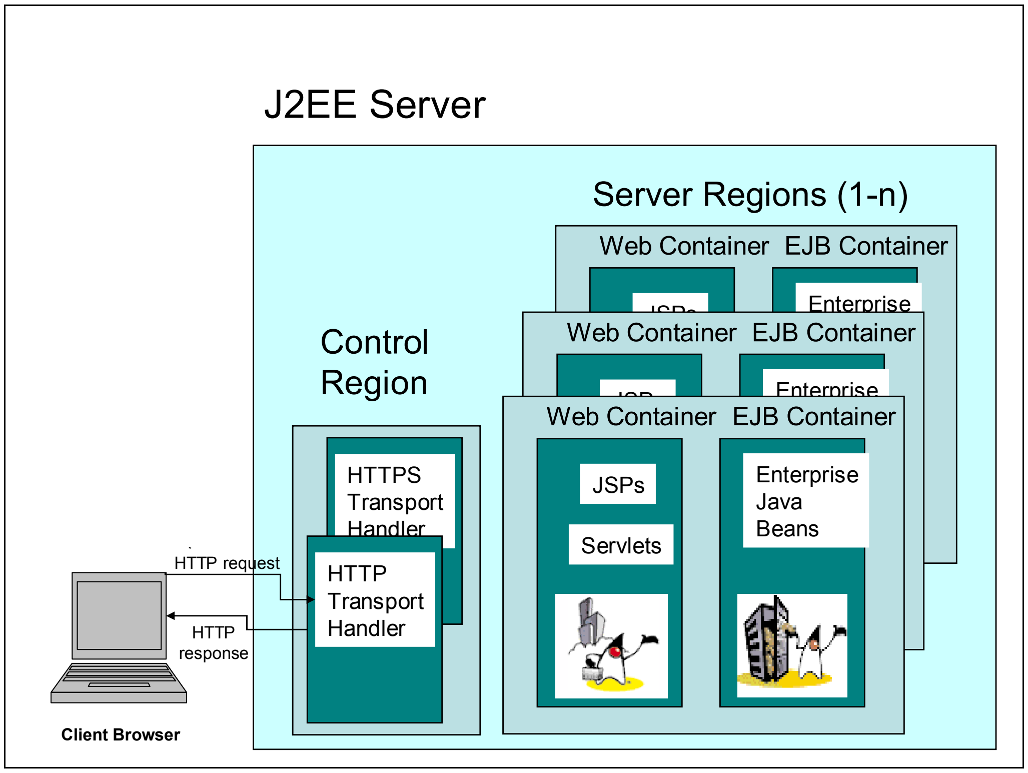


Figure 6: HTTP Transport Handler

The protocol handler can speed up HTTP processing significantly, although, because it is not designed to deal with browser clients, it cannot handle the differences between browsers. It assumes that there is an HTTP server in front of it. In our case we included the IBM HTTP Server for z/OS in our setup and therefore do not cover in the rest of this paper the case where the Transport Handler is used to communicate with client browsers directly or through non-z/OS HTTP servers.

2.3 WebSphere Application Server in a Sysplex Environment

In a sysplex environment some (but not necessarily all) z/OS images can contain WebSphere Application Server for OS/390 and z/OS components. On each individual image we speak of a server instance, a server is made up of all the server instances on the z/OS images. Thus we have a WebSphere Application Server on the sysplex made up of several WebSphere Application Server instances. The WebSphere Application Server consists of a Daemon Server with Daemon Server instances, a Naming Server with Naming Server instances, an IR server with IR Server instances. Similarly J2EE and MOFW servers are made up out of J2EE and MOFW server instances.

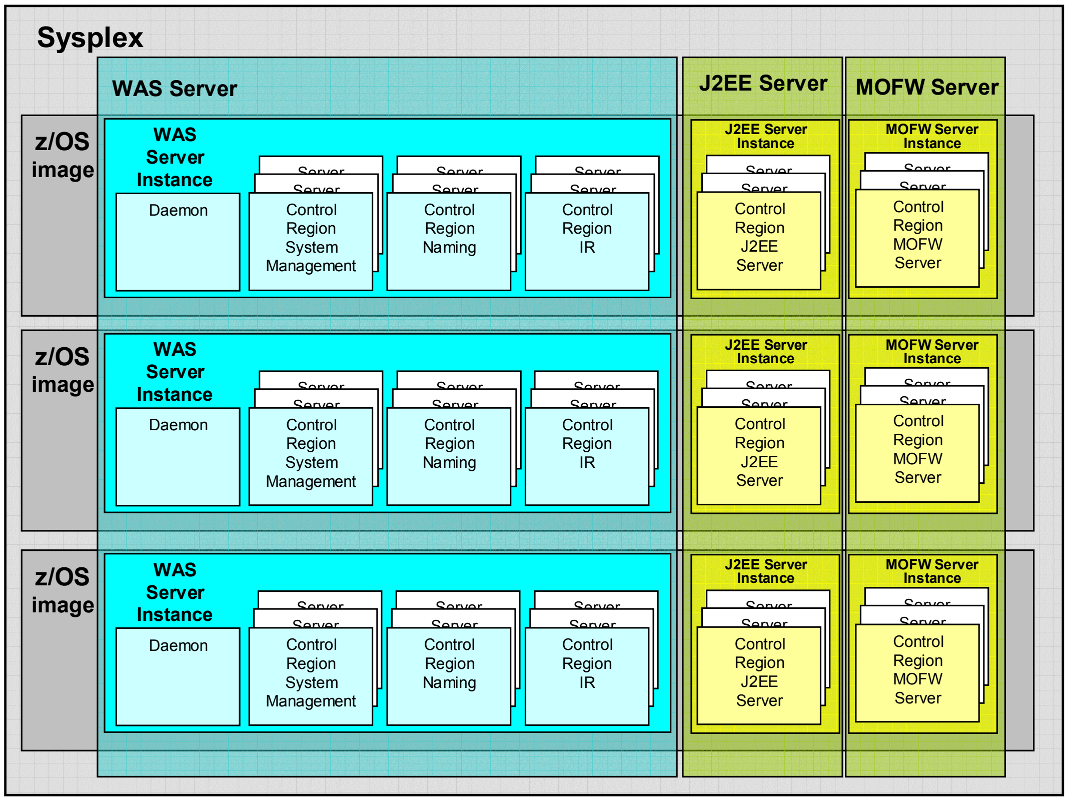


Figure 7: WebSphere in a Sysplex

A sysplex cluster is configured into the WebSphere for z/OS namespace as a host and is represented by a single Daemon IP Name. Because of this, systems and applications outside the sysplex treat the sysplex as a single host. Functions in WebSphere for z/OS route work through the sysplex according to the availability of server instances and to workload balancing rules, in cooperation with subsystems in z/OS or OS/390, such as TCP/IP, the domain name server (DNS), and workload management.

On each image on that a J2EE or MOFW server instance runs an instance of the Daemon and of System Management is required. Naming instances and IR instances are not strictly required on each image of a sysplex where you want to run J2EE or MOFW servers, but it is recommended to have them running on all images.

Of course multiple J2EE Servers or MOFW servers can run under the control of one WebSphere Application Server on a sysplex.

2.4 Options for WebSphere Application Server for OS/390 and z/OS

There are various options for setting up your environment for WebSphere Application Server for OS/390 and z/OS. One of the first choices is whether you include an HTTP server in your z/OS setup, or use an external HTTP server (on Linux, Unix or Windows platforms). Because this choice impacts how you can implement HTTP sessions, we will discuss the options you have for HTTP sessions first.

.

2.4.1 **HTTP Sessions**

When users dynamically collect data as they move through a Web application, the application needs a mechanism to hold the user's state information over a period of time. However, HTTP treats each user request as a discrete, independent interaction.

The Java servlet specification provides a mechanism, known as a session, for servlet applications to maintain state information. This mechanism allows a Web application developer to maintain all user state information at the host, while passing minimal information back to the user via cookies or URL rewrite. The state information associated with a series of client requests is represented as an HTTP session object and identified by a session ID. The session manager module that is part of each servlet engine is responsible for managing HTTP sessions, providing storage for session data, allocating session IDs, and tracking the session ID associated with each client request, through the use of cookies or URL rewriting techniques.

There are two ways that this can be implemented: session affinity or persistent sessions.

Session Affinity

With session affinity the session information is held just in the memory of the servlet engine that started the session.

Session affinity means that when an application starts a session the session information is stored in a session object that resides in the specific JVM where the session is started. All further requests belonging to this session also need to run in this specific JVM. You must ensure that all requests during this session reach this specific servlet engine.

Persistent Sessions

With persistent sessions, the session information is stored in a shared database, and each servlet engine has access to it.

Besides storing session objects in memory when you enable persistent session management WebSphere places session objects in a predefined database. All information stored in a persistent session database must be serialized. As a result, all the objects held by a session must implement java.io.Serializable in this case.

2.4.2 Using the IBM HTTP Server for z/OS

When you use the IBM HTTP server for z/OS, you also use the WebSphere Application Server for z/OS plugin to route requests from the HTTP server to the application server.

This has the advantage that you have the entire infrastructure required to run your Web applications on z/OS where you can control and automate it more easily. In addition the workload can be distributed across all z/OS images using the information available to WLM to choose the optimal system.

There is, however, one disadvantage: The WebSphere Application Server for z/OS plugin does not currently support session affinity. You therefore have to implement persistent sessions in this setup.

Figure 8 gives an overview of the flow of requests from browser to Web application in such a setup.

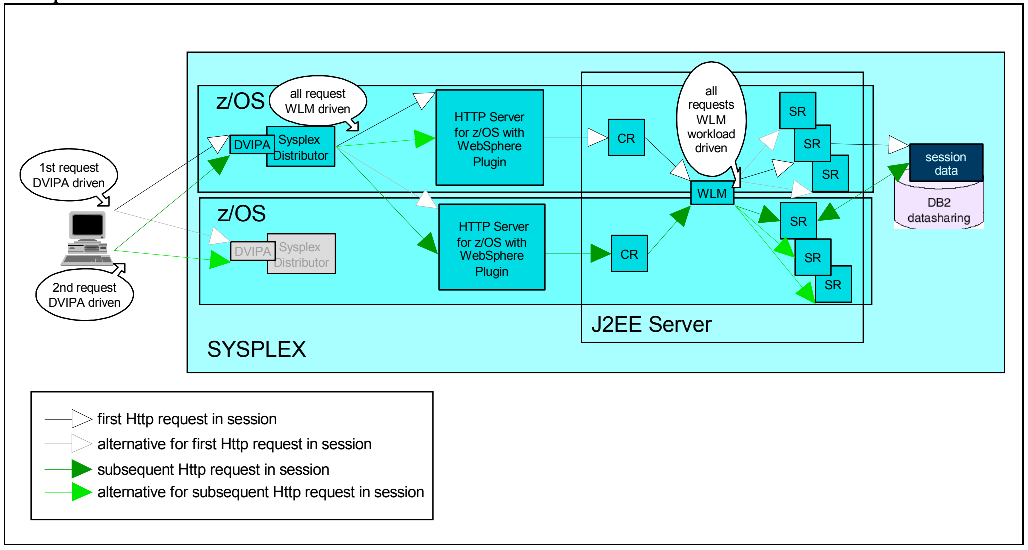

z/OS
all request WLM driven
DVIPA
Sysplex Distributor
HTTP Server for z/OS with WebSphere Plugin
CR
all requests WLM workload driven
SR
SR
SR
session data
DB2 datasharing
WLM
1st request DVIPA driven
2nd request DVIPA driven
z/OS
DVIPA
Sysplex Distributor
HTTP Server for z/OS with WebSphere Plugin
CR
SR
SR
SR
J2EE Server
SYSPLEX
first Http request in session
alternative for first Http request in session
subsequent Http request in session
alternative for subsequent Http request in session

Figure 8: Persistent Sessions with IBM HTTP Server

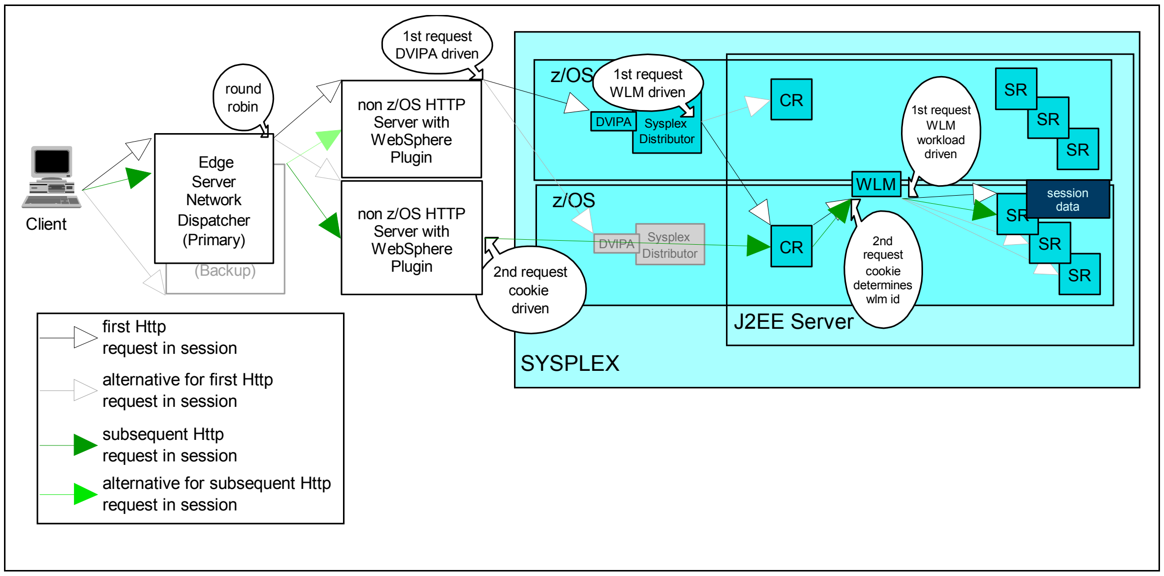
As you can see, subsequent requests can end up being executed in any server region belonging to the J2EE Server. Since session data is stored in a shared DB2 database, it is accessible from any J2EE server region. You can also see that the workload is distributed at a HTTP request and an Application Server instance level. In both cases we can make use of the information available to WLM about sysplex load and distribute accordingly.

2.4.3 **Using an HTTP Server for Distributed Platforms**

If you use IBM HTTP Server (IHS) based on the Apache HTTP server on distributed platforms, you can use the WebSphere Advanced Edition Web server plugin available for the respective distributed platform. This plugin can connect directly to the HTTP Transport Handler of an application server on WebSphere Application Server for z/OS.

The main value of using the WebSphere Advanced Edition Web server plugin is that it can honor session affinity to a specific application server control region on z/OS. Within the correct control region the WebSphere Application Server honors session affinity to route the request to the right server region.

Figure 9 shows the request flow when you implement session affinity.



Client
round robin
Edge Server Network Dispatcher (Primary)
(Backup)
1st request DVIPA driven
non z/OS HTTP Server with WebSphere Plugin
non z/OS HTTP Server with WebSphere Plugin
2nd request cookie driven
z/OS
1st request WLM driven
DVIPA
Sysplex Distributor
z/OS
DVIPA
Sysplex Distributor
CR
CR
WLM
1st request WLM workload driven
2nd request cookie determines wlm id
SR
SR
SR
SR
SR
SR
session data
J2EE Server
SYSPLEX
first Http request in session
alternative for first Http request in session
subsequent Http request in session
alternative for subsequent Http request in session

Figure 9: Session Affinity with a distributed HTTP Server

Normally it is not sufficient to move just the HTTP Server to a non-z/OS platform. To implement a high availability solution and enable scalability and workload distribution, you will also need a product such as the Network Dispatcher on the distributed platforms.

Note that the Network Dispatcher does not directly have access to information about the workload on your sysplex. It can only distribute to various HTTP Servers based only on HTTP workload. You should also note that workload distribution with the sysplex can only be done for the first request within an HTTP session. From then on the z/OS image with the sysplex and the J2EE server region must remain bound to the session, and can not be changed to better distribute system utilization.

Chapter 3. High Availability for WebSphere Application Server for OS/390 and z/OS

In this chapter we briefly discuss the basic high availability considerations for WebSphere Application Server for OS/390 and z/OS and the role System Automation for OS/390 can play to support it.

3.1 WebSphere Application Server for OS/390 and z/OS Administration

It is necessary to perform administration actions for a WebSphere Application Server for OS/390 and z/OS when you want to:

- add new application servers or resources
- change parameters for the existing environment, that is manage environment variables
- install new applications into an application server after the applications were assembled and deployed.

WebSphere Application Server for OS/390 and z/OS keeps the complete setup information of its configuration in so called conversations.

After initial installation and customization, WebSphere for z/OS manages environment data through the Administration application and writes the environmental data into the system management database. When you activate a conversation, the data is also written to HFS files.

To install an application on WebSphere for z/OS, you must also use the WebSphere for z/OS Administration application. During the application installation process, deployment descriptors are customized for the WebSphere for z/OS environment, and application components are loaded into the WebSphere for z/OS application server.

There are two interfaces to administer a WebSphere Application Server for OS/390 and z/OS:

1. The WebSphere for z/OS Administration application that runs on your Windows workstation and communicates with the System Management server, that is the 'System Management Enhanced User Interface' (SMEUI)
2. The SM Scripting API that provides the same functions as the SMEUI. REXX is currently the only script language that is supported by the SM Scripting API.

WebSphere Application Server for OS/390 and z/OS shuts down and restarts all J2EE servers with changed configuration when the conversation containing these changes is activated. The impact this has on availability and how this needs to be reflected with respect to System Automation for OS/390 is discussed in 3.3 "Changing Configurations for WebSphere Application Server for OS/390 and z/OS" on page 28.

3.2 WebSphere Application Server for OS/390 and z/OS Operations

Through WebSphere for z/OS operations you can manage WebSphere for z/OS servers and server instances. This includes actions to:

- display the status of all server instances
- stop application servers and server instances
- cancel application servers and server instances
- restart servers and server instances.

WebSphere for z/OS operations can be performed from:

- a z/OS or OS/390 MVS console
- the Systems Management Enhanced User Interface on Windows
- TSO or RRS panels (for some operations)

If you use SA OS/390 to automate WebSphere, it will perform these tasks for you. There is normally no need , for example, for you to start, stop or restart servers. When, in exceptional cases, you have to initiate any of these tasks manually, you should use SA OS/390 facilities to do so. You may get confusing results if, for example, you use the SMEUI to stop or start applications servers that are controlled by SA OS/390.

3.3 Changing Configurations for WebSphere Application Server for OS/390 and z/OS

When you want to change parameters for a J2EE Server or install applications on a J2EE server, you have to do this by defining the changes in a new conversation through the SMEUI or the Administration API and, finally, activate the conversation. For your changes to be reflected in the running servers, they need to be stopped and restarted. This is done automatically during the activation step for a conversation. If a server cannot be stopped during activation, activation fails.

When System Automation for OS/390 controls WebSphere operations, it is, as a rule, responsible for monitoring servers and restarting them if they end normally or abnormally. This may lead to conflicts with the activation process.

There are two options for avoiding a conflict:

1. To let activation restart servers, perform the following steps
   a) Switch off automation for all J2EE servers
   b) Activate the conversation; all running servers will be stopped and restarted through activation
   c) Switch on automation after all servers have restarted
2. To use SA OS/390 to restart servers perform the following steps:
   a) Before you activate a conversation stop all J2EE Servers through SA /390
   b) Activate the conversation; no server restart will be triggered through activation since no servers are running

c) After activation is complete restart all J2EE servers through SA OS/390

The advantages and disadvantages of each approach are discussed below in 3.3.2 "Conversation Activation and System Automation" on page 30. First we outline the operational alternatives and their consequences on the SA OS/390 definitions.

3.3.1 Starting, Stopping, Monitoring Address Spaces

Four items play a role when you start, stop and monitor WebSphere control and server regions in sysplex:

1. the member name --

   this is the name of the library member containing the catalogued procedure that is used to start the server. This is a required parameter when you start a task.
2. the jobname --

   this is the name that will be assigned to the job. This is an optional parameter when you start a task. If the source JCL in the member is a procedure and you omit the JOBNAME keyword, the member name will be assigned as the job name.
3. the identifier used in the start command --

   this is optional; it is not used as a replacement for the jobname. You can not specify both jobname and identifier. If the identifier is specified, it is used in the DISPLAY, MODIFY, RESET, CANCEL, FORCE, and STOP commands for the started tasks
4. the server name --

   this is a WebSphere Application Server for OS/390 and z/OS specific parameter that tells the WebSphere which server instance is to be started.

Using Identifiers, not Jobname

In the WebSphere manuals it is recommended that you use the server name also as an identifier, and start the control regions with commands in the form

```
S ControlRegionProcName.server-instance,SRVNAME=' server-instance '
```

For example to start the daemon on the first system, issue

```
S BBODMN.DAEMON01,SRVNAME='DAEMON01'
```

To start it on the second system, issue

```
S BBODMN.DAEMON02,SRVNAME='DAEMON02'
```

To start an application server, issue

```
S BBOASR2.BBOASR2A,SRVNAME='BBOASR2A'
```

Stop commands that correspond to these start commands are

`P ControlRegionProcName.server-instance` or simply `P server-instance`

Cancel commands are analogous. The jobname in this case is the ControlRegionProcName.

Since SA OS/390 requires unique jobnames for the resources that it manages, you cannot start two servers on the same system using the same procedure but different server-instance names. You could, for example, attempt to start two application servers (APPSRV1 and APPSRV2) using the same procedure (J2EEPROC) within the same system by issuing

```
        S J2EEPROC.APPSRV1,SRVNAME='APPSRV1' and
        S J2EEPROC.APPSRV1,SRVNAME='APPSRV1'
```

You would not, however, be able to automate these two servers with SA OS/390.

WebSphere Application Server for OS/390 and z/OS Administration uses this form of commands when it starts and stops servers.

Using Jobname, no Identifiers

In the publication “MVS System Commands” IBM recommends that you use the JOBNAME parameter rather than an identifier. If you use the JOBNAME parameter, SMF records, messages, and automated programs can reflect or react to job status; identifiers can only be viewed at a console.

With the JOBNAME parameter you start the control regions with commands in the form

```
        S ControlRegionProcName,JOBNAME=server-instance,SRVNAME=' server-instance '
```

For example to start the daemon on the first system, issue

```
        S BBODMN,JOBNAME=DAEMON01,SRVNAME='DAEMON01'
```

To start it on the second system, issue

```
        S BBODMN,JOBNAME=DAEMON02,SRVNAME='DAEMON02'
```

To start an application server, issue

```
        S BBOASR2,JOBNAME=BBOASR2A,SRVNAME='BBOASR2A'
```

Stop commands that correspond to these start commands use the jobname (that we set to the name of the server-instance):

```
        P server-instance
```

Cancel command are analogous.The jobname in this case is the name of the server-instance.

SA OS/390 can automatically create start, stop and cancel commands from the information it has or gets when a server is started or ends normally or abnormally.

3.3.2 Conversation Activation and System Automation

As mentioned in 3.2 “WebSphere Application Server for OS/390 and z/OS Administration” on page 27, WebSphere Application Server for OS/390 and z/OS Administration shuts down and restarts all J2EE servers for which the configuration has been changed when the conversation containing these changes is activated.

When you automate WebSphere operations one way of achieving continuous availability is to automatically restart WebSphere components as fast as possible if they end normally or abnormally. Stopping severs that are controlled by an automation product will usually cause the automation product to immediately restart the servers.

If the automatic restart ends before the attempt by WebSphere Application Server Administration to restart, then this may prevent WebSphere Administration from considering activation to have successfully finished. (You may see the message: "Activate failed because server cannot be stopped." and the conversation state will not have changed to activated.)

If, however, the restart issued by WebSphere Application Server Administration is successful before the one initiated by the automation product the automatic restart will fail. This causes automation to get confusing messages: The ones coming from Administration's restart would say that the servers are up; the ones coming from automatic restart would say that restart failed. System automation would in such a case not set the actual state of the respective server to AVAILABLE and thus indicate that manual intervention is required.

To avoid this you can either

- temporarily disable automation and let activation restart servers or
- use SA OS/390 to stop servers before activation and restart them after activation has finished.

In both cases there is a maintenance interval period during that not all WebSphere Application Server for OS/390 and z/OS services are available.

Operations during such a maintenance interval can be automated. System Automation for OS/390 allows you to define a service period. Thus it provides the ability to make stop and start requests at specified points in time independently of human intervention. An operator can temporarily modify a service period if manual control is desired. The activities to be performed during this interval can be initiated manually or you can use OPC Automation.

OPC is now called Tivoli Workload Scheduler (TWS). OPC Automation is an extension of SA OS/390 that capitalizes on the strengths of NetView, SA OS/390, and TWSC by providing the ability to greatly expand job execution, scheduling, monitoring, and alert notification capabilities.

Using OPC Automation you could during a service period schedule a job to run that activates WebSphere Application Server for OS/390 and z/OS conversations automatically. But you still need to decide how you handle the process for stopping and restarting servers.

Disable Automation and let Activation Restart Servers

The steps to be performed with the maintenance interval are

1. Make your changes to the configuration. This can be done outside the service window.
2. Disable automation for all J2EE servers
3. Activate the conversation; all running servers will be stopped and restarted through activation
4. Enable automation after all servers have restarted

This approach impacts application availability as little as possible: WebSphere Administration shuts down servers as late in the activation process as possible and restarts the immediately after shutdown. In addition, it leaves untouched servers for which nothing has changed.

WebSphere Administration always uses the start command with the identifier notation (and thus without the JOBNAME keyword). Therefore SA OS/390 has to monitor on procedure name as job name after the restart by WebSphere Administration. To be consistent SA OS/390 must use the same form of the start command itself for restarts and moves. It must also use the identifier rather than the job name in its stop and cancel commands for WebSphere. Because these are not the commands SA OS/390 generates by default, they need to be individually defined for each server. In cases were there are many servers and systems involved this makes SA OS/390 definitions much more complex.

During the maintenance interval automation for all servers is switched off. Should any recovery action be required during this time for servers that would otherwise be unaffected by activation this action will not be triggered by SA OS/390.

Stop and Restart Servers using System Automation for OS/390

The steps to be performed with the maintenance interval are

1. Make your changes to the configuration. This can be done outside the service window.
2. Before you activate a conversation stop all J2EE Servers through SA OS/390
3. Activate the conversation; no server restart will be triggered through activation since no servers are running
4. After activation is complete restart all J2EE servers through SA OS/390

All J2EE servers are shut down before activation and can only be restarted when activation is complete. Therefore users cannot use any of the Web applications offered through these servers during this time.

WebSphere Administration does not restart servers in this case; SA OS/390 performs the restart. It can therefore use the start command with the JOBNAME parameter and the matching stop and cancel commands that are generated according to SA OS/390 standards. This makes setting up SA OS/390 for WebSphere much easier.

Chapter 4. System Setup

Our test environment consisted of a sysplex with three systems: KEYA, KEYB and KEYC. The infrastructure on each system was a shared HFS, a shared DB2, and TCP/IP and this was included in the automation setup.

The Sysplex Distributor was used for load balancing and port-availability monitoring.

The HTTP server was running under z/OS and was included in the automation set up.

We used two J2EE Servers: BBOASR2 and BBOBANK.

The defined availability target was:

- LDAP on two of the three systems
- IBM HTTP server on two of the three systems
- Two of three systems for BBOASR2
- Two of three systems for BBOBANK

We were using dynamic VIPA. The IP addresses on each of the three systems were defined for dynamic VIPA backup.

All relevant ports were defined to the Sysplex Distributor and distributed across all system images. This included the LDAP port, the HTTP Server port, the WebSphere Application Server's base environment (Daemon, Naming) ports and the J2EE server ports for the HTTP Transport Handler. Thus it was irrelevant on which two systems LDAP or HTTP Servers or J2EE Servers ran.

We run the base environment for WebSphere Application Server (Daemon, Naming) on each system, considering it as part or the WebSphere infrastructure. In case of a failover of a J2EE server this setup makes moving it to a different system much faster.
Alternatively you could start the base environment only on those systems where a J2EE server was started. So if for example BBOASR2 ran on KEYA and KEYB and BBOBANK ran on KEYB and KEYC all three systems had a Daemon and Naming running. But if both BBOASR2 and BBOBANK were running on KEYA and KEYB only, daemon and Naming server would not be required on KEYC.

Both HTTP servers had identical service definitions for the WebSphere plug-in. This was necessary because the HTTP port is distributed using the Sysplex Distributor.

Both LDAP servers could be used by any of the WebSphere instances because they used shared DB2 data and because their VIPA address was used in the WebSphere definitions.

As a consequence of using the IBM HTTP Server for z/OS we could not use session affinity. We therefore used persistent sessions, storing session information in shared DB2 tables.

Figure 10 shows how the components could be distributed over our three systems.

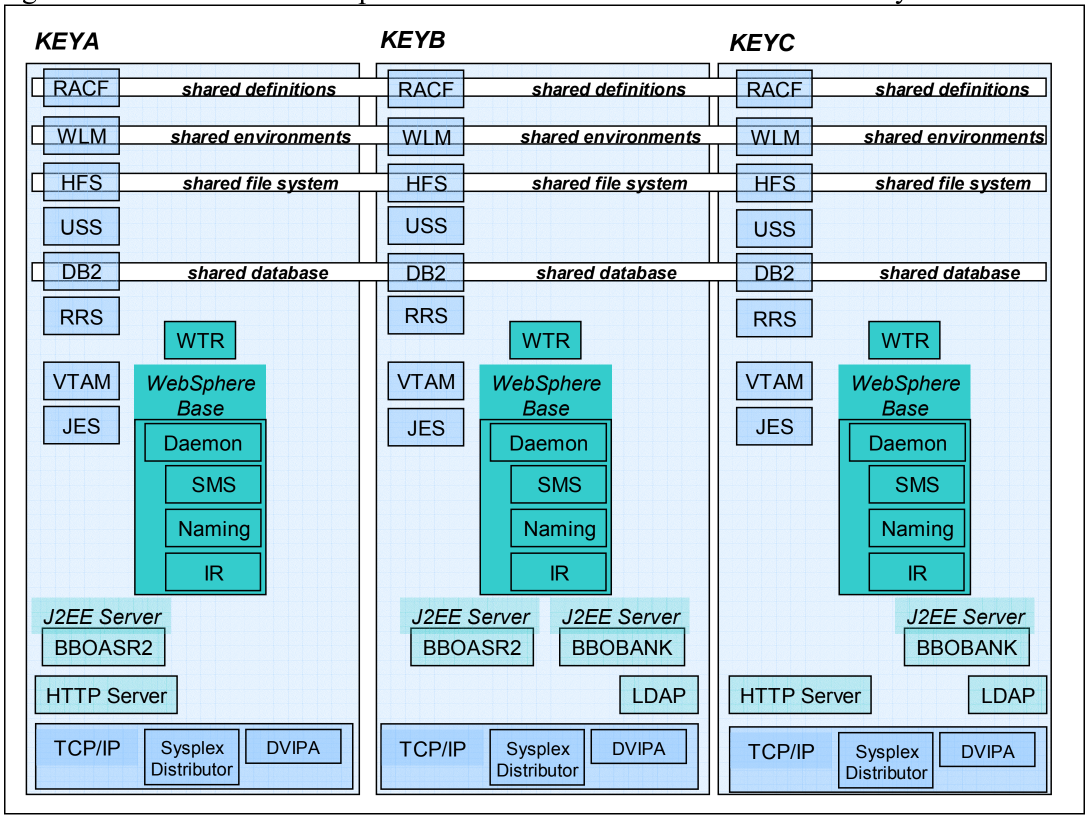


Figure 10: WebSphere on a Sysplex: Sample Configuration

Note that

- there are two instances of the HTTP Server (KEYA and KEYC)
- there are two instances of the LDAP Server (KEYB and KEYC)
- there are two instances of each J2EE Server
  - BBOASR2 runs on KEYA and KEYB
  - BBOBANK runs on KEYB and KEYC
- the WebSphere Base, as part of the infrastructure runs on all systems

Also note that HTTP server, LDAP server and J2EE servers were able to run on any of the three systems and that System Automation for OS/390 would move them to any other system in case of a failure that prevent restarts on the failing system. In this sense Figure 10 shows only one possible run-time environment for our sysplex.

The system set up is still fully functional (possibly with degraded performance) as long as at least one LDAP, one HTTP server, and one of each of the J2EE servers is running on any one of the three sysplex images. With backup IP stacks provided by dynamic VIPA the IP stack was not a single point of failure either. The only exception to completely continuous operation comes through the fact that WebSphere Application Servers needs to restart servers to activate configuration changes (see the discussion in the section 3.3 “Changing Configurations for WebSphere Application Server for OS/390 and z/OS” on page 28).

The following sections give details of our configured sample setup for WebSphere system automation. Only the major components that required setup decisions are listed and, even for these, only the most relevant parameters are mentioned.

4.1 Sysplex

Our sysplex consisted of three images: KEYA, KEYB and KEYC.

The set up covered a heterogeneous MVS environment:

- z/OS 01.04.0 on KEYA
- OS/390 02.10 on KEYB
- z/OS 01.02.0 on KEYC

Apart from this the systems were mostly symmetrical: all prerequisite products and all components of WebSphere Application Server for OS/390 and z/OS were installed and were able to run on any of the three systems.

Cross-system sharing was implemented where applicable: RACF definitions, DB2 and HFS were shared.

4.2 TCP/IP

The TCP addresses and the corresponding DNS names were:

- KEYA: DNS-Name boekeya.boeblingen.de.ibm.com on 9.152.64.73
- KEYB: DNS-Name boekeyb.boeblingen.de.ibm.com on 9.152.64.74
- KEYC: DNS-Name boekeyc.boeblingen.de.ibm.com on 9.152.64.77

Three VIPAs were defined:

- boekpla.boeblingen.de.ibm.com on 9.152.84.225 providing ports 9000, 5555, 1389, and 2389. These were the ports used for WebSphere base services and LDAP
- boekplb.boeblingen.de.ibm.com on 9.152.84.226 providing ports 80, 8081, 8082, and 8083. These were the ports used by the IBM HTTP server and the HTTP Transport Handlers of the J2EE Servers (8083 is spare)
- boekplc.boeblingen.de.ibm.com on 9.152.84.227 providing ports 20, 21, 23, and 623 for FTP, TELNET, and OTELNET

Dynamic XCF addresses were defined for KEYA, KEYB and KEYC: 9.152.84.249, 9.152.84.250 and 9.152.84.251.

The VIPAs were distributed over the dynamic XCF addresses. The tcp.profile for each system contained the following statements:

```
VIPADISTRIBUTE DEFINE 9.152.84.225 PORT 9000 5555 1389 2389
 DESTIP 9.152.84.249 9.152.84.250 9.152.84.251
VIPADISTRIBUTE DEFINE 9.152.84.226 PORT 80 8081 8082 8083
```

```
 DESTIP 9.152.84.249 9.152.84.250 9.152.84.251
VIPADISTRIBUTE DEFINE 9.152.84.227 PORT 20 21 23 623
 DESTIP 9.152.84.249 9.152.84.250 9.152.84.251
```

Each VIPA served as backup address, the tcp.profile BACKUP contained the following statements:

```
VIPADYNAMIC
VIPAbackup 100 9.152.84.225
VIPAbackup 100 9.152.84.226
VIPAbackup 100 9.152.84.227
ENDVIPADYNAMIC
```

This setup ensured high availability of the IP applications running on the sysplex cluster even if one physical network interface should fail or an entire IP stack or z/OS LPAR should be lost.

4.3 Base WebSphere Application Server for OS/390 and z/OS Setup

We used the defaults for WebSphere Application Server whenever possible. Here we only indicate the parameters that relate to the setup for automation and high availability.

4.3.1 **TCP Related Definitions**

Our bootstrap system was KEYA and we used its DNS name boekeya.boeblingen.de.ibm.com for the DAEMON_IPNAME parameter. Later we introduced VIPA. However, you cannot easily change the DAEMON_IPNAME at a later stage. Therefore we could not use the DNS name that is used for the dynamic IP address as DAEMON_IPNAME. This has the disadvantage that we had a single point of failure for WebSphere Administration: you can connect with the SMEUI to any system with a running daemon but connection will only be established if the daemon on KEYA is running. Had we used the VIPA DNS name boekpla.boeblingen.de.ibm.com that distributes the ports we used for System Management (9000) and for the DAEMON (5555), then we would have avoided this problem.

Other TCP related parameters were set as follows:

```
DAEMON_PORT=5555
RESOLVE_IPNAME= boekpla.boeblingen.de.ibm.com
RESOLVE_PORT=9000
```

We defined the port for the HTTP Transport Handlers for the J2EE Servers as:

```
for BBOASR2: BBOC_HTTP_PORT=8081
for BBOBANK: BBOC_HTTP_PORT=8082.
```

These ports were distributed through boekplb.boeblingen.de.ibm.com.

In the webcontainer.conf files we had virtual host definitions as follows:

for BBOASR2A on KEYA:

```
host.default_host.alias=
   boekeya.boeblingen.de.ibm.com:8081,boekeya.boeblingen.de.ibm.com,
   boekplb.boeblingen.de.ibm.com:8081,boekplb.boeblingen.de.ibm.com
```

for BBOASR2B on KEYB:

```
host.default_host.alias=
   boekeyb.boeblingen.de.ibm.com:8081,boekeyc.boeblingen.de.ibm.com,
   boekplb.boeblingen.de.ibm.com:8081,boekplb.boeblingen.de.ibm.com
```

for BBOASR2C on KEYC:

```
host.default_host.alias=
   boekeyc.boeblingen.de.ibm.com:8081,boekeyc.boeblingen.de.ibm.com,
   boekplb.boeblingen.de.ibm.com:8081,boekplb.boeblingen.de.ibm.com
```

for BBOBANKA on KEYA:

```
        host.default_host.alias=
            boekeya.boeblingen.de.ibm.com:8082,boekeya.boeblingen.de.ibm.com,
            boekplb.boeblingen.de.ibm.com:8082,boekplb.boeblingen.de.ibm.com
```

for BBOBANKB on KEYB:

```
        host.default_host.alias=
            boekeyb.boeblingen.de.ibm.com:8082,boekeyc.boeblingen.de.ibm.com,
            boekplb.boeblingen.de.ibm.com:8082,boekplb.boeblingen.de.ibm.com
```

for BBOBANKC on KEYC:

```
        host.default_host.alias=
            boekeyc.boeblingen.de.ibm.com:8082,boekeyc.boeblingen.de.ibm.com,
            boekplb.boeblingen.de.ibm.com:8082,boekplb.boeblingen.de.ibm.com
```

This allowed us to contact a specific instance on a specific system by using the DNS names mapped to the real IP address of that system. If, on the other hand, we used boekplb.boeblingen.de.ibm.com we reached the system chosen by the Sysplex Distributor.

On the Naming server, we defined the LDAP URL as follows:

```
com.ibm.ws.naming.ldap.masterurl=ldap://boekpla.boeblingen.de.ibm.com:1389
```

Port 1389 is distributed through boekplba.boeblingen.de.ibm.com. Therefore an LDAP server can run on any of our systems and still be reached by WebSphere Naming.

4.3.2 **HTTP Session Related Definitions**

We installed an application on BBOBANK that required HTTP session support. As discussed in 2.4.1 “HTTP Sessions” on page 24 we had to use persistent sessions. To enable persistent session support for BBOBANK we defined in the WebSphere Application Server conversation a J2EE resource named IBMHttpSession. Its resource type was DB2datasource, and the location name was KEYDNSA that defines the same shared DB2 we used for all WebSphere definitions and for all application data.

In the webcontainer.conf file for all BBOBANK instances we enabled persistent sessions as follows:

```
        session.enable=true
        session.urlrewriting.enable=false
        session.cookies.enable=true
        session.protocolswitchrewriting.enable=false
        session.cookie.name=JSESSIONID
        session.dbenable=true
        session.dbtablename=CU05.SESSTAB
```

4.4 HTTP server

The HTTP server listened on port 80. We enabled the WebSphere Application Server V4.0.1 for z/OS and OS/390 plug-in for all our applications. The definitions in the httpd.con file were:

```
      Port     80
      # ==========================
      #       WebSphere Directives V 4.0.1
      # ==========================
      ServerInit /usr/lpp/WebSphere/WebServerPlugIn/bin/was400plugin.so:init_exit
           /usr/lpp/WebSphere,/local/websrv/was.conf
      ServerTerm /usr/lpp/WebSphere/WebServerPlugIn/bin/was400plugin.so:term_exit
      Service /PolicyIVP/*
           /usr/lpp/WebSphere/WebServerPlugIn/bin/was400plugin.so:service_exit
      Service /PolicyTwo/*
           /usr/lpp/WebSphere/WebServerPlugIn/bin/was400plugin.so:service_exit
```

```
Service /webapp/examples/*
     /usr/lpp/WebSphere/WebServerPlugIn/bin/was400plugin.so:service_exit
Service /webapp/examplesTwo/*
     /usr/lpp/WebSphere/WebServerPlugIn/bin/was400plugin.so:service_exit
Service /TheBankWeb/*
     /usr/lpp/WebSphere/WebServerPlugIn/bin/was400plugn.so:service_exit
```

The SERVER INIT statement pointed to /local/websrv/was.conf. This file contained no application definitions, since we did not run servlets in the plug-in directly.

In the httpd.envars files for all systems we added the following statement:

```
HTTRESOLVE_IPNAME=boekpla.boeblingen.de.ibm.com
RESOLVE_PORT=9000
```

Chapter 5. Setup for System Automation for OS/390

In this chapter we describe the System Automation for OS/390 setup hat we implemented to automate WebSphere Application Server for OS/390 and z/OS. As outlined in 1.3 WebSphere Application Automation and High Availability on page 8, the main objective was to provide

- ease of operations support:
  - start all prerequisites and components of WebSphere in the right order
  - stop all components of WebSphere in the right order
  - show the status of all prerequisites and components
- availability support reducing downtimes:
  - monitor prerequisites and components
  - automatically restart failing components.

These objectives apply to a monoplex environment. In a sysplex environment System Automation for OS/390 can additionally increase availability by

- ensuring that WebSphere Application Servers run on a predefined number of images in the sysplex
- moving all or parts of WebSphere Application Server for OS/390 and z/OS from a failing image to another image when restart of vital component fails
- cleaning up failed system.

For WebSphere Application Server for OS/390 and z/OS administration a separate service period can be defined.

All applications and systems that you want to include in automation must be defined to System Automation for OS/390 in an automation *policy databas*e. This database contains the objects to be managed by System Automation for OS/390, and the rules according to that automation of these objects proceeds.

The objects that are defined in the policy database are called *policy objects* or *entrie*s. Applications and systems, for example, are policy objects. Every policy object belongs to an *entry type* that is identified by a three letter code. The most common entry types are APL (application), APG (application group), GRP (groups of systems, i.e. sysplexes) and SYS (system).

Using the System Automation for OS/390 customization dialog you provide information to System Automation for OS/390 such as:

- which resources you want to automate, monitor and control.
- how resources are to be associated (grouped) with each other for automation
- the relationships between resources and groups of resources.
- which automation, such as automatic startup or shutdown, is to be applied to these resources and how

- under which conditions automated actions should occur, and what actions should be performed
- when automation should be active, and how automation handles certain tasks and events during system operation.

As a starting point, the following table contains a list of the main applications defined for WebSphere Application Server automation. Included are the names used in the policy database, a short description and the number of systems the application should run on. Please note that all the applications that are listed were able to run on any of our systems.

	Name	Description	Active Instances
JES2	JES2	Job Entry Subsystem 2	On all systems
VTAM	VTAM	Virtual Telecommunication Access Method	On all systems
TCP/IP	TCPIP	TCP/IP	On all systems
WTR	BBOWTR	WebSphere trace writer	On all systems where the Daemon runs
RRS	RRS	Resource Recovery Services	On all systems
DB2 Server	DB2_DBM1 DB2_DIST DB2_IRLM DB2_MSTR DB2_SPAS	DB2 Subsystem	On all systems
LDAP Server	BBOLDAP	Websphere LDAP	On 2 of 3 systems
WebSphere Base	BBODMN	WebSphere Daemon	On all systems
	BBOSMS	WebSphere SMS control region	On all systems
	BBONM	WebSphere Naming-Server control region	On all systems
	BBOIR	WebSphere I/F-repository control region	On all systems
J2EE Server	BBOASR2	WebSphere J2EE-server control region	On 2 of 3 systems
	BBOBANK	WebSphere J2EE-server control region	On 2 of 3 systems
HTTP Server	WEBSRV	IBM HTTP Server	On 2 of 3 systems

Additionally, for WebSphere MQSeries, we automated the MQSeries Queue Manager and the Channel Initiator, although this is not required for WebSphere.

Figure 11 gives an overview of the defined groups.

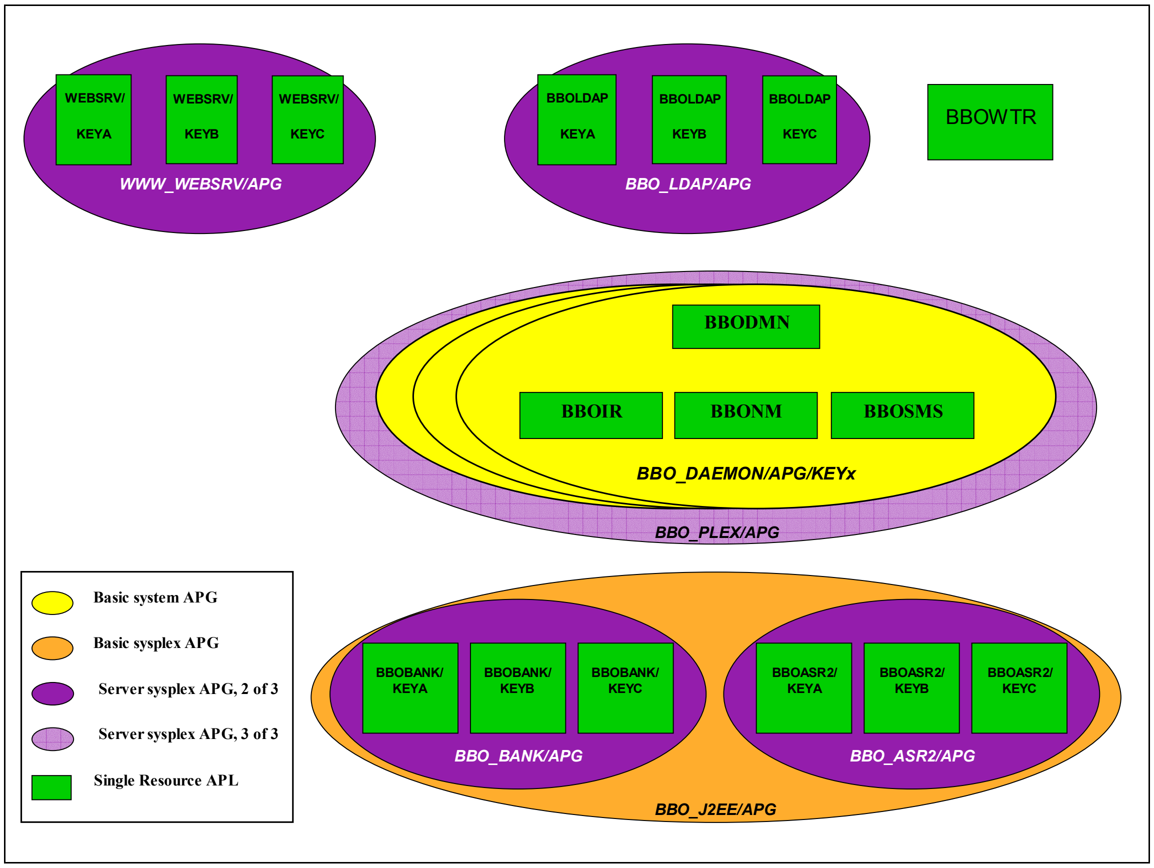


Figure 11: Groups and Applications in a Sysplex Environment

5.1 Sysplex Group

The main purpose of the sysplex group named KEYAPLEX group is to select the systems included in the sysplex and the application groups of the sysplex. It contains the following:

Short description	sysplex with systems KEYA, KEYB, KEYC
Group type	SYSPLEX

Linked Systems	
Entry Name	**Short Description**
KEYA	System KEYA
KEYB	System KEYB
KEYC	System KEYC

Linked ApplicationGroups	
Entry Name	**Short Description**
BBO_ASR2	WebSphere J2EE sysplex server group
BBO_BANK	WebSphere J2EE sysplex server group
BBO_J2EE	WebSphere sysplex basic group
BBO_LDAP	WebSphere LDAP sysplex server group
BBO_MQPLEX	WebSphere MQ Series sysplex basic group
BBO_PLEX	WebSphere sysplex server group
SGA_PLEX	DB2 - SGAn sysplex basic group
WWW_WEBSRV	IBM HTTP Server sysplex server group

If you have more systems in your sysplex include them in **Linked Systems.** If you have additional J2EE or CORBA servers include them as **Linked ApplicationGroups.**

5.2 KEYA, KEYB, KEYC

These system groups define the clone values to be used for AOCCLONE and AOCCLONE1 and all application groups for each system.

Short description	System KEYA
Clone value 0	1
Clone value 1	A

Linked ApplicationGroups	
Entry Name	**Short Description**
BBO_DAEMON	WebSphere system basic group
BBO_MQ	WebSphere MQ Series system group
SGA_SYS	DB2 - SGAn basic system group

KEYB and KEYC are similar (except for the clone values):

Short description	System KEYB
Clone value 0	2
Clone value 1	B

and

Short description	System KEYC
Clone value 0	3
Clone value 1	C

5.3 Application Groups

The AM_KEY application group defines all Automation Managers, SYSVIEW SA OS/390 Agent Resources. The BASE_SYS group defines basic system components such as JES2,VLF,LLA. The groups BBO_MQPLEX and BBO_MQ are included for WebSphere MQ Series and are not used here. NETWORK defines networking resources. SGA_PLEX and SGA_SYS defines DB2. These groups are considered extensions of the base system. No details are given here.
The following application groups are directly related to our WebSphere set-up:

- BBO_PLEX
- BBO_DAEMON
- BBO_ASR2
- BBO_BANK
- BBO_LDAP
- WWW_WEBSRV

Some details are given below. For a full description please refer to our sample PDB.

The relationships between groups or applications are discussed in 5.5 Relationships on page 54.

5.3.1 **BBO_PLEX**

This group defines the WebSphere base environment.

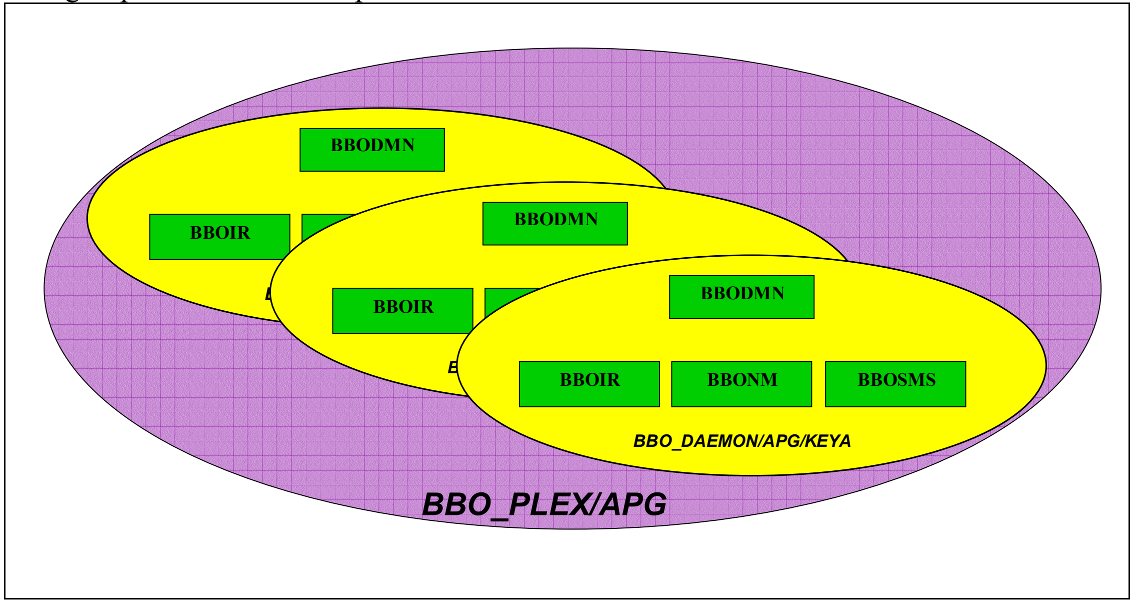


Figure 12: BBO_PLEX WebSphere Base Environment

Here are the definitions:

Short description	WebSphere sysplex server group
Long description	group with BBO_DAEMON groups included
Application Group Type	SYSPLEX
Nature	Server
Automation Name	BBO_PLEX
Behaviour	ACTIVE
Default Preference	*DEF
Auto-link APL Resources to APG	YES
Availability Target	*ALL

Linked Resources				
Resource Name	**Entry Type**	**System**	**Entry Name**	**Preference Value**
BBO_DAEMON/APG/KEYA	APG	KEYA	BBO_DAEMON	700
BBO_DAEMON/APG/KEYB	APG	KEYB	BBO_DAEMON	700
BBO_DAEMON/APG/KEYC	APG	KEYC	BBO_DAEMON	700

Since BBO_PLEX is included in the **Linked ApplicationGroups** of our sysplex group KEYAPLEX all systems defined there are by default included as **Linked Resources.**
If you have systems where you never want the WebSphere infrastructure to run you must exclude then from this list.

Note that we consider the WebSphere base as part of the infrastructure and therefore specified an **Availability Target** of 3. This means that it will automatically be started on all of our systems. The advantage is that System Automation for OS/390 can move a J2EE server from a failed system to a different system without first having to start the base WebSphere environment. You can change the availability target to a lower value. In that case it can happen that System Automation for OS/390 will first have to start the base environment before it can move a J2EE server that, of course takes additional time.

A J2EE server can only run on a system where the base WebSphere Application Server environment is running. You should therefore set the **Availability Target** at least as high as the maximum of the targets for all the J2EE Servers APGs.

5.3.2 BBO_DAEMON

This basic system group defines that the WebSphere infrastructure consists of the Daemon, System Management, Naming and Interface Repository.

Short description	WebSphere system basic group
Long description	Group with Daemon, SMS, Naming and IR included
Application Group Type	SYSTEM
Nature	Basic
Automation Name	BBO_DAEMON

Behaviour	ACTIVE
Default Preference	*DEF
Auto-link APL Resources to APG	YES

Linked Applications	
Entry Name	**Short Description**
BBODMN	WebSphere Daemon
BBOIR	WebSphere I/F-repository control region
BBONM	WebSphere Naming-Server control region
BBOSMS	WebSphere SMS control region

5.3.3 **BBO_J2EE**

This basic sysplex group joins together the server groups for the two J2EE servers we use. It can be used to stop and restart all J2EE servers during WebSphere maintenance (see the discussion in 3.3 Changing Configurations for WebSphere Application Server for OS/390 and z/OS on page 28).

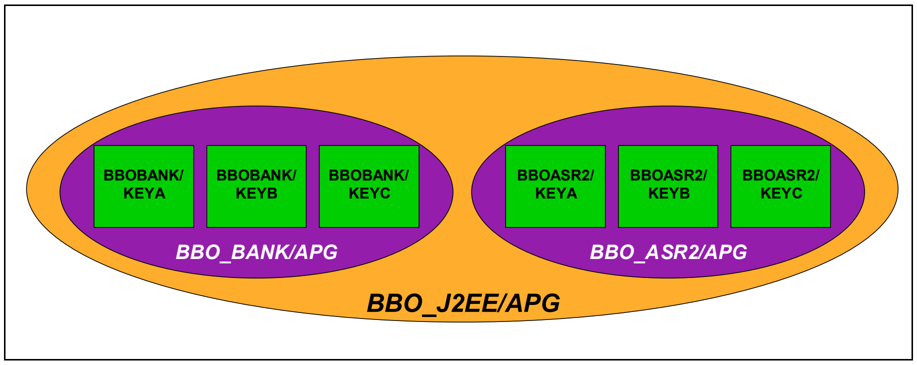


Figure 13: BBO_J2EE Group for J2EE Servers

Here is its description:

Short description	WebSphere sysplex basic group
Application Group Type	SYSPLEX
Nature	Basic
Automation Name	BBO_J2EE
Behaviour	ACTIVE
Default Preference	*DEF
Auto-link APL Resources to APG	YES

Linked Resources			
Resource Name	**Entry Type**	**Entry Name**	**Preference Value**

BBO_ASR2/APG	APG	BBO_ASR2	SELECTED
BBO_BANK/APG	APG	BBO_BANK	SELECTED

If you have more J2EE servers you should include them here as **Linked Resources**.

5.3.4 **BBO_ASR2**

This sysplex server group covers one of our two J2EE servers.

Short description	WebSphere J2EE sysplex server group
Application Group Type	SYSPLEX
Nature	Server
Automation Name	BBO_ASR2
Behaviour	ACTIVE
Default Preference	*DEF
Auto-link APL Resources to APG	YES
Availability Target	2

Linked Resources				
Resource Name	**Entry Type**	**System**	**Entry Name**	**Preference Value**
BBOASR2/APL/KEYA	APL	KEYA	BBOASR2	700
BBOASR2/APL/KEYB	APL	KEYB	BBOASR2	700
BBOASR2/APL/KEYC	APL	KEYC	BBOASR2	700

The main purpose of this group is to define that an instance of this J2EE server can run on any of our three systems (with no specific priority as to on that it should be started) and that two instances should be started under normal circumstances.

If you have more systems in your sysplex, all of that are included in the sysplex group, then all are listed in **Linked Resources. Y**ou may exclude some systems explicitly to prevent them from being a candidate for running the J2EE server.
If you have preferred systems for this specific J2EE server, change the **Preference Value** of the systems.

Adapt the **Availability Target** for each J2EE server to your needs according to the expected load for the server.

Note: you have a single point of failure if you use a target of 1. The applications installed on this J2EE server are then not available while System Automation for OS/390 restarts the J2EE server on another system in case of a failover of the current system.

5.3.5 **BBO_BANK**

This group does exactly the same for our second J2EE server what BBO_ASR2 does for the first one.

5.3.6 **BBO_LDAP**

This sysplex server group defines that LDAP servers can run on any of our three systems and that two LDAP servers should normally be started.

Short description	WebSphere LDAP sysplex server group
Application Group Type	SYSPLEX
Nature	Server
Automation Name	BBO_LDAP
Behaviour	ACTIVE
Default Preference	*DEF
Auto-link APL Resources to APG	YES
Availability Target	2

Linked Resources				
Resource Name	**Entry Type**	**System**	**Entry Name**	**Preference Value**
BBOLDAP/APL/KEYA	APL	KEYA	BBOLDAP	700
BBOLDAP/APL/KEYB	APL	KEYB	BBOLDAP	700
BBOLDAP/APL/KEYC	APL	KEYC	BBOLDAP	700

If you use a set-up similar to the one we use you can have LDAP servers on as many systems as you need for covering the load (increase **Availability Target** if necessary). If you have preferred systems for your LDAP servers change the settings for **Preference Value**.

5.3.7 **WWW_WEBSRV**

This sysplex server group defines that HTTP servers can run on any of our three systems and that two HTTP servers should normally be started.

Short description	IBM HTTP Server sysplex server group
Application Group Type	SYSPLEX
Nature	Server
Automation Name	WWW_WEBSRV
Behaviour	ACTIVE
Default Preference	*DEF
Auto-link APL Resources to APG	YES

Availability Target	2

Linked Resources				
Resource Name	**Entry Type**	**System**	**Entry Name**	**Preference Value**
WEBSRV/APL/KEYA	APL	KEYA	WEBSRV	700
WEBSRV/APL/KEYB	APL	KEYB	WEBSRV	700
WEBSRV/APL/KEYC	APL	KEYC	WEBSRV	700

If you use a set-up similar to the one we use you can have HTTP servers on as many systems as you need for covering the load (increase **Availability Target** if necessary). If you have preferred systems for your HTTP servers change the settings for **Preference Value**.

5.4 Applications

The following applications are specific to our WebSphere set-up and will be briefly discussed here:

- BBO_CLASS
- BBODMN
- BBOSMS
- BBONM
- BBOIR
- BBOWTR
- BBOASR2
- BBOBANK
- BBOLDAP
- WEBSRV

The relationships between groups or applications are discussed in 5.5 Relationships on page 54.

Also included in the PDB are the definitions for WebSphere MQSeries. Since MQSeries is not required for WebSphere Application Server the definitions are not discussed here.

5.4.1 **BBO_CLASS**

BBO_CLASS represents a common policy to be inherited the WebSphere daemon, the J2EE and the LDAP servers.
It decreases the effort for defining the entries for each server individually and keeps the common definitions consistent.

Short description	WebSphere class with general definitions
Object Type	CLASS
Application Type	STANDARD
Subsystem Name	BBO_CLASS

Application Messages and User Data

Message id	Description
SHUTFORCE	Executed when force shutdown is invoked

Pass	Command Text
1	MVS C &SUBSJOB

Application Messages and User Data	
Message id	**Description**
SHUTIMMED	Executed when immediate shutdown is invoked

Pass	Command Text
1	MVS P &SUBSJOB
2	MVS C &SUBSJOB

Application Messages and User Data	
Message id	**Description**
SHUTNORM	Executed when normal shutdown is invoked

Pass	Command Text
1	MVS P &SUBSJOB
3	MVS C &SUBSJOB

Linked Instances	
Instance Name	**Description**
BBOASR2	WebSphere J2EE-control region
BBOBANK	WebSphere J2EE-control region
BBODMN	WebSphere Daemon
BBOLDAP	WebSphere LDAP

The common definitions for all servers cover the actions to be taken during shutdown. If you have additional J2EE servers you should link them to this class.

5.4.2 **BBODMN**

This is the definition for the WebSphere daemon.

Short description	WebSphere Daemon
Linked to Class	BBO_CLASS
Application Type	STANDARD
Subsystem Name	BBODMN
MVS job name	DAEMON0&AOCCLONE.

JCL Procedure Name	BBODMN

Subsystem Startup Parameters	,SRVNAME=DAEMON0&AOCCLONE.

Application Messages and User Data	
Message id	**Description**
PRESTART	Executed before startup is initiated

Command Text
MVS V WLM,APPLENV=CBNAMING,RESUME
MVS V WLM,APPLENV=CBINTFRP,RESUME
MVS V WLM,APPLENV=CBSYSMGT,RESUME
INGRCLUP BBOSMSS
INGRCLUP BBONMS
INGRCLUP BBOIRS
MVS C BBOSMSS
MVS C BBONMS
MVS C BBOIRS

Application Messages and User Data	
Message id	**Description**
SHUTFINAL	Executed after final termination message

Command Text
INGRCLUP BBOSMSS
INGRCLUP BBONMS
INGRCLUP BBOIRS
MVS C BBOSMSS
MVS C BBONMS
MVS C BBOIRS

The WLM environments for System Management, Naming and Interface Repository may have been stopped by too many failing attempts to restart the component. Because the scope of WLM is sysplex-wide this may have happened on a different system from the one where the Daemon is to be started now. Therefore the WLM environments are activated again (RESUMEd) before the Daemon is started.

Normally, when the WebSphere daemon terminates SMS, Naming and IR end as well and the respective control region and server region address spaces are no longer available. If termination of the daemon fails to end all the dependent servers, inactive address spaces may be left in the system. To avoid problems System Automation issues cancel commands for the dependent control regions in the final termination phase of the daemon. The same cancel commands are issued before the daemon is started to protect against the case that the

SHUTFINAL phase was not executed because no final termination message was received by System Automation.

Note that only control regions are known to System Automation. To terminate server regions you can use INGRCLUP. INGRCLUP calls a clean up routine that checks for and, if necessary, terminates address spaces. INGRCLUP is called after termination of the WebSphere daemon to terminate SMS, Naming and IR server regions if the daemon ends in such a way, that if cannot clean up on its own accord. INGCLUP is also executed before the daemon is started. This may be necessary if the WebSphere daemons abends in such a way that it even fails to notify System Automation of the abend.

For more details on the cleanup routine refer to 5.7 Clean-up Program on page 58.

5.4.3 **BBOSMS**

The WebSphere System Management Server is supervised by, but not started or stopped by System Automation.

Short description	WebSphere SMS control region
Application Type	STANDARD
Subsystem Name	BBOSMS
MVS job name	BBOSMS

External Startup	ALWAYS
External Shutdown	ALWAYS

Application Messages and User Data	
Message id	**Description**
BBOU0199E	WLM environment stopped

Pass	**Command Text**
1	MVS V WLM,APPLENV=CBSYSMGT,RESUME
2	HALTMSG JOBNAME=&SUBSJOB

The WebSphere System Management Server is started and stopped by the WebSphere Daemon (**External** as far as System Automation is concerned, see 5.5.3 The WebSphere Base Environment on page 56).
Note: Only the SMS Control Region is known to System Automation. Server regions are not under System Automation control. However, the WLM environment that starts server regions is resumed if it has previously been stopped by WLM. This happens while the SMS control region is running and is triggered by the BBOU0199E message.

5.4.4 **BBONM and BBOIR**

BBONM and BBOIR are treated analogously to BBOSMS.

5.4.5 **BBOWTR**

System Automation starts and stops the WebSphere trace writer as part of the Base system.

Short description	WebSphere trace writer
Application Type	STANDARD
Subsystem Name	BBOWTR
MVS job name	BBOWTR

Application Messages and User Data	
Message id	**Description**
SHUTFORCE	Executed when force shutdown is invoked

Pass	**Command Text**
1	MVS TRACE CT,WTRSTOP=&SUBSJOB
2	MVS C &SUBSJOB
2	MVS FORCE &SUBSJOB,ARM

Application Messages and User Data	
Message id	**Description**
SHUTIMMED	Executed when immediate shutdown is invoked

Pass	**Command Text**
1	MVS TRACE CT,WTRSTOP=&SUBSJOB
2	MVS C &SUBSJOB
2	MVS FORCE &SUBSJOB,ARM

Application Messages and User Data	
Message id	**Description**
SHUTINIT	Executed when shutdown is initiated

Command Text
MVS TRACE CT,OFF,COMP=SYSBBOSS

Application Messages and User Data	
Message id	**Description**
SHUTNORM	Executed when normal shutdown is invoked

Pass	**Command Text**
1	MVS TRACE CT,WTRSTOP=&SUBSJOB
3	MVS C &SUBSJOB
3	MVS FORCE &SUBSJOB,ARM

Application Messages and User Data	
Message id	**Description**

STARTUP	Executed to initiate the startup

Command Text
MVS TRACE CT,WTRSTART=&SUBSJOB

If you want to start trace writers for other components you can model System Automation definitions for those after the definitions for the WebSphere trace writer shown here.

5.4.6 **BBOASR2**

This is the first J2EE server definition:

Short description	WebSphere J2EE-control region
Linked to Class	BBO_CLASS
Application Type	STANDARD
Subsystem Name	BBOASR2
MVS job name	BBOASR2&AOCCLONE1.
JCL Procedure Name	BBOASR2

Subsystem Startup Parameters	,SRVNAME=BBOASR2&AOCCLONE1.

Application Messages and User Data	
Message id	**Description**
BBOU0199E	WLM environment stopped

Pass	**Command Text**
1	MVS V WLM,APPLENV=BBOASR2,RESUME
2	HALTMSG JOBNAME=&SUBSJOB

Application Messages and User Data	
Message id	**Description**
PRESTART	Executed before startup is initiated

Command Text
MVS V WLM,APPLENV=BBOASR2,RESUME
INGRCLUP BBOASR2S

Application Messages and User Data	
Message id	**Description**
SHUTFINAL	Executed after final termination message

Command Text
INGRCLUP BBOASR2

Note: Only the Control Region is known to System Automation. Server Regions are not under System Automation control.

However, the WLM environment that starts server regions is resumed if it has previously been stopped by WLM. This happens once while the J2EE control region is running triggered by the BBOU0199E message. It also happens before the J2EE server is started.

Normally, when the J2EE control region terminates all its server regions end as well and their address spaces are no longer available. If this process fails inactive address spaces may be left in the system. INGRCLUP is therefore called after termination of the J2EE server control ends to terminate server regions if the control region ends in such a way, that if cannot clean up on its own accord. INGRCLUP is also executed before the control region is started. This may be necessary if the J2EE control region abends in such a way that it even fails to notify System Automation of the abend.

For more details on the cleanup routine refer to 5.7 Clean-up Program on page 58.

5.4.7 **BBOBANK**

The definitions for the second J2EE server are analogous to the definitions for the first J2EE server.

5.4.8 **BBOLDAP and WEBSRV**

No special considerations apply to the LDAP server and the HTTP server.

5.5 Relationships

Figure 14 completes Figure 11 “Groups and Applications in a Sysplex Environment” on page 41 by including the relationships defined between groups and applications.

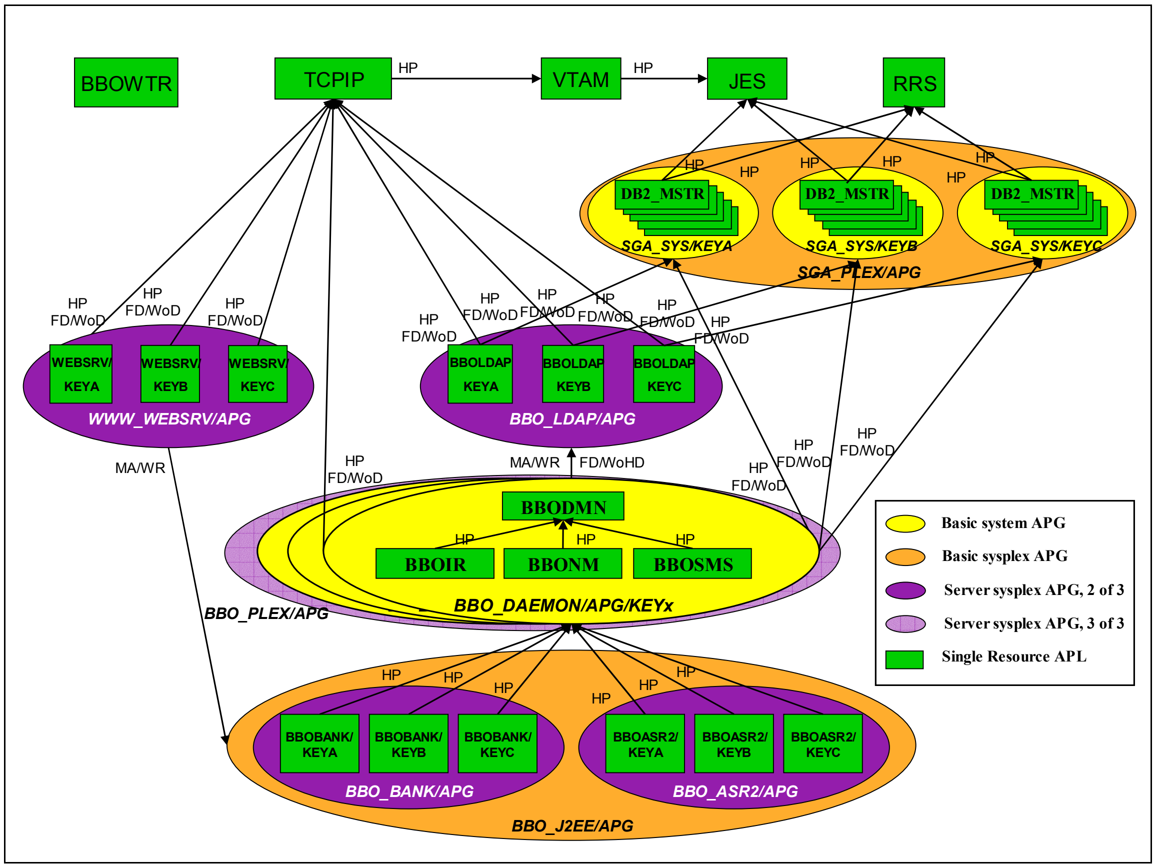


Figure 14: Relationships

Some details follow.

5.5.1 **LDAP**

Each LDAP server needs TCP/IP and DB2.

Relationships			
Relationship Type	**Supporting Resource**	**Description**	**Satisfy condition**
FORCEDOWN	SGA_SYS/APG/=	stop LDAP if DB2 fails	WhenObservedDown
FORCEDOWN	TCPIP/APL/=	stop LDAP if TCPIP fails	WhenObservedDown
HASPARENT	SGA_SYS/APG/=	LDAP is dependent on DB2	
HASPARENT	TCPIP/APL/=	LDAP is dependent on TCPIP	

5.5.2 **The IBM HTTP Server**

The HTTP Server needs TCP/IP.

Relationships			
Relationship Type	**Supporting Resource**	**Description**	**Satisfy condition**
FORCEDOWN	TCPIP/APL/=	stop HTTP server if TCPIP fails	WhenObservedDown
HASPARENT	TCPIP/APL/=	HTTP server is dependent on TCPIP	

No webserver should be started if not needed, i.e. if no J2EE server runs. Therefore the WWW_WEBSRV group depends on the J2EE server group.

Relationships					
Relationship Type	**Supporting Resource**	**Description**	**Automation**	**Chaining**	**Satisfy condition**
MAKEAVAILABLE	BBO_J2EE/APG	start if at least 1 J2EE server active	ACTIVE	WEAK	WhenRunning

5.5.3 **The WebSphere Base Environment**

The applications BBOIR, BBONM and BBOSMS have a HASPARENT relationship to the application BBODMN.

Relationships			
Relationship Type	**Supporting Resource**	**Description**	**Satisfy condition**
HASPARENT	BBODMN/APL/=	IR is started by WAS daemon	StartsMeAndStopsMe

BBOIR, BBONM and BBOSMS cannot be started until the BBODMN is available. They cannot be stopped before the daemon is unavailable.

Actually the **Satisfy condition** StartsMeAndStopsMe takes care of the fact that BBOIR, BBONM and BBOSMS are started by the daemon.

The daemon, BBOIR, BBONM and BBOSMS are the applications in the BBO_DAEMON group that represent the WebSphere base environment on each system. This group has the following relationships:

Relationships

Relationship Type	Supporting Resource	Description	Automation	Chaining	Satisfy condition
FORCEDOWN	BBO_LDAP/APG	stop WAS if LDAP fails			WhenObservedHardDown
FORCEDOWN	SGA_SYS/APG/=	stop WAS if DB2 fails			WhenObservedDown
FORCEDOWN	TCPIP/APL/=	stop WAS if TCPIP fails			WhenObservedDown
HASPARENT	SGA_SYS/APG/=	start/stop WAS dependent on DB2			
HASPARENT	TCPIP/APL/=	start/stop WAS dependent on TCPIP			
MAKEAVAILABLE	BBO_LDAP/APG	start WAS if at least 1 LDAP available	ACTIVE	WEAK	WhenRunning

This means that the WebSphere base environment can only be started when the local DB2 is running and has to be stopped if the local DB2 fails.

The WebSphere base environment can only be started if the local TCP is running and has to be stopped if the local TCP fails.

It should be started when the LDAP group is running, that means that at least one of the LDAP servers runs and has to be stopped when the group is hard down, that means that no LDAP server is running.

5.5.4 J2EE Servers

BBOASR2 and BBOBANK have a relationship of HASPARENT to the application group BBO_DAEMON.

Relationships		
Relationship Type	**Supporting Resource**	**Description**
HASPARENT	BBO_DAEMON/APG/=	J2EE server dependent on daemon

A J2EE server cannot be started until the local daemon group is available. The daemon group cannot be stopped before the J2EE Server resource is unavailable. The J2EE server is not started by the daemon. We assume that the daemon is part of the infrastructure and should therefore always be running. So the request to start the J2EE Server will not result in the issuing of a start request against the daemon. A start request must have been issued against the daemon before the start request for the J2EE server resource is processed.

If you have additional J2EE servers in your installation you should define an equivalent relationship for each of them.

You should also check if you have additional dependencies between the applications you run on your J2EE servers and other resources such as MQSeries or CICS or IMS and so on. If you identified any other dependencies you should add additional relationships here.

5.6 Message Automation Table

SA OS/390 provides an enhanced message table including messages needed to automate WebSphere Application Server for OS/390 and z/OS through APAR OA02375.

Messages issued during start and termination processing as well as messages that indicate error conditions on that System Automation can react are included for the following components:

- WebSphere
  - the base environment with daemon, System Management Services, Naming and Interface Repository
  - J2EE Server control region
- Related components
  - Trace writer
  - LDAP Server
  - IBM HTTP Server
  - TCP/IP subcomponents

Also included in the enhanced message table are WebSphere MQ Series messages.

5.7 Clean-up Program

The clean-up program is a safeguard against the case that address spaces are left over in the system after termination of a component that would normally also terminate dependent address spaces.

For WebSphere Application Server for OS/390 and z/OS could happen for

- System Management Services, Naming and Interface Repository control regions that are started by the daemon
- Server regions started by those control regions or by J2EE control regions when needed to satisfy the WLM defined service goals.

Since the control regions will normally have unique jobnames simple MVS cancel or stop commands can be issued by System Automation during shut-down or pre-start processing as shown in 5.4.2 BBODMN on page 49.

The server regions for a control region will however all have the same jobname. To cancel those you have to issue individual cancel commands including jobname and address space ID. This is done by the cleanup program. You call the cleanup program with the command

```
INGRCLUP jobname
```

To reduce the risk of inadvertently destroying address spaces, we allow only fully qualified jobnames and only jobnames that do not address a resource defined in the automation policy.

5.8 WebSphere Maintenance

This section addresses what you can do to automate the activation of WebSphere configuration changes and minimize the impact this has on availability due to the fact that WebSphere Application Server for OS/390 and z/OS Administration shuts down and restarts all running J2EE servers with changed configuration when the conversation containing these changes is activated.

You can either use basic System Automation for OS/390 or extend it by OPC Automation.

- System Automation for OS/390 allows you to define service period. In this way it provides the possibility to make stop and start requests at specified points in time independently of human intervention. An operator can temporarily modify a service period if manual control is desired. The activities to be performed during this interval could be initiated manually or started by System Automation.
- OPC itself refers to the Tivoli Workload Scheduler. OPC Automation is an extension of SA OS/390 to collaborate with Tivoli Workload Scheduler. This expands your job execution, scheduling, monitoring, and alert notification capabilities. Using OPC Automation you could, during a service period, use Tivoli Workload Scheduler to stop through System Automation, the WebSphere J2EE servers, then run a job that activates WebSphere Application Server for OS/390 and z/OS conversations and finally restart the WebSphere J2EE servers through System Automation.

The steps to be performed with the maintenance interval are

1. Make your changes to the configuration. This can be done at any time outside the maintenance window. You should use the WebSphere Application Server for OS/390 and z/OS SMEUI and go through all the steps except the last one that is conversation activation.
2. Before you activate a conversation stop all J2EE Servers through SA OS/390 or OPC Automation.
3. Activate the conversation; no server restart will be triggered through activation since no servers are running. You can either do this manually through the SMEUI or write little REXX execs to perform the conversation activation using the WebSphere Application Server for OS/390 and z/OS SM Scripting API. For this moment the only language that

is supported by the SM Scripting API is REXX. When you use this interface you can invoke the execs through a job that you can automate through OPC. Details are described below. Samples can be found in the download zone.

4. After activation is complete restart all J2EE servers through SA OS/390 or OPC Automation.

5.8.1 **Activation using the SM Scripting API**

To use the SM Scripting API you have to write a REXX script that you should run under OMVS.

A typical SM Scripting API script consists of three parts:

- One part to generate the input file
- One part to call the function
- One part to process the result

The following functions, supplied by the API, were used in the sample EXECs:

- CB390CFG(...) for calling an administrations function
- XMLGEN(...) for generating the input file

To use one of the administration functions, the default input XML files must be accessible. Each default XML file lists all valid attributes for the corresponding administrative function.

To use the API additions to the LIBPATH, CLASSPATH and PATH plus one new environment variable are needed.

You must run under an OMVS userid registered as a WebSphere for z/OS Systems Management Administrator such as CBADMIN.

Note: The scripts cannot be run from another userid that has done a switch user to become CBADMIN.

Tools Samples

As a sample we provide two jobs, scripts and REXX EXECs.

- Each job sets the userid to CBADMIN and calls a script to be executed under OMVS.
- Each OMVS script set up the required environment variables and calls the respective REXX EXEC.
- The REXX EXECs call the administration function and set the return code.
  - WASMNTR1 scans all conversations to find if there are any conversations ready to be deployed.
    - If it finds no conversation it set the return code to 4.
    - If it finds exactly one conversation, it generates an XML file that can be used as input by WASMNTR2 to activate the conversation. The return code is set to 0.
    - If it finds more than one conversation it still generates the XML file, but set the return code to 8.

– WASMNTR2 commits (activates) the conversation named in the XML input file. If more than one conversation is contained in the input file, it activates the most recent one.

Note that WebSphere Application Server for OS/390 and z/OS allows you to have more than one conversation that is in a state where it could be activated. But normally this should be avoided: After one of them has been activated WebSphere will not allow you to active the others. This is the reason for setting a return code of 8 by WASMNTR1 in this case.

For more details, look at the source of the samples.

Using these or similar jobs you could perform the following steps:
Run the first job

1. if return code is 4: no further action is needed, there is nothing to be activated.
2. if return code is 0: there is exactly one conversation to be activated. You should
   - stop all J2EE servers either through System Automation commands or through OPC Automation,
   - run the second job to activate the conversation,
     - if return code is 0, the new conversation has successfully been actived. Restart all stopped J2EE servers either through System Automation commands or through OPC Automation.
     - else restart all stopped J2EE servers, offline check what the problem was, schedule a new maintenance period when the problem is resolved.
3. if return code is 8
   - offline check the input file for the second step. Through SMEUI delete all conversations you do not want to activate and schedule a new maintenance period.
4. if return code is greater than 8
   - offline check the output and error files and fix the problem,
   - when the problem is fixed, schedule a new maintenance period.

Possible error causes could be:

- set-up problems, e.g.: wrong CLASSPATH, PATH, LIBPATH environment variables
- system problems, e.g.: WebSphere SMS on bootstrap system not running
- authorization problems, e.g.: no access to USS files, no WebSphere Administrator userid
- conversation problems, e.g.: conversation found ready for activation is not based on currently active conversation.