WebSphere® Application Server V4.0 for z/OS and OS/390

Operations and Administration

SA22-7835-01

WebSphere® Application Server V4.0 for z/OS and OS/390

Operations and Administration

SA22-7835-01

Note

Before using this information and the product it supports, be sure to read the general information under "Appendix C. Notices" on page 125.

Second Edition (June 2001)

This is a major revision of SA22–7835–00

This edition applies to WebSphere Application Server V4.0 for z/OS and OS/390 (5655-F31), and to all subsequent releases and modifications until otherwise indicated in new editions.

The most current versions of the WebSphere Application Server V4.0 for z/OS and OS/390 publications are at this Web site: `http://www.ibm.com/software/webservers/appserv/`

Contents

Figures

Tables

About this book

This book describes operations and administration procedures for WebSphere for z/OS.

Note: The full product name is ″WebSphere Application Server V4.0 for z/OS and OS/390″, hereafter referred to in this text as ″WebSphere for z/OS″ or the ″Application Server.″

Who should use this book

This book is for WebSphere for z/OS system operators and administrators. Practical experience using the Application Server, OE, RRS, and WLM is recommended, but not essential. To get familiar with the Application Server, the operator or administrator should first read the *WebSphere Application Server V4.0 for z/OS and OS/390: System Management User Interface*, SA22-7838, which describes the Administration and Operations applications that manage WebSphere for z/OS. You can also visit the WebSphere Application Server Web site at `http://www.ibm.com/software/webservers/appserv/` for related information and publications.

How this book is organized

- "Chapter 1. Introduction" on page 1, is an overview of WebSphere for z/OS operations and administration.
- "Chapter 2. Identifying where to perform WebSphere for z/OS operations" on page 5, lists common operations tasks and describes when to use the Systems Management GUI or the MVS console to perform these tasks.
- "Chapter 3. Operating WebSphere for z/OS" on page 11, describes basic Application Server operations tasks.
- "Chapter 4. Operational considerations for z/OS or OS/390 subsystems" on page 29, describes operational considerations for z/OS or OS/390 subsystems in an Application Server environment.
- "Chapter 5. WebSphere for z/OS backup guidelines and procedures" on page 33, describes Application Server backup guidelines and procedures.
- "Chapter 6. Monitoring and recovering WebSphere for z/OS and dependent subsystems" on page 37, describes guidelines for monitoring and recovering Application Server and its dependent subsystems.
- "Chapter 7. WebSphere for z/OS administration procedures" on page 53, describes Application Server administration tasks.

- “Chapter 8. WebSphere for z/OS tuning and performance monitoring” on page 79 , describes Application Server performance monitoring guidelines.
- “Chapter 9. Systems Management Facility (SMF) recording and monitoring” on page 93, describes Systems Management Facility (SMF) recording and monitoring for the Application Server.
- “Appendix A. SMF record type 120 (WebSphere for z/OS)” on page 101, describes Systems Management Facility (SMF) record type 120 for the Application Server.
- “Appendix B. Naming conventions for application servers” on page 121, describes how to establish a solid naming convention for your application servers.
- “Appendix C. Notices” on page 125, provides notices about programming interfaces, examples used in this book, and trademarks.

Where to find related information

This is a list of books that are in the WebSphere for z/OS library. They can be found at the following Web site:

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

- *WebSphere Application Server V4.0 for z/OS and OS/390: Program Directory*, GI10-0680, describes the elements of and the installation instructions for WebSphere for z/OS.
- *WebSphere Application Server V4.0 for z/OS and OS/390: License Information*, LA22-7855, describes the license information for WebSphere for z/OS.
- *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834, describes the planning, installation, and customization tasks and guidelines for WebSphere for z/OS.
- *WebSphere Application Server V4.0 for z/OS and OS/390: Messages and Diagnosis*, GA22-7837, provides diagnosis information and describes messages and codes associated with WebSphere for z/OS.
- *WebSphere Application Server V4.0 for z/OS and OS/390: Operations and Administration*, SA22-7835, describes system operations and administration tasks.
- *WebSphere Application Server V4.0 for z/OS and OS/390: Assembling J2EE Applications*, SA22-7836, describes how to develop, assemble, and install J2EE applications in a WebSphere for z/OS J2EE server. It also includes information about migrating applications from previous releases of WebSphere Application Server for OS/390, or from other WebSphere family platforms.
- *WebSphere Application Server V4.0 for z/OS and OS/390: Assembling CORBA Applications*, SA22-7848, describes how to develop, assemble, and deploy CORBA applications in a WebSphere for z/OS (MOFW) server.

- *WebSphere Application Server V4.0 for z/OS and OS/390: System Management User Interface*, SA22-7838, describes the system administration and operations tasks as provided in the Systems Management User Interface.
- *WebSphere Application Server V4.0 for z/OS and OS/390: System Management Scripting API*, SA22-7839, describes the functionality of the WebSphere for z/OS Systems Management Scripting API product.

You might also need to refer to information about other z/OS or OS/390 elements and products. All of this information is available through links at the following Internet locations:

```
http://www.ibm.com/servers/eserver/zseries/zos/
http://www.ibm.com/servers/s390/os390/
```

Here are some books that you might find particularly helpful:

- *Getting Started with WebSphere Application Server*, SC09-4581, provides an overview of WebSphere for z/OS and describes requirements for setting up the environment.
- *Building Business Solutions with WebSphere*, SC09-4432

How to send your comments

Your feedback is important in helping to provide the most accurate and highest quality information. You can e-mail your comments to:

```
wasdoc@us.ibm.com
```

or fax them to 919-254-0206.

Be sure to include the document name and number, the WebSphere Application Server version, and, if applicable, the specific page, table, or figure number on which you are commenting.

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

Summary of changes

Summary of changes
for SA22–7835–01
WebSphere for z/OS
as updated, June 2001,
service level W400018

This book contains information previously presented in SA22–7835–00, which supports WebSphere for z/OS. The following is a summary of changes to this information:

- "Chapter 2. Identifying where to perform WebSphere for z/OS operations" on page 5
- "Chapter 8. WebSphere for z/OS tuning and performance monitoring" on page 79

Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Chapter 1. Introduction

Managing information technology (IT) for the effective delivery of IT services is a complex business challenge. The demand for high system availability is critical for enterprises seeking to become business leaders.

One key success factor to control complex environments like WebSphere for z/OS is to implement robust monitoring and operations to maximize system availability and performance. Every enterprise should carefully assess its business needs, then determine what is possible with the current technology and the availability of its resources to implement that technology. Very few enterprises can afford long, planned or unscheduled outages. The need for high availability will always be required, and continuous availability will increasingly become a major competitive advantage.

Availability can mean different things to different organizations:

High availability

A system characteristic that minimizes or masks the effects of **unscheduled** outages. It attempts to keep applications running during planned service hours. It involves redundancy of components to ensure that service is always delivered, regardless of component failures. It also involves thorough testing to ensure that potential problems are detected before they affect the production environment.

Continuous operations

A system characteristic that minimizes or masks the effects of **scheduled** outages. It attempts to deliver IT services to customers without outages, planned or otherwise. This is not that difficult to achieve. There are many examples of specialized systems, such as a Communication Management Configuration, which can run for many months without any type of outage. However, this requires few or no changes to the system, which is an unrealistic scenario in an actual production system.

Continuous availability

A system characteristic that minimizes or masks the effects of **all** outages. It is the result of combining high availability and continuous operations. It means that the IT services provided by applications will remain available across scheduled and unscheduled system outages.

Overview of WebSphere for z/OS operations

The WebSphere for z/OS operations application lets you manage WebSphere for z/OS servers and server instances using the Systems Management User Interface that runs on NT. You can display the status of all server instances, stop application servers and server instances, cancel application servers and server instances, cancel and restart servers and server instances, and filter the operations window. For information on how to use the WebSphere for z/OS operations application, see the *WebSphere Application Server V4.0 for z/OS and OS/390: System Management User Interface*, SA22-7838.

This book, *WebSphere Application Server V4.0 for z/OS and OS/390: Operations and Administration*, SA22-7835, provides guidelines and procedures for operating and administering WebSphere for z/OS. It includes:

- Performing operations tasks from the z/OS or OS/390 console
- Hints and tips for managing servers
- Operations guidelines
- Tuning dependent subsystems to improve system performance
- Recovery scenarios and guidelines
- Monitoring and backup guidelines.

Overview of WebSphere for z/OS administration

The WebSphere for z/OS administration application allows you to use the Systems Management User Interface on NT to display and modify the WebSphere for z/OS applications and the environment in which they run. For information on how to use the WebSphere for z/OS administration application, see the *WebSphere Application Server V4.0 for z/OS and OS/390: System Management User Interface*, SA22-7838. Additional administration tasks and guidelines are in “Chapter 7. WebSphere for z/OS administration procedures” on page 53 and “Chapter 8. WebSphere for z/OS tuning and performance monitoring” on page 79.

Overview of required WebSphere for z/OS elements and subsystems

WebSphere for z/OS elements

The following are the required elements for a WebSphere for z/OS host system:

- WebSphere for z/OS System Server Instances:
  - Daemon
  - System Management Server (SMS)
  - Naming
  - Interface Repository (IR)

- WebSphere for z/OS Applications Server Instances:
  - Control Region (CR)
  - Server Region (SR)

The following are optional elements that can run on other WebSphere Application Server hosts (either S/390 or distributed):

- WebSphere Application Server Clients

See *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834 for more information.

Required z/OS or OS/390 subsystems

See "Chapter 6. Monitoring and recovering WebSphere for z/OS and dependent subsystems" on page 37 for information regarding the order in which subsystems must be started and stopped, and how to recover your system when a subsystem fails.

For information about the required z/OS or OS/390 subsystems, see the *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834.

Chapter 2. Identifying where to perform WebSphere for z/OS operations

| This chapter lists the main WebSphere for z/OS operations tasks and directs
| you to information that helps to perform these tasks. All Application Server
| operations can be performed from a z/OS or OS/390 MVS console. Some
| activities can also be done from the the TSO or RRS panels, and the Systems
| Management User Interface (SM/EUI) on NT. For references to the Systems
| Management User Interface, see *WebSphere Application Server V4.0 for z/OS and*
| *OS/390: System Management User Interface*, SA22-7838.

Note: The term ″application server″ may be used below to reference the Daemon, SM, Naming or IR.

Table 1. WebSphere for z/OS operations tasks

Task	**MVS Console (z/OS or OS/390)**	**SM/EUI (on NT)**	**TSO panel**	**Reference to associated procedure**
Canceling the daemon.	Yes	No	No	See “Canceling the daemon” on page 18.
Canceling an application server or server instance	Yes	Yes	No	See “Canceling application servers and server instances” on page 18.
Checking the contents of the name space	No	No	No	See “Checking the contents of the name space” on page 20.
Displaying the status of ARM registered address spaces including WebSphere for z/OS servers	Yes	No	No	See “Displaying the status of ARM-registered address spaces including WebSphere for z/OS servers” on page 20.
Displaying the status of a server or server instance	Yes	Yes	No	See “Chapter 3. Operating WebSphere for z/OS” on page 11 and *WebSphere Application Server V4.0 for z/OS and OS/390: System Management User Interface*, SA22-7838.

Table 1. WebSphere for z/OS operations tasks (continued)

Task	MVS Console (z/OS or OS/390)	SM/EUI (on NT)	TSO panel	Reference to associated procedure
Displaying units of work (threads) for DB2	Yes	No	No	See “Displaying units of work (threads) for DB2” on page 22.
Displaying in-doubt units of work (threads) for DB2	Yes	No	No	See “Displaying in-doubt units of work (threads) for DB2” on page 22.
Displaying units of work for RRS	No	No	Yes	See “Displaying units of work for RRS” on page 23. See*z/OS MVS Programming: Resource Recovery*, SA22-7616, for information on how to display units of work for RRS.
Displaying units of work for CICS	Yes	No	Yes	See “Displaying units of work for CICS” on page 23. See *CICS Operations and Utilities Guide*, SC34-5717, for details on displaying units of work for CICS.
Display units of work for IMS	Yes	No	No	See “Displaying units of work (transactions) for IMS” on page 23. Also see *IMS/ESA Summary of Operator Commands*, SC26-8766.
Hot starting the Application Server	Yes	No	No	See “Hot starting and quick starting WebSphere for z/OS” on page 19. Also see *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834.
Quick starting the Application Server	Yes	No	No	See “Hot starting and quick starting WebSphere for z/OS” on page 19. Also see *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834.
Setting up error log streams for different servers	No	You can associate a log stream with a server from the SMUI.	No	See “Setting up error log streams for different servers and server instances” on page 20. See *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834, for details on setting up error log streams.

Table 1. WebSphere for z/OS operations tasks (continued)

Task	MVS Console (z/OS or OS/390)	SM/EUI (on NT)	TSO panel	Reference to associated procedure
Setting up SMF recording	Yes	Enable it from here, but initiate it from the console.	No	See "Setting up SMF recording" on page 95. Also see *WebSphere Application Server V4.0 for z/OS and OS/390: System Management User Interface*, SA22-7838.
Shutting down the Application Server host environment	Yes	No	No	See "Shutting down the WebSphere for z/OS host environment" on page 15.
Starting the Application Server host environment	Yes	No	No	See "Starting the WebSphere for z/OS host environment" on page 12.
Starting a server or server instance	Yes	Application server only	No	See "Starting servers and server instances" on page 15 and *WebSphere Application Server V4.0 for z/OS and OS/390: System Management User Interface*, SA22-7838.
Stop a server	No	Application server only	No	See *WebSphere Application Server V4.0 for z/OS and OS/390: System Management User Interface*, SA22-7838.
Stopping a server instance	Yes	Yes	No	See "Stopping application server instances" on page 17 and *WebSphere Application Server V4.0 for z/OS and OS/390: System Management User Interface*, SA22-7838.

Table 1. WebSphere for z/OS operations tasks (continued)

Task	MVS Console (z/OS or OS/390)	SM/EUI (on NT)	TSO panel	Reference to associated procedure
Taking a WebSphere for z/OS system server out of service	Yes	Application server only; You cannot take a WebSphere for z/OS system server out of service from the SMUI (Daemon, IR, Naming, SM)	No	See "Taking a WebSphere for z/OS system server out of service" on page 19.
Workload Management				
Checking and managing the workload management application environment (display, stop/queisce, restart/resume)	Yes	No	No	See "Displaying the status of a WLM application environment" on page 24.
Getting out of the stopped state and back to the available state (Workload Management)	Yes	No	No	See "Getting out of the stopped state and back to the available state" on page 26.

Operating WebSphere for z/OS from the Systems Management User Interface

The operations application of the Systems Management User Interface allows you to perform the following tasks to operate your Application Server environment:

- Start a server or server instance.
- Stop a server or server instance.

- Cancel a server or server instance.
- Cancel and restart a server or server instance.
- Filter the operations window.

See *WebSphere Application Server V4.0 for z/OS and OS/390: System Management User Interface*, SA22-7838 for more information.

Operating WebSphere for z/OS from the MVS console

WebSphere for z/OS can be operated from the MVS console as described in this book.

It should be noted that all automations for the Application Server environment are done using interfaces from the MVS console. Products such as Netview are presented copies of messages that are to be displayed on the MVS console. These automation products can also enter commands into the system using a ″virtual″ MVS console as a source.

Chapter 3. Operating WebSphere for z/OS

This chapter describes basic WebSphere for z/OS operating procedures that you can run from the MVS console.

Before performing these tasks, please review the following terms:

Daemon

The initial point of contact within the Application Server node. It publishes a network address that other servers or clients use to make requests to the Application Server system. The daemon accepts the requests, determines which server in the node can provide the function requested, and then routes the request to the server.

z/OS or OS/390 system

A computer and its associated devices where z/OS or OS/390 and the Application Server are running.

Sysplex

A set of z/OS or OS/390 systems communicating and cooperating with each other through certain multi-system hardware components and software services to process customer workloads. A sysplex is a single-image system complex. It is two or more z/OS or OS/390 systems that, together, provide a single system complex (for example, it may be two LPARs on the same hardware). This means that while a sysplex is composed of multiple systems, it acts and reacts like a single instance.

server

A logical grouping of server instances. All server instances within a server are identical in structure and run the same set of applications. Administration is usually done at the server level. In addition, a server, from a management perspective, is a single entity in the sysplex. The server presents a single interface to the network and operator for control.

server instance

A functional component on which the Application Server applications run. It is an instance of a replicated server that can provide all the functions that the server makes available. All server instances within a server are identical in structure.

You can manage a server instance through the Systems Management User Interface Operations application or the MVS console via its unique name.

A server instance has two kinds of address spaces: a control region and one or more server regions. Application server code runs in a server region. A server region can be replicated based on the workload demands of the system. The control regions queue messages to the server regions.

Starting the WebSphere for z/OS host environment

This section describes how to startup the WebSphere for z/OS host environment. See *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834, for details about the required subsystems that must be in place before you can startup the host environment.

Steps for starting up the WebSphere for z/OS host environment

Before you begin: When you start the Daemon, you will get SMS, Naming, and IR by default. However, you will need to go to the console or through automation to start the WebSphere for z/OS host environment.

Note: This procedure also includes steps to start DB2. WebSphere for z/OS uses DB2 and requires that, in a sysplex configuration, each system that runs the Application Server has access to a data-sharing DB2 instance.

Perform the following steps to startup the Application Server host environment:

1. Start all prerequisite subsystems (see *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834).

2. Start RRS with the MVS start command:

   ```
   start atrrrs,sub=master
   ```

   Note: You must start RRS before starting DB2.

3. Start DB2 on each system.

   WebSphere for z/OS requires that a shared DB2 configuration be running on all the systems in the sysplex on which the Application Server runs. This is because the Application Server places its operational and management data in this shared DB2.

   The following example demonstrates how DB2 is started on each system in the sysplex:

Example: One DB2 is shared between all systems. However, you need to use a unique DB2 name for each subsystem.

–DB1G start DB2
: where `DB1G` is the name of the DB2 subsystem on system 1 in a 3–way sysplex.

–DB2G start DB2
: where `DB2G` is the name of the DB2 subsystem on system 2 in a 3–way sysplex.

–DB3G start DB2
: where `DB3G` is the name of the DB2 subsystem on system 3 in a 3–way sysplex.

---

4. Start the daemon.

   Because the daemon may be a single point of failure, IBM recommends that, in a two-way sysplex, you have at least two daemons. The advantage of having more than one daemon in the sysplex is availability. If one system goes down, your jobs will continue to run. The full **Application Server runtime configuration** consists of:

   - Daemon (DM)
   - System Management (SM) — (Started by the daemon)
   - Naming (NM) — (Started by the daemon)
   - Interface Repository (IR) — (Started by the daemon)

   Notes:

   a. WebSphere for z/OS servers can only run on systems where a daemon has been started.

   b. IBM recommends that customer application servers be started on each system in the sysplex.

   See *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834, for more information.

   To start the daemon on the first system in your sysplex, issue the following command:

   ```
   s bbodmn.daemon01,srvname='DAEMON01'
   ```

   where `daemon01` is the step name of the control region you want to start (daemon and server names are different on each system).

   Note: The name of the control region (`srvname='DAEMON01'`) in quotes is case-sensitive—it must be in **UPPERCASE**.

To start the daemon on the second system in your sysplex, issue the following command:

```
s bbodmn.daemon02,srvname='DAEMON02'
```

where `daemon02` is the step name of the control region you want to start.

Start the daemon on all of the images or systems in your sysplex.

Example: This is an example from the syslog of the command and responses to the start of the daemon.

```
 S BBODMN.DAEMON01
 BBOU0007I CB SERIES DAEMON DAEMON01 IS STARTING.
 START BBOSMS.SYSMGT01,SRVNAME='SYSMGT01',PARMS=''
 BBOU0001I CB SERIES CONTROL REGION SYSMGT01 IS STARTING.
 START BBONM.NAMING01,SRVNAME='NAMING01',PARMS=''
 BBOU0001I CB SERIES CONTROL REGION NAMING01 IS STARTING.
 START BBOIR.INTFRP01,SRVNAME='INTFRP01',PARMS=''
 BBOU0001I CB SERIES CONTROL REGION INTFRP01 IS STARTING.
 BBOU0016I INITIALIZATION COMPLETE FOR DAEMON DAEMON01.
 BBOU0020I INITIALIZATION COMPLETE FOR CB SERIES CONTROL REGION SYSMGT01.
 BBOU0020I INITIALIZATION COMPLETE FOR CB SERIES CONTROL REGION INTFRP01.
 BBOU0020I INITIALIZATION COMPLETE FOR CB SERIES CONTROL REGION NAMING01.
```

Note: The only command entered is the start of the daemon. The remaining start commands are generated internally.

5. Start the application servers (assuming you have already configured them).

   The identifier associated with the started task, when specified on the start command, is used by other MVS commands.

   a. Start the first application server:

      ```
      s bboasr1.bboasr1a,srvname='BBOASR1A'
      ```

      where `BBOASR1A` is the application server name.

   b. Start the second application server:

      ```
      s bboasr1.bboasr1b,srvname='BBOASR1B'
      ```

      where `BBOASR1B` is the application server name.

   c. Start the third application server:

      ```
      s bboasr1.bboasr1c,srvname='BBOASR1C'
      ```

      where `BBOASR1C` is the application server name.

You know you are done when you see an initialization complete message such as:

```
BBOU0020I INITIALIZATION COMPLETE FOR CB SERIES CONTROL REGION BBOASR1C
```

Shutting down the WebSphere for z/OS host environment

This section is the reverse of the startup procedures. However, it describes what has to be canceled first before shutting down the Application Server host environment.

Steps for shutting down the WebSphere for z/OS runtime environment

Before you begin: When you stop the daemon on one system, it doesn't bring down the servers on the other system(s). Also, you don't have to individually stop the Systems Management, Naming, and IR servers. The daemon brings down Systems Management, which in turn brings down Naming, IR, and all other control regions. You may optionally stop these servers directly. However, if you bring down Systems Management, it will bring down everything except the daemon because control regions cannot exist without Systems Management.

Perform the following steps to shutdown the WebSphere for z/OS runtime environment:

1. Stop all of your application control regions

   A **stop** can be initialized on the server from the SMUI. All server instances in the sysplex associated with this server will stop. In addition, a **stop** can be issued against each individual server instance from the MVS console. **Stop** means that all currently-running transactions are carried out before the application control region is taken down, while **cancel** means that the application control region is immediately taken down without waiting for the active transactions to complete.

2. Stop the daemon on each system in the sysplex (or cancel if it takes too long).

   This must be done from the MVS console. The daemon brings down Systems Management which, in turn, brings down Naming, IR, and all other control regions.

Note: When you stop the daemon, you may get an A03 abend and dump in your address space, but this does not impair the stop.

Starting servers and server instances

This section describes how to start servers and server instances.

Note: From the MVS console, you need to start each individual server instance that you wish to have started. The SMEUI, however, gives you the option to either do that or start a server which will start all defined server instances automatically.

When starting both servers and server instances, it helps to know if the system address space is up. To do this, you could do one of the following four things:

- Display a list of *all* address spaces:

  ```
  d a,l
  ```

- Display a list of *all active* address spaces:

  ```
  d a,a
  ```

- Display only the address space in which you are interested:

  ```
  d a,address-space-name
  ```

  (`D A,BBOASR1`, for example)

  Note: This command is recommended over the first two because it will not yield such a lengthy list on a production system. Of course, you need to know the name of the address space for which you are looking.

- Display a list of all active address spaces that start with BBO:

  ```
  D A,BBO*
  ```

You'll know the system is up if you see the address space you are looking for.

Steps for starting servers

Perform the following steps to start a server:

1. Determine if the system address space is up by issuing one of the commands above. If it's not up, go to the next step to start the server.

2. To start a server from the MVS console, you need to start each individual server instance that you wish to have started. See "Steps for starting an application server instance" on page 16.

Steps for starting an application server instance

Perform the following steps to start an application server instance:

1. Before you start any of the application server instances, you must ensure that any resource managers required by your applications are available (DB2, CICS, etc). See the related publications for details.

2. Determine if the system address space is up by issuing one of the commands above. If it's not up, go to the next step to start the server instance.

3. Before you start any of the application server instances, you must validate that any resource managers required by your applications are available.

   To start a server instance, issue the following command:

   ```
   start controlregionprocname.serverinstance,srvname='serverinstance name',parms=''
   ```

   where:

   controlregionprocname
   Is the JCL procedure name in the proclib that is used to start the server.

   .serverinstance
   Is the name of the server instance (or the step name used to start the proc). This allows you to identify the address space that is running when you view it in the SDSF panels.

   srvname
   Is used when you want to specify a server instance specifically. **This parameter is case-sensitive.**

   Note: This parameter is only optional when the server instance name that is defaulted in the JCL proc is started. Otherwise, it is required.

   'serverinstance name'
   Is the name of the specific server instance you are starting.

   parms Is used to pass parameter information to the JCL procedure. For example, ′-ORBCBI COLD′ is used to specify a cold start.

You know the server instance is up when you get the following message:

```
BBOU0020I INITIALIZATION COMPLETE FOR CBSERIES CONTROL REGION server-instance...
```

Stopping application server instances

This section describes how to stop an application server instance. When you stop a server instance, the current work is finished before the server is stopped. When you cancel a server, the current work is not completed before termination.

Steps for stopping application server instances

Before you begin: This procedure is written with the assumption that you started the server instance either:

- With the stepname specified
- From the SMUI, where stepname qualifies the server instances that it starts.

Perform the following step to stop server instances:

1. Issue the command:

   ```
   stop server-instance
   ```

   where `server-instance` is the name of the server instance.

   Note: Stopping one server instance does not effect the other server instances that make up the server. The only exception to this is that the workload will be balanced across the remaining server instances.

Canceling application servers and server instances

This section describes how to cancel application server instances that make up a server. Canceling a server instance ends it immediately, while stopping ends it after the current work has finished processing.

Steps for canceling application server instances

Before you begin: You cannot cancel a server from the MVS console. Instead, you must cancel each of the server instances that make up the server.

Perform the following step to cancel a server instance using the modify cancel command:

1. Issue the command:

   ```
   modify server-instance,cancel
   ```

Canceling the daemon

Note: Use caution with the cancel command.

This section describes how to cancel the daemon. Canceling the daemon brings down System Management which, in turn, brings down Naming, IR, and all other control regions.

Steps for canceling the daemon

Before you begin: If you cancel the daemon, it cancels all WebSphere for z/OS servers on that system.

Perform the following step to cancel the daemon:

1. Issue one of the following commands:

   ```
   cancel bbodmn.daemon01
   ```

   or

   ```
   cancel bbodmn.daemon01,norestart
   ```

Note: "`norestart`" must be used if ARM is active and you don't want it to restart the daemon.

Cold starting WebSphere for z/OS

See *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834, for the cold start procedures.

Hot starting and quick starting WebSphere for z/OS

See *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834, for the hot start and quick start procedures.

Taking a WebSphere for z/OS system server out of service

This section describes how to take a server out of service.

Steps for taking a server out of service.

Before you begin: Perform this task when you want to take an application out of service. This typically refers to a customer-written application. You should not take a system server out of service unless you are taking the entire installation down. Taking a server out of service involves stopping the server and ensuring that any automation you have in place does not cause it to be restarted until an explicit action is taken by the operator.

To take a server out of service, you need to stop all server instances on all systems for that server. It would be easiest to do this from the SMUI operations application. See *WebSphere Application Server V4.0 for z/OS and OS/390: System Management User Interface*, SA22-7838 for more information. This applies to application servers only.

Perform the following steps when you need to take a server out of service from the MVS console.

1. **Stop** the server instance (control region).

2. If stopping the server instance doesn't work, you should **cancel** the server instance (control region).

Note: Stopping or canceling the server instance (control region) should stop or cancel the server regions. If not, you must stop or cancel them as well.

ARM and restart

There are some things to remember when using ARM to restart your servers:

1. If you are ARM-enabled and you cancel or stop a server, it will restart in place or on another system.
2. If you start the daemon on a system that already has a deamon, it will terminate.
3. Every other server will come up on a dynamic port unless the configuration has a fixed port. Therefore, the fixed ports must be unique in a sysplex.

Checking the contents of the name space

You can check the contents of the name space using the Naming Dump Utility, which is described in the *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834.

Setting up error log streams for different servers and server instances

See the *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834, for instructions on how to set up the error log stream for different servers.

Displaying the status of ARM-registered address spaces including WebSphere for z/OS servers

WebSphere for z/OS ships with all control regions issuing automatic restart management (ARM) registration commands. If your installation enables ARM, you should read this section.

This section describes how to use ARM to display the status of all ARM-registered address spaces (including the address spaces of server instances) in the WebSphere for z/OS runtime environment. ARM is used to restart all address spaces that go down, if they are registered with ARM. This does not apply if the address spaces are canceled.

Each Application Server control region registers with ARM. If a control region terminates abnormally or the system fails, ARM will try to restart the failing address spaces. In doing this, ARM will ensure that dependent address spaces are grouped together and will start in the appropriate order. In general, the default ARM policy will restart the Application Server in place. If using a sysplex, see the *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834, for setup guidelines to ensure that no cross-system restarts are performed.

Steps for displaying the status of ARM-registered address spaces

Perform the following steps to use ARM to display the status of ARM registered address spaces (including the address spaces of server instances) in the WebSphere for z/OS runtime environment:

1. Initialize all servers.

2. To display all registered address spaces (including the address spaces of server instances), issue the command:

   ```
   d xcf,armstatus,detail
   ```

Displaying the status of a server or server instance

This section describes how to use ARM to display the status of a specific server or server instance in the WebSphere for z/OS runtime environment.

Steps for displaying the status of a server or server instance

Perform the following steps to use ARM to display the status of a specific server or server instance in the WebSphere for z/OS runtime environment.

Notes on automatic restart management and run-time: At run-time, be aware of the following regarding automatic restart management:

1. Once your server instances have initialized, you can display their status with regard to automatic restart management. To display all registered address spaces (including the address spaces of server instances), issue:

   ```
   d  xcf,armstatus,detail
   ```

   To display the status of a particular server instance, use the `DISPLAY` command and identify the job name. For example, to display the status of the Daemon server instance (job BBODMN), issue:

   ```
   d  xcf,armstatus,jobname=bbodmn,detail
   ```

2. If you issue `STOP`, `CANCEL`, or `MODIFY` commands against server instances, be aware of how automatic restart management behaves regarding WebSphere for z/OS server instances:

Table 2. Behavior of automatic restart management regarding WebSphere for z/OS server instances

If you issue . . .	**Automatic restart management will . . .**
`STOP` *address_space*	not restart the address space
`CANCEL` *address_space*	not restart the address space
`CANCEL` *address_space*`, ARMRESTART`	restart the address space
`MODIFY` *address_space*`,CANCEL`	not restart the address space
`MODIFY` *address_space*`,CANCEL,ARMRESTART`	restart the address space

Displaying active address spaces

This command is useful for displaying active address spaces (for example, when you want to know if DB2 is up).

Steps for displaying active address spaces

Perform the following step to display (list) all active address spaces:

1. Issue the command:

   ```
   d a,l
   ```

Displaying active replies

Displaying active replies from the MVS console allows you to observe system activity and determine if the system requires an operator response.

Steps for displaying active replies

Perform the following step to display (list) all active replies:

1. Issue the command:

   ```
   d r,r
   ```

Displaying units of work (threads) for DB2

This section describes how to display units of work (threads) for DB2.

Steps for displaying units of work (threads) for DB2

Perform the following step to display units of work (active threads) for DB2:

1. Issue the command:

   ```
   –db2 dis thread(*)
   ```

Displaying in-doubt units of work (threads) for DB2

This section describes how to display in-doubt units of work (threads) for DB2.

An in-doubt unit of work (thread) is a unit of recovery (UR or set of changes that are to be made, or not made, as a unit) in an in-doubt state when a resource manager is coordinating the processing and RRS is waiting for the coordinator to tell it whether to resolve the UR by a commit or a backout. See *z/OS MVS Programming: Resource Recovery*, SA22-7616, for more information.

Steps for displaying in-doubt units of work (threads) for DB2

Perform the following step to display units of work (active threads) for DB2:

1. Issue the command:

   ```
   –db2 dis thread(*) type(indoubt)
   ```

You know that no unresolved threads exist if you see the message:

```
DB2 No Indoubt Threads Found
```

Displaying units of work for CICS

See *CICS Operations and Utilities Guide*, SC34-5717, for more information.

Displaying units of work (transactions) for IMS

This section describes how to display units of work (transactions) for IMS.

Steps for displaying units of work (transactions) for IMS

Perform the following steps to display units of work (transactions) for IMS:

1. To display the status of a specific transaction, issue the command:
   ```
   /dis tran trans-name
   ```
2. To display the status of a specific program, issue the command:
   ```
   /dis prog program-name
   ```
3. To display the number of Message Processing Regions (MPRs) that are currently active, issue the command:
   ```
   /DISPLAY ACTIVE REGION
   ```

   For more information about IMS commands, see *IMS/ESA Summary of Operator Commands*, SC26-8766.

Displaying units of work for RRS

See *z/OS MVS Programming: Resource Recovery*, SA22-7616, to display units of work for RRS.

Using Workload Management for WebSphere for z/OS operations

This section describes general Workload Management (WLM) tasks you may use to operate WebSphere for z/OS.

Note: WLM commands are sysplex in scope. Therefore, if you quiesce an application environment, it is, in effect, on all systems in the sysplex. Keep this in mind when using WLM commands.

Displaying the status of a WLM application environment

This section shows you how to display the status of an application environment.

Note: The WLM application environment name is the same as the specific server name.

Steps for displaying the status of application environments

Perform the following step to display the status of all your application environments:

1. Issue the command:

   ```
   d, wlm,applenv=*
   ```

   Where ″*″ displays all application environments and states.

   See *z/OS MVS System Commands Summary*, SA22-7628, for more information about the display command.

Example: Here is a sample display:

```
- SY1     d wlm,applenv=*
   SY1     IWM029I  11.21.11  WLM DISPLAY 469
     APPLICATION ENVIRONMENT NAME     STATE      STATE DATA
     BBOABBOA                         AVAILABLE
     BBOASR1                          AVAILABLE
     BBOASR2                          AVAILABLE
     BBOASR3                          AVAILABLE
     BBOASR4                          AVAILABLE
     BBOASR5                          AVAILABLE
     BBOASR6                          AVAILABLE
     BBOASR7                          AVAILABLE
     BBOASR8                          AVAILABLE
     BBOASR9                          AVAILABLE
     CBINTFRP                         AVAILABLE
     CBNAMING                         AVAILABLE
     CBSYSMGT                         AVAILABLE
     PAAWYSV                          AVAILABLE
     PAAXFSV                          AVAILABLE
     PAAX1SV                          AVAILABLE
     PAAYYSV                          AVAILABLE
```

Perform the following step to display the status of a specific application environment:

1. Issue the command:

   ```
   d wlm,applenv=bboasr1
   ```

See *z/OS MVS System Commands Summary*, SA22-7628, for more information about the display command.

Example: Here is a sample display:

```
0- SY1     d wlm,applenv=bboasr1
   SY1     IWM029I  11.21.30  WLM DISPLAY 474
     APPLICATION ENVIRONMENT NAME      STATE     STATE DATA
     BBOASR1                         AVAILABLE
     ATTRIBUTES: PROC=BBOASR1S SUBSYSTEM TYPE: CB
```

The PROC in the above example is the JCL PROC that WLM uses to start the server region.

When you issue the display command, there are two things of interest for WebSphere for z/OS:

1. Whether or not the WLM application environment names match the server names.
2. The state of the WLM application environment.
3. Whether the proc associated with the application environment is the intended proc for the corresponding server region.

The most important information is the state of the application servers.

The states that may be displayed indicate the following:

available

Indicates that the everything is normal. The application servers are available.

quiesced (q)

Indicates that no server regions will start. A server will only be in the quiesced state when there is a problem with the application (e.g., it is abending). The quiesced state takes the control region out of service. New requests will continue to come in, but WLM will not start the quiesced server region.

stopped

Acts like the quiesced state but it won’t start any application server regions. Requests will come into the control region and stay there without being processed. The server gets into the stopped state if there are terminating errors in the server region. If three address spaces terminate in ten minutes, the server gets into the stopped state. There are other reasons a server may stop, such as when an operator cancels a server region. It is not a good idea to cancel server regions. To take a server out of service, you should take down the control region rather than canceling the server region

Handling workload management and server failures

During operations, if your application fails repeatedly, causing the application server regions to terminate, workload management may terminate the application environment for the application. WebSphere for z/OS issues the following message if it tries to use a failed application environment:

```
BBOU199E Unable to schedule work.  WLM application environment applenv has
         stopped.
```

You must fix the problem with your application, then restart the application environment with the `RESUME` option of the `VARY WLM` command.

Note: The application environment is sysplex in scope. This means that when WLM stops the application environment, it is stopped across the entire sysplex. When you resume, it is resumed on every system in the sysplex.

WLM shuts down the application environment because server regions are failing. The system cannot determine why, so it shuts down the environment and requests help from the operator. If it did not do this, it would continue in a failure loop, known as a *storm drain*. The failing system appears like it is performing well because the transactions are ending quickly, but they are actually failing.

Steps for checking and starting the workload management application environment

Perform these steps to check and start the workload management application environment:

1. To display the application environment, issue the command:

   ```
   d wlm,applenv=*
   ```

2. To start the application environment, issue the command:

   ```
   v wlm,applenv=environment_name,resume
   ```

   where `environment_name` is the application environment name.

A message is issued to the console stating that the application environment was resumed.

Getting out of the stopped state and back to the available state

If a server region goes down in your sysplex and you get into the stopped state, it is important to understand that the stopped state is sysplex-wide for an application environment. The state is at the server level, not the instance level. If this occurs, WLM won't be able to start additional server regions to finish the work in progress. WLM can't tell if it is a runtime error or an environment problem. Since all systems in your sysplex are identical, if you have this problem on one system, it may happen on the other system, even if you haven't seen it yet.

Steps for getting out of the stopped state and back to the available state

Perform the following steps to get a system back to the available state from the stopped state.

1. Determine why the system is in the stopped state. This could be due to any of the following:
   - A JCL error in the server region proc
   - A WebSphere for z/OS runtime bug
   - An application bug
   - Other environmental problems
   - Other MVS-related problems.

   If you are unable to determine the cause of the problem, contact the IBM Support Center.

2. Resolve the problem.

3. Resume operations by issuing the `resume` command.

   The WLM `resume` command will resume WLM, in turn starting the server regions. However, if you haven't fixed the problem, the application environment will most likely return back to the stopped state. `Resume` will get you back to the available state if you fixed the problem.

4. Issue the following command:

   ```
   D WLM,APPLENV=applenv
   ```

   to see if the application environment is active.

Chapter 4. Operational considerations for z/OS or OS/390 subsystems

This chapter describes operational considerations for z/OS or OS/390 subsystems required or recommended for WebSphere for z/OS. These are considerations for the System Programmers that are responsible for these subsystems, not instructions for installing or configuring these subsystems.

Operational considerations for z/OS or OS/390 subsystems

DB2 for z/OS or OS/390 operations

Guidelines for DB2 operations: This section provides guidelines and tips for DB2 for z/OS or OS/390 operations. See the *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834 for details on how the Application Server uses DB2.

- When you create and commit conversations, you may see a lot of DB2 activity as DB2 is offloading data. In some cases you may need to add another log volume or clean up the logs. Check the size of your DB2 logs. If the DB2 log runs out of space, the program will stop and you will need to add additional log data sets.
- When your configurations get larger, you may need to increase your default buffers. See *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834, for recommendations for increasing the buffer pools. Check the 32K temporary work space for DB2. The work space should have been allocated during the Application Server installation. If it is not large enough, however, you may get an SQLCODE -904 return code when bringing up the LDAP server, the System Administration Server, or the Naming Server.
- If you need to stop DB2 to do maintenance, you must also stop WebSphere for z/OS. The Application Server uses DB2 for its control information. Therefore, DB2 must be running to allow the Application Server runtime servers to run.
- When displaying DB2 threads, the correlation ID is equal to the MVS user ID of the requesting client.
- If you get an error code indicating that the DB2 tables have filled up a volume, the solution is to move the DB2 tables to a larger volume or, if possible, add more space to that volume.

CICS operations

Guidelines for CICS operations: This section provides guidelines and tips for CICS operations as they relate to WebSphere for z/OS.

- **Configuring the CICS region for running a sample application:** See *Configuring the CICS region for running the sample application* section in *WebSphere Application Server V4.0 for z/OS and OS/390: Assembling J2EE Applications*, SA22-7836, for details on configuring CICS. You need to ensure that the RRMS attribute is set to "Yes" and that the NETNAME is the name of the WebSphere for z/OS server under which you will run the CICS adapter.

IMS operations

This section provides guidelines and tips for IMS operations as they relate to the Application Server. Also see *IMS/ESA Operations Guide*, SC26-8741, for IMS operations guidelines as they relate to WebSphere for z/OS.

- When using IMS, you need to set up a significant number of message processing regions to handle the total number of IMS transactions that might be issued in a WebSphere for z/OS transaction. One Application Server transaction could drive three or more transactions to IMS. To successfully process these transactions, IMS may need additional message processing regions available to handle the request. In general, IMS needs the same number of started message processing regions as the number of generated IMS transactions. See *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834, and *WebSphere Application Server V4.0 for z/OS and OS/390: Assembling J2EE Applications*, SA22-7836, for more information.
- You may also need to set the following parlim if it wasn't set at installation time:

  ```
  assign parlim 0 tran tranname
  ```

  Assigning a parallel limit value of "0" indicates that there is no limit on the number of transactions you can run at one time. This can also be specified at IMS generation during configuration time using the TRANSACT statement. This value allows transactions to be scheduled in multiplicity (or parallel) so that more transactions can run at the same time. You need to set this value in addition to setting up multiple message processing regions to make IMS work properly. See *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834, for more information.
- **IMS OTMA support**: When defining the logical resource manager for a server, you must select IMS_OTMA_PAA. The IMS control region has to be started with the OTMA interfaces active. The IMS procedure parameter should specify OTMA=YES for the Application Server to make the connection to the right IMS. You also need to define the XCF group name for the OTMA interfaces that IBM supports. See the instructions in *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834, for details about setting up the IMS-OTMA Procedural Application Adapter.
- **IMS OTMA support runtime tip:** If you have to re-IPL your system, you must redefine the IMS OTMA SVC (supervisor call). You can do this by

running the exec statement `PGM=DFSYSVI0`, which dynamically installs the SVC needed. If you don't redefine it, when you try to run IMS OTMA support, you will get an F92 abend and the server will come down.

RRS operations

See Table 6 on page 40 and *z/OS MVS Programming: Resource Recovery*, SA22-7616, for RRS operations guidelines.

Tips for RRS operations:

- If you have configured your logstreams to the coupling facility, then monitor your log streams to ensure offload is not occurring. RRS will perform better if its recovery logs do not offload.

  Note: Proper sizing of the RRS logs is important. Too small and you get reduced throughput since logger is offloading the logs too frequently. Too large and you could overflow your coupling facility.
- Keep the main and delayed (only contains active or live data) logs in your coupling facility. Make sure the CF definitions don't overflow.

  Note: A commit cannot occur until the log record is written.
- Until you stabilize your workloads, it is a good idea to use the archive log. If you have an archive log configured, RRS will unconditionally use it. However, there is a performance penalty for using it.

Workload Management (WLM) operations

See *z/OS MVS Planning: Workload Management* and *z/OS MVS Programming: Workload Management Services*, SA22-7619, for WLM operations guidelines.

Chapter 5. WebSphere for z/OS backup guidelines and procedures

This chapter describes WebSphere for z/OS backup guidelines and procedures.

Backing up the OS/390 runtime environment

Guidelines for backup of the WebSphere for z/OS system

Use the following guidelines to back up parts of your WebSphere for z/OS system:

1. Be sure to back up the RMDATA log for RRS. Otherwise, a failure could force you to do a cold start of RRS.
2. Keep the ARCHIVE log retention period to one day.
3. Follow your own backup procedures to back up the LDAP database that contains naming and interface repository data.

   Note: If you restore LDAP data, be sure to coordinate the restoration with:
   - Other WebSphere systems in the federated naming space (otherwise, your naming space will not be consistent).
   - The System Management database (the SM tables need to be restored at the same point in time as the LDAP tables).
4. Incorporate the following in your normal backup procedures:
   - WebSphere for z/OS proclibs
   - WebSphere for z/OS loadlibs
   - WebSphere for z/OS environment files
   - The directory where applications are written by the Administration application (the value of the CBCONFIG environment variable; the default is `/WebSphere390/CB390`).
5. Back up reference collection data in these DB2 for z/OS or OS/390 tables:
   - BBO.RCTABLE
   - BBO.KRCTABLE
   - BBO.RCHMTABLE
6. Back up your own application executables and bindings.
7. When you activate a conversation, System Management automatically backs up the current environment files for each server instance in `/`*`path`*`/controlinfo/envfile/`*`sysplex`*`/`*`server_instance`*`/backup/`, where

path
Is the value of the CBCONFIG environment variable (default is `/WebSphere390/CB390`).

sysplex
Is the name of your sysplex.

server_instance
Is the name of the server instance.

The backup files have a time stamp in their names. You may wish to erase the older backup files as the backup directory fills up.

8. When you prepare for a cold start, System Management backs up control information in XML format in `/`*`path`*`/configuration/backup/`, where

   path
   Is the value of the CBCONFIG environment variable (default is `/WebSphere390/CB390`).

   The backup files have a time stamp in their names. You may wish to erase older backup files as the backup directory fills up.
9. If you wish to back up a single server instance, you can use the export/import function in the Administration application. For details on how to do this, see *WebSphere Application Server V4.0 for z/OS and OS/390: Assembling J2EE Applications*, SA22-7836.
10. Regarding the system management database, decide what to back up by following this table:

If you have . . .	Then back up . . .
Added an administrator	Table spaces: BBOMDB01.BBOMS51 BBOMDB01.BBOMS54

If you have . . .	Then back up . . .
Created a new conversation or committed a conversation	Table spaces: BBOMDB01.BBOMS00 BBOMDB01.BBOMS02 BBOMDB01.BBOMS04 BBOMDB01.BBOMS06 BBOMDB01.BBOMS10 BBOMDB01.BBOMS15 BBOMDB01.BBOMS19 BBOMDB01.BBOMS23 BBOMDB01.BBOMS25 BBOMDB01.BBOMS27 BBOMDB01.BBOMS29 BBOMDB01.BBOMS31 BBOMDB01.BBOMS33 BBOMDB01.BBOMS35 BBOMDB01.BBOMS37 BBOMDB01.BBOMS39 BBOMDB01.BBOMS41 BBOMDB01.BBOMS43 BBOMDB01.BBOMS45 BBOMDB01.BBOMS48 BBOMDB01.BBOMS52 BBOMDB01.BBOMS56 BBOMDB01.BBOMS58 BBOMDB01.BBOMS60 BBOMDB01.BBOMS62 BBOMDB01.BBOMS64 BBOMDB01.BBOMS66 BBOMDB01.BBOMS68 BBOMDB01.BBOMS70 BBOMDB01.BBOMS72 BBOMDB01.BBOMS74 BBOMDB01.BBOMS76 BBOMDB01.BBOMS80 BBOMDB01.BBOMS81 BBOMDB01.BBOMS82 BBOMDB01.BBOMS83 BBOMDB01.BBOMS84 BBOMDB01.BBOMS85 BBOMDB01.BBOMS86 BBOMDB01.BBOMS87 BBOMDB01.BBOMS90

If you have . . .	Then back up . . .	
Activated a conversation	Table spaces/database:	
	BBOMDB01.BBOMS00	BBOMDB01.BBOMS53
	BBOMDB01.BBOMS02	BBOMDB01.BBOMS55
	BBOMDB01.BBOMS04	BBOMDB01.BBOMS56
	BBOMDB01.BBOMS06	BBOMDB01.BBOMS58
	BBOMDB01.BBOMS10	BBOMDB01.BBOMS60
	BBOMDB01.BBOMS15	BBOMDB01.BBOMS62
	BBOMDB01.BBOMS19	BBOMDB01.BBOMS64
	BBOMDB01.BBOMS23	BBOMDB01.BBOMS66
	BBOMDB01.BBOMS25	BBOMDB01.BBOMS68
	BBOMDB01.BBOMS27	BBOMDB01.BBOMS70
	BBOMDB01.BBOMS29	BBOMDB01.BBOMS72
	BBOMDB01.BBOMS31	BBOMDB01.BBOMS74
	BBOMDB01.BBOMS33	BBOMDB01.BBOMS76
	BBOMDB01.BBOMS35	BBOMDB01.BBOMS80
	BBOMDB01.BBOMS37	BBOMDB01.BBOMS81
	BBOMDB01.BBOMS39	BBOMDB01.BBOMS82
	BBOMDB01.BBOMS41	BBOMDB01.BBOMS83
	BBOMDB01.BBOMS43	BBOMDB01.BBOMS84
	BBOMDB01.BBOMS45	BBOMDB01.BBOMS85
	BBOMDB01.BBOMS48	BBOMDB01.BBOMS86
	BBOMDB01.BBOMS52	BBOMDB01.BBOMS87
		BBOMDB01.BBOMS90
	LDAP Database	BBOMDB01.BBOSLS01
		BBOMDB01.BBOSLS02

Notes:

a. Coordinate your backup of WebSphere for z/OS table spaces with other WebSphere system managers, such as those on Windows NT.

b. If you have federated the naming tree with another system, such as Windows NT, you must synchronize your backup of the LDAP database with the backup on Windows NT. Otherwise, your federated naming space will not be consistent.

Chapter 6. Monitoring and recovering WebSphere for z/OS and dependent subsystems

This chapter describes how to monitor and recover WebSphere for z/OS and its dependent subsystems.

Startup order for WebSphere for z/OS and dependent subsystems

The following table shows the order in which you need to bring up the dependent subsystems for WebSphere for z/OS. When subsystems are shown on the same line, it indicates they can be started at the same time or in any order. See *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834, for more information about the order in which you need to start dependent z/OS or OS/390 subsystems.

Startup order	**subsystem**	**subsystem**	**subsystem**
1	Workload Manager (WLM) *automatically started*		
2	RACF		
3	System Logger *automatically started*		
4	RRS	USS (doesn't complete until TCP/IP is done, but can be started before TCP/IP)	
5	VTAM	TCP/IP	
6	APPC	TSO	
7	DB2	CICS	IMS
8	LDAP	NFS (only required in a SYSPLEX) **Note:** NFS is not required at all if using a shared HFS.	

Startup order	subsystem	subsystem	subsystem
9	WebSphere for z/OS - JVM: Started inside the Application Server and WebServer - LE: Started inside the Application Server and WebServer	WebServer (Servlet)	

Automation and recovery scenarios and guidelines

The following section provides information on how to monitor and recover WebSphere for z/OS and the subsystems it uses. It provides startup, shutdown, and recovery procedures and scenarios. It also tells you how to determine if the subsystems are up or down, and tells you where to find more information.

APPC automation and recovery scenarios

Table 3. APPC automation and recovery scenarios

Task	APPC automation and recovery scenarios
Startup	APPC should be started before WebSphere for z/OS. In theory, the Application Server *could* be started before APPC, but only as long as no objects get dispatched in containers that have an IMS APPC LRMI associated with them. If APPC is not up before the Application Server, and you want to use an APPC connector to talk to IMS, then you will have no connectivity. APPC/MVS does not have to be up for CICS. APPC does not have to be started after VTAM.
Shutdown	Reverse the startup procedure. Shutdown the Application Server, APPC, then VTAM.
Handling in-flight or in-doubt transactions if there is a failure	If you are using APPC for communications and it fails, do the following: 1. Shutdown all servers with APPC connectivity. 2. Restart APPC (if it totally failed). 3. Restart the WebSphere for z/OS server. **Note:** APPC resyncs itself. If your transaction is in-doubt, IMS sits until you restart APPC. IMS relies on RRS for recovery. RRS will resolve in-doubts by handshaking with every subsystem it was communicating with before it went down. If you are using CICS, note that CICS has its own coordinator.

Table 3. APPC automation and recovery scenarios (continued)

Task	APPC automation and recovery scenarios
How to determine if APPC is running	Issue the `DISPLAY APPC,LU,ALL` command. If APPC is not active, it will say so. In addition, the status of the LUs used by the Application Server and/or IMS should be active or no APPC work will be successful.
What happens to the Application Server if APPC goes down?	Any objects attempting to use the IMS APPC PAA will not work. The server region running on behalf of the container attempting to use APPC will likely get a `C9C24C05` error, indicating that an APPC ALLOCATE request was attempted and failed. Additional APPC error diagnostic information that helps to pinpoint the APPC problem is contained in the logs associated with this region.
Where to find more information	• *z/OS MVS Planning: Operations*, SA22-7601 • *z/OS MVS Planning: APPC/MVS Management*, SA22-7599 • *z/OS MVS Programming: Resource Recovery*, SA22-7616

WLM automation and recovery scenarios

Table 4. Workload Manager (WLM) automation and recovery scenarios

Task	WLM automation and recovery scenarios
Startup	WLM is automatically started by z/OS or OS/390 when you IPL your system. You don’t have to start it.
Shutdown	You cannot shutdown WLM.
How to handle a catostrophic failure of the WebSphere for z/OS Server Regions	Following a catastrophic failure of the WebSphere for z/OS server regions, you can use the following WLM resume command: `V WLM,APPLENV=XYZ,RESUME`
Where to find more information	• *z/OS MVS Planning: Workload Management*, SA22-7602 • *z/OS MVS Programming: Workload Management Services*, SA22-7619

RACF automation and recovery scenarios

Table 5. RACF automation and recovery scenarios

Task	RACF automation and recovery scenarios
Startup	If it is installed, RACF is started as a part of IPL.
Shutdown	RACF is not shutdown.
How to determine if RACF is running	Use the `RACF SETROPTS` command to display the status of RACF.

Table 5. RACF automation and recovery scenarios (continued)

Task	RACF automation and recovery scenarios
What happens to the Application Server if RACF goes down?	RACF goes into fail safe mode. This means that for every resource that is accessed, the operator is asked to verify if it is okay. In general, the system is IPLed if this occurs.
What happens to other subsystems if RACF goes down?	It depends on what subsystem and how RACF fails.
Where to find more information	• *z/OS SecureWay Security Server RACF System Programmer's Guide*, SA22-7681 • *z/OS SecureWay Security Server RACF Security Administrator's Guide*, SA22-7683

RRS automation and recovery scenarios

Table 6. RRS automation and recovery scenarios

Task	RRS automation and recovery scenarios
Startup	Ensure System Logger has been started before RRS. **Note:** RRS will display error messages indicating that System Logger must be started first if you try to start RRS without starting System Logger.Ensure RRS is started before WebSphere for z/OS. RRS does not start by itself. RRS will start automatically only if it was registered with the Automatic Restart Manager (ARM) and if ARM is running. To start RRS, issue the `start` command: `start rrs` **Note:** RRS doesn't restart itself if you issue the cancel command, so you need to restart it manually if it was canceled or if ARM isn't running.
Shutdown	Shutdown RRS in the reverse order that you started RRS. Shutdown the Application Server, then RRS, followed by System Logger. There is no controlled way to bring down RRS. The best approach is: 1. Quiesce the Application Server. 2. Shutdown the Application Server. 3. Cancel RRS. **Note:** You may want to bring down the DB2 you are using for WebSphere for z/OS before canceling RRS. To cancel RRS, issue the command: `setrrs cancel`

Table 6. RRS automation and recovery scenarios (continued)

Task	RRS automation and recovery scenarios
Handling in-flight and in-doubt transactions if there is a failure	Refer to the RRS system management panels to display in-flight and resolve in-doubt transactions. You can display the resource managers on the RM panels in RRS, display all units of recovery (UR), filter the URs, and then resolve the in-doubts. You cannot resolve in-flights. You can display all RRS-managed transactions.
How to determine if RRS is running	Use the display command: `d a,atrrs` `atrrs` is the name of the default RRS proc shipped with the Application Server. Use the procname that you use to start RRS. The address space comes from the proc.
What happens to the Application Server if RRS goes down?	If RRS goes down, the Application Server hangs until RRS is restarted. Because RRS is part of the base operating system, RRS is resilient, with built-in fault tolerance. Operators do not need to be overly concerned about frequent failures.
What happens to other subsystems if RRS goes down?	RRS is the z/OS or OS/390 transaction monitor. If you cancel RRS, you will have problems with any subsystems using it (for example, WebSphere for z/OS, DB2, IMS). Ensure you understand the implications before you cancel RRS.
Where to find more information	• *z/OS MVS Programming: Resource Recovery*, SA22-7616

USS automation and recovery scenarios

Table 7. UNIX System Services (USS) automation and recovery scenarios

Task	USS automation and recovery scenarios
Startup	USS is a permanent component of the BCP and is started automatically at IPL time.
Shutdown	USS does not support a shutdown capability, so it is always available.
Handling in-flight or in-doubt transactions if there is a failure	The only data that could be considered transactional in nature is data stored in the HFS.
How to determine if USS is running	USS is always available.
What happens to the Application Server if USS goes down?	If USS fails, the system must be re-IPLed. the Application Server will get errors and terminate.

Table 7. UNIX System Services (USS) automation and recovery scenarios (continued)

Task	USS automation and recovery scenarios
What happens to other subsystems if USS goes down?	If USS fails, the system must be re-IPLed.
Where to find more information	• *z/OS UNIX System Services Planning*, GA22-7800

TCP/IP automation and recovery scenarios

Table 8. TCP/IP automation and recovery scenarios

Task	TCP/IP automation and recovery scenarios
Startup	TCP/IP must be up before starting WebSphere for z/OS.
Shutdown	Shutdown the Application Server before shutting down TCP/IP.
Handling in-flight or in-doubt transactions if there is a failure	Methods in flight will have their transactions rolled back when the attempt to send a response to the method fails. Other transactions will wait for a timeout.
How to determine if TCP/IP is running	Use the display command looking for the TCP/IP proc.
What happens to the Application Server if TCP/IP goes down?	If TCP/IP goes down, then the Application Server on the system must be restarted. You will get an SVC dump because the socket layer was destroyed.
What happens to other subsystems if TCP/IP goes down?	If TCP/IP goes down, sessions break and transactions react as described above. **Note:** The Application Server, which cannot recognize when TCP/IP comes back up, must be restarted.

DB2 automation and recovery scenarios

Table 9. DB2 automation and recovery scenarios

Task	DB2 automation and recovery scenarios
Startup	DB2 is started after RRS but before LDAP, NFS, and WebSphere for z/OS.
Shutdown	Reverse of startup sequence.

Table 9. DB2 automation and recovery scenarios (continued)

Task	DB2 automation and recovery scenarios

Handling in-flight or in-doubt transactions if there is a failure	Use the RRS panels to resolve. See *z/OS MVS Programming: Resource Recovery*, SA22-7616. The RRS panels are the preferred way to resolve DB2 in-doubts because they allow you to view all resource managers that have an interest in the transaction. However, you can also use DB2 to resolve in-doubts. You can issue the command:
	`DISPLAY THREAD(*) TYPE(INDOUBT)`
	to display DB2 information about the in-doubt threads it knows about (if there are too many, you can go into S.LOG to view the information). This display will give you a DB2 identifier called a "nid". Copy the nid and paste it into this command:
	`-RECOVER INDOUBT (RRSAF) ACTION(COMMIT)`
	`NID(B1D379D17ED6CF900000009401010000)`
	where the `nid` is the one that you cut from the display command. You can issue this command to roll back the transaction:
	`-RECOVER INDOUBT (RRSAF) ACTION(ABORT)`
	`NID(B1D379D17ED6CF900000009401010000)`

How to determine if DB2 is running	Use the display command to display the DB2 address space.
Where to find more information	• See the DB2 books under "Where to find related information" on page xii.

CICS automation and recovery scenarios

Table 10. CICS automation and recovery scenarios

Task	CICS automation and recovery scenarios
Startup	CICS needs to be properly installed, initialized, and started before any workflows to a CICS-enabled WebSphere for z/OS application control server region are run.
Shutdown	Shutdown the WebSphere for z/OS application control region that uses CICS as a backing store, then shutdown the CICS service.

Table 10. CICS automation and recovery scenarios (continued)

Task	CICS automation and recovery scenarios
Handling in-flight or in-doubt transactions if there is a failure	If there is an error during processing, both CICS and the Application Server rely on the underlying RRS subsystem to handle all rollback notifications to the registered interests. In the case of in-flight transactions, RRS will notify all participants that a rollback is required, and normal rollback processing will occur in each registered party. In the case of in-doubt transactions, it may be necessary to recycle the WebSphere for z/OS Application Control/Server region to release any pending transaction in CICS.
How to determine if CICS is running	This is installation dependent.
What happens to CICS if the Application Server goes down?	Should the Application Server happen to go down, one of two situations could occur: 1. If the Application Server and CICS are currently engaged in a unit of work, then RRS processing as described above would occur and it may be necessary to recycle the application control server regions to release pending transactional work in CICS. 2. If the Application Server and CICS are not currently engaged in a unit of work, CICS is not affected.
What happens to other subsystems if CICS goes down?	Not applicable
Where to find more information	• *CICS Operations and Utilities Guide*, SC34-5717

IMS automation and recovery scenarios

Table 11. IMS automation and recovery scenarios

Task	IMS automation and recovery scenarios
Startup	IMS needs to be properly installed, initialized, and started before any workflows to an IMS-enabled WebSphere for z/OS application control server region are run.
Shutdown	Shutdown the WebSphere for z/OS application Control Region which uses IMS as a backing store, then shutdown the IMS service

Table 11. IMS automation and recovery scenarios (continued)

Task	IMS automation and recovery scenarios
Handling in-flight or in-doubt transactions if there is a failure	If there is an error during processing, both IMS and the Application Server rely on the underlying RRS subsystem to handle all rollback notifications to the registered interests. In the case of in-flight transactions, RRS will notify all participants that a rollback is required and normal rollback processing will occur in each registered party. In the case of in-doubt transactions, it may be necessary to recycle the WebSphere for z/OS Application Control/Server region to release any pending transaction in the IMS MPRs.
How to determine if IMS is running	This is installation-dependent.
What happens to IMS if the Application Server goes down?	Should the Application Server happen to go down, one of two situations could occur: 1. If the Application Server and IMS are currently engaged in a unit of work, then RRS processing as described above would occur and it may be necessary to recycle the application control server regions to release pending transactional work in the IMS MPR. 2. If the Application Server and IMS are not currently engaged in a unit of work, IMS is not affected.
What happens to other subsystems if IMS goes down?	Not applicable
Where to find more information	• *IMS/ESA Operator's Reference*, SC26-8742

LDAP automation and recovery scenarios

Table 12. LDAP automation and recovery scenarios

Task	LDAP automation and recovery scenarios
Startup	LDAP, as used by WebSphere for z/OS, is completely run within the Application Server address spaces using something called "the local backend." This support takes the front side of the LDAP client APIs and the backside database implementation and runs them completely inside the Application Server Naming Server and Interface Repository. For Naming and IR, OMVS and DB2 must be up *before* Naming and IR. To run the LDAP server, TCPIP, OMVS, and DB2 must all be up before the LDAP server. **Note:** There are two LDAP modes supported: 1. Local LDAP backend. 2. Remote LDAP Server: the Application Server environment has to be set up correspondingly, and DB2, TCP/IP, and the remote server have to be up and running before WebSphere for z/OS is started.
Shutdown	Shutdown Naming and IR, then OMVS and DB2. For the LDAP server, shutdown the LDAP server, then TCPIP and DB2, and then OMVS.
Handling in-flight or in-doubt transactions if there is a failure	If there is a failure during processing, Naming and IR rely on RRS to issue a rollback directly to DB2 and, as a result, any work done by the LDAP code is rolled back along with it. For the LDAP server, AUTOCOMMIT is set to NO, causing any error to ROLLBACK for that transaction. This ensures the atomicity characteristic of LDAP operations.
How to determine if LDAP is running	In the case of WebSphere for z/OS, if Naming and IR are operating, then LDAP is operating. In the case of the LDAP server, and a started task is used for the LDAP server, use the SDSF to see if the started task is running. Examine the output log for the started task to see if any error messages were displayed. Alternatively, the `LDAPSRCH` command (from TSO), or `LDAPSEARCH` command (from USS shell) can be used to perform a simple search to verify that the LDAP server is running.
What happens to the Application Server if LDAP goes down?	• In MOFW Application Server regions, LDAP runs within the Application Server address space, so this is not an issue. If the server goes down, then LDAP also goes down. • In J2EE server regions, the LDAP server **must** be active since it is a separate server that no longer runs inside the Application Server region.

Table 12. LDAP automation and recovery scenarios (continued)

Task	LDAP automation and recovery scenarios
What happens to other subsystems if LDAP goes down?	Most z/OS or OS/390 subsystems do not depend on LDAP, but this may change in the future. In the case of accessing LDAP through the LDAP server, there is a way to configure the LDAP server to operate in a sysplex environment such that (using sysplex-enabled DNS) LDAP requests will be sent to the LDAP server in the sysplex that is operating (assuming that there is one). As an alternative, subsystems that want to use LDAP could configure a backup LDAP server to be contacted in case the primary server is not accessible. In this case, the application would assume that it could retrieve all of the same data that it could get from the backup on the primary which would be handled by some replication mechanism. The LDAP server currently supports a master/slave replication mechanism, but you could also try duplicating the sysplex server using DB2 data sharing.
Where to find more information	• For Naming and IR, see the WebSphere for z/OS books. • For LDAP Server, see *z/OS SecureWay Security Server LDAP Server Administration and Use*, SC24-5923.

NFS automation and recovery scenarios

Note: NFS is used as a shared file system in OS/390 R8. Shared HFS is used in OS/390 R9 and above. The following comments relate to the runtime usage of NFS, not the application development time usage.

Table 13. NFS automation and recovery scenarios

Task	NFS automation and recovery scenarios
Startup	During USS filesystem initialization, NFS Client is started and run in an NFS colony address space. The FILESYSTYPE parmlib statement for NFS Client must be present in the SYS1.PARMLIB(BPXPRMxx) member.
Shutdown	To stop the NFS Client gracefully, the system operator could issue the modify operator command `STOP` with the OS/390 NFS Client address space name. If the `STOP` command fails to gracefully shut down the NFS Client, the operator could force an abnormal termination by issuing the `CANCEL` command.
How to determine if NSF is running	Run the nfsstat utility in directory `/usr/lpp/NFS`.
What happens to the Application Server if NSF goes down?	New server starts will fail. An attempt to access an environment variable will fail.

Table 13. NFS automation and recovery scenarios (continued)

Task	NFS automation and recovery scenarios
What happens to other subsystems if NSF goes down?	Other subsystems should continue to work fine.
Where to find more information	• See the *OS/390 NFS User's Guide*, SC26-7254, and *OS/390 NFS Customization and Operation*, SC26-7253.

WebSphere for z/OS (daemon) automation and recovery scenarios

Table 14. WebSphere for z/OS automation and recovery scenarios

Task	WebSphere for z/OS (daemon) automation and recovery scenarios
Startup	Refer to ″Starting up the WebSphere Application Server runtime environment.″
Shutdown	Refer to ″Shutting down the WebSphere Application Server runtime environment.″
Handling in-flight or in-doubt transactions if there is a failure	The daemon is a location agent. If the daemon fails during the course of a transaction, locate requests to the daemon will fail. These request failures will be surfaced by the client ORB. If the client is a WebSphere for z/OS client running in a sysplex, the locate request will be routed to another available daemon in the sysplex, if present.
How to determine if the Application Server is running	Use the MVS display command.
What happens to other subsystems if the Application Server goes down?	Other subsystems will continue to work fine. If the Application Server daemon goes down, all Application Server servers started on the same system as the terminating daemon will also be terminated. As a general rule, if the daemon goes down and there is another one in the sysplex, clients won't be affected.
Where to find more information	• See the WebSphere for z/OS books listed in "Where to find related information" on page xii.

Naming automation and recovery scenarios

Table 15. Naming automation and recovery scenarios

Task	Naming automation and recovery scenarios
Startup	The Naming control region is started automatically (if you set it up as recommended in *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834) and implicitly when the Application Server is started. Naming server regions (that actually do the work) are started on demand by the server when work has to be done.
Shutdown	When the Application Server is stopped, the Naming control region and all active server regions are automatically stopped.
Handling in-flight or in-doubt transactions if there is a failure	If there is a failure during processing, Naming relies on RRS to issue a rollback directly to LDAP and DB2. As a result, any work is rolled back along with it.
How to determine if Naming is running	If the Application Server is running, Naming is running. If Naming is down, WebSphere for z/OS is unusable. In addition, the Naming task can be monitored using SDSF.
What happens to other subsystems if the Application Server goes down?	Naming server region failures are recovered by WLM. WLM just starts a new server region. This doesn't have any impact on other subsystems. If the Naming control region drops and isn't restarted by ARM, WebSphere for z/OS is unusable.
Where to find more information	• See the WebSphere for z/OS books listed in "Where to find related information" on page xii.
Note: In a sysplex, you only need one Naming server. Therefore, if Naming comes down on one system, you can keep running as long as one Naming is running somewhere in the sysplex.	

Interface Repository automation and recovery scenarios

Table 16. Interface Repository (IR) automation and recovery scenarios

Task	Interface Repository automation and recovery scenarios
Startup	The IR control region is started automatically (if you set it up as recommended in *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834) and implicitly when the Application Server is started. IR server regions (that actually do the work) are started on demand by the server when work has to be done.
Shutdown	When the Application Server is stopped, the IR control region and all active server regions are automatically stopped.
Handling in-flight or in-doubt transactions if there is a failure	If there is a failure during processing, IR relies on RRS to issue a rollback directly to LDAP and DB2, and, as a result, any work is rolled back along with it.

Table 16. Interface Repository (IR) automation and recovery scenarios (continued)

Task	Interface Repository automation and recovery scenarios
How to determine if IR is running	The IR server can be monitored using SDSF.
What happens to other subsystems if the Application Server goes down?	IR server region failures are recovered by WLM. WLM just starts a new server region. This doesn't have any impact on other subsystems.
Where to find more information	• See the WebSphere for z/OS books listed in "Where to find related information" on page xii.
Note: In a sysplex, you only need one IR server. Therefore, if IR comes down on one system, you can keep running as long as one IR is running somewhere in the sysplex.	

Systems Management (SM) automation and recovery scenarios

Table 17. Systems Management (SM) automation and recovery scenarios

Task	Systems Management (SM) automation and recovery scenarios
Startup	Systems Management is a WebSphere for z/OS server that is started automatically during Application Server daemon startup. As a WebSphere for z/OS server, it is a prerequisite to the Application Server infrastructure (DB2, RRS, OMVS, LDAP, WLM, etc.).
Shutdown	The Systems Management server is shutdown automatically when the daemon is shutdown.
Handling in-flight or in-doubt transactions if there is a failure	Systems Management lets ORB+OTS handle its transactions. Each request that is routed to Systems Management implicitly starts a transaction (this is handled by ORB/OTS). If something goes wrong, either Systems Management explicitly requires a rollback from OTS or the rollback is done automatically by the ORB. ORB and OTS rely on RRS to manage commits and rollbacks.
How to determine if Systems Management is running	Check whether the Systems Management control region is operating. Use SDSF.
What happens to other subsystems if the Application Server goes down?	The Application Server cannot operate if Systems Management fails. There is no impact to other subsystems.

Table 17. Systems Management (SM) automation and recovery scenarios (continued)

Task	Systems Management (SM) automation and recovery scenarios
Where to find more information	• See the WebSphere for z/OS books listed in "Where to find related information" on page xii, particularly: – *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834 – *WebSphere Application Server V4.0 for z/OS and OS/390: System Management User Interface*, SA22-7838

WebServer automation and recovery scenarios

Table 18. WebServer (Servlet) automation and recovery scenarios

Task	WebServer automation and recovery scenarios
Startup	The WebSphere for z/OS product and infrastructure do not require a WebServer or Application Server. WebServers and the Application Server Standard Edition runtime have a relationship with the Application Server only in the sense that a client application program that is written to use WebSphere for z/OS facilities may be written as a servlet. Any implications for ordering of startup will be introduced by the applications. You would probably want to have the Application Server Object servers up and ready before starting up the Web server hosting a client application.
Shutdown	Therefore, there are no dependencies from the product code. Similar to most applications, you may want to quiesce the clients prior to taking down the target WebSphere for z/OS servers.
Handling in-flight or in-doubt transactions if there is a failure	This is a statement of the WebSphere for z/OS client ORB capability. There are no requirements where, in a failure, a client needs to be restarted. OTS takes cares of this through timeout/broken connection, and presumed abort.
How to determine if WebServer is running	Use display commands, SMF records, and viewer tools (SDDF) to monitor the Application Server.
What happens to the Application Server if WebServer goes down?	Nothing; the application must adjust.
What happens to other subsystems if WebServer goes down?	Nothing.
Where to find more information	• See *Application Server Planning, Installing, and Using for OS/390*, GC34–4757.

Chapter 7. WebSphere for z/OS administration procedures

This chapter describes WebSphere for z/OS administration tasks and guidelines.

For further information, please see:

- *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834 for information on setting up RACF and DCE system security and userids.
- *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834, for updating DNS definitions/ setting up your TCP/IP network, and updating your hosts file as you expand your sysplex.
- *WebSphere Application Server V4.0 for z/OS and OS/390: Assembling J2EE Applications*, SA22-7836, for information on setting up userids, deploying applications in a Parallel sysplex Environment, and sysplex-wide installation of Application Server applications.

SSL security administration

Setting up SSL security for WebSphere for z/OS

This topic assumes you understand the SSL protocol and how Cryptographic Services System SSL works on OS/390. For information about the SSL protocol, go to the following web site:

```
http://home.netscape.com/eng/ssl3/ssl-toc.html
```

For more information about Cryptographic Services System SSL, see *z/OS System Secure Sockets Layer Programming*, SC24-5901.

If you want the added security of protected communications and user authentication in a network, you can use Secure Sockets Layer (SSL) security. The SSL support in WebSphere for z/OS has several objectives:

- To provide ways accepted by the industry to protect the security of messages as they flow across the network. This is often called *transport layer security*. Transport layer security is a function that provides privacy and data integrity between two communicating applications. The protection occurs in a layer of software on top of the base transport protocol (for example, on top of TCP/IP).

  SSL provides security over the communications link through encryption technology, ensuring the integrity of messages in a network. Because communications are encrypted between two parties, a third party cannot tamper with messages. SSL also provides confidentiality (ensuring the message content cannot be read), replay detection, and out-of-sequence detection.

- To provide a secure communications medium through which various authentication protocols may operate. A single SSL session can carry multiple authentication protocols, that is, methods to prove the identities of the parties communicating.

  SSL support always provides a mechanism by which the server proves its identity. The SSL support on WebSphere for z/OS allows these ways for the client to prove its identity:

  – Basic authentication (also known as SSL Type 1 authentication), in which a client proves its identity to the server by passing a user identity and password known by the target server.

    With SSL basic authentication:

    - An OS/390 or z/OS client can communicate securely with a WebSphere for z/OS server by using a user ID and password.
    - An OS/390 or z/OS client can communicate securely with a server on a WebSphere distributed platform by using a DCE principal and password.
    - A distributed platform client can communicate securely with a WebSphere for z/OS server by using a MVS user ID and password.
    - Because a password is always required on a request, only simple client-to-server connections can be made. That is, the server cannot send a client's user ID to another server for a response to a request. This function is called *identity assertion* or *trusted association*. More about that below.

  – Client certificate support, in which both the server and client supply digital certificates to prove their identities to each other.

    Web applications may have thousands of clients, which makes managing client authentication an administrative burden. Through RACF *certificate name filtering*, SSL support on WebSphere for z/OS allows you to map client certificates, without storing them, to MVS user IDs. Through certificate name filtering, you can authorize sets of users to access servers without the administrative overhead of creating MVS user IDs and managing client certificates for every user.

  – Kerberos security, in which a server proves its identity by passing a digital certificate to the client. A client proves its identity to the server using Kerberos authentication.

  – Identity assertion, or trusted association, in which an intermediate server can send the identities of its clients to a target server in a secure yet efficient manner. This support uses client certificates to establish the intermediate server as the owner of an SSL session. Through RACF, the system can check that the intermediate server can be trusted (special SAF permission is given to address spaces, such as control regions, that run secure system code). Once trust in this intermediate server is established, client identities (MVS user IDs) need not be separately verified by the target server; those client identities are simply asserted without requiring authentication.

- To interoperate in a secure way with other products such as:
  - CICS Transaction Server for z/OS
  - WebSphere on distributed platforms
  - CORBA-compliant Object Request Brokers

SSL support is optional: running WebSphere for z/OS without using SSL affects only the SSL functions that protect communication and authenticate clients and servers.

The following describes how an SSL connection works:

Stage	Description
Negotiation	After the client locates the server, the client and server negotiate the type of security for communications. If SSL is to be used, the client is told to connect to a special SSL port.
Handshake	The client connects to the SSL port and the SSL handshake occurs. If successful, encrypted communication starts. The client authenticates the server by inspecting the server's digital certificate. If client certificates are used during the handshake, the server authenticates the client by inspecting the client's digital certificate.
If basic authentication is used	After the SSL handshake occurs, the client supplies a user identity and password over an SSL-encrypted pipe to establish the client's identity to the server. If the server is on OS/390, the client supplies a user ID and password. If the server is on a workstation, the client supplies a DCE principal and password.
First client request	When the server receives the first client request, the server and RACF establish an OS/390 user identity for the client certificate and runs the request under that client identity. If RACF authenticates the user ID, the server runs work requests under the client identity. If client authentication fails, communication stops.
Ongoing communication	During the SSL handshake, the client and server negotiate a cipher spec to be used to encrypt communications.

Rules:

- Only server control regions and OS/390 clients require access to Cryptographic Services System SSL. Your control regions and OS/390 clients require access to the *hlq*.SGSKLOAD data set. Place SGSKLOAD into LPA. For more information, see *z/OS System Secure Sockets Layer Programming*, SC24-5901.
- Either a Java or C++ client on OS/390 can interoperate with a WebSphere for z/OS or workstation server and use SSL.
- Part of the handshake is to negotiate the cryptographic specs used by SSL for message protection. The security level of the Cryptographic Services installed on your system determines the cipher specs and key sizes available for WebSphere for z/OS. (For more information, see *z/OS System Secure Sockets Layer Programming*, SC24-5901.)
- You must use RACF or equivalent for storing digital certificates and keys. Placing digital certificates and keys into a key database in the HFS is not an option.
- The Daemon server does not use SSL.

Overview of SSL basic authentication security for your application server and clients

To define SSL basic authentication security, you must first request a signed certificate for your server and a certificate authority (CA) certificate from the certificate authority that signed your server certificate. The process of requesting certificates is beyond the scope of this manual. For more information about requesting a certificate, see *z/OS System Secure Sockets Layer Programming*, SC24-5901.

After you have received a signed certificate for your server and a CA certificate from the certificate authority, you must use RACF to authorize the use of digital certificates, store server certificates and server key rings in RACF, and define SSL security properties for your server through the Administration application.

For clients, you must create a key ring and attach to it the CA certificate from the certificate authority that issued the server’s certificate. For an OS/390 client, you must use RACF to create a client key ring and to attach the CA certificate to that key ring.

Figure 1 on page 57 shows the certificate arrangement involved in SSL basic authentication.

- **For the client to authenticate the server**, the server (actually, the control region user ID) must possess a signed certificate created by a certificate authority (CA). The server passes the signed certificate to prove its identity to the client. The client must possess the CA certificate from the same certificate authority that issued the server’s certificate. The client uses the

CA certificate to verify that the server's certificate is authentic. Once verified, the client can be sure that messages are truly coming from that server, not someone else.

- **For the server to authenticate the client**, note that there is no client certificate that the client passes to prove its identity to the server. In the SSL basic authentication scheme, the server authenticates the client by challenging the client for a user ID and password.

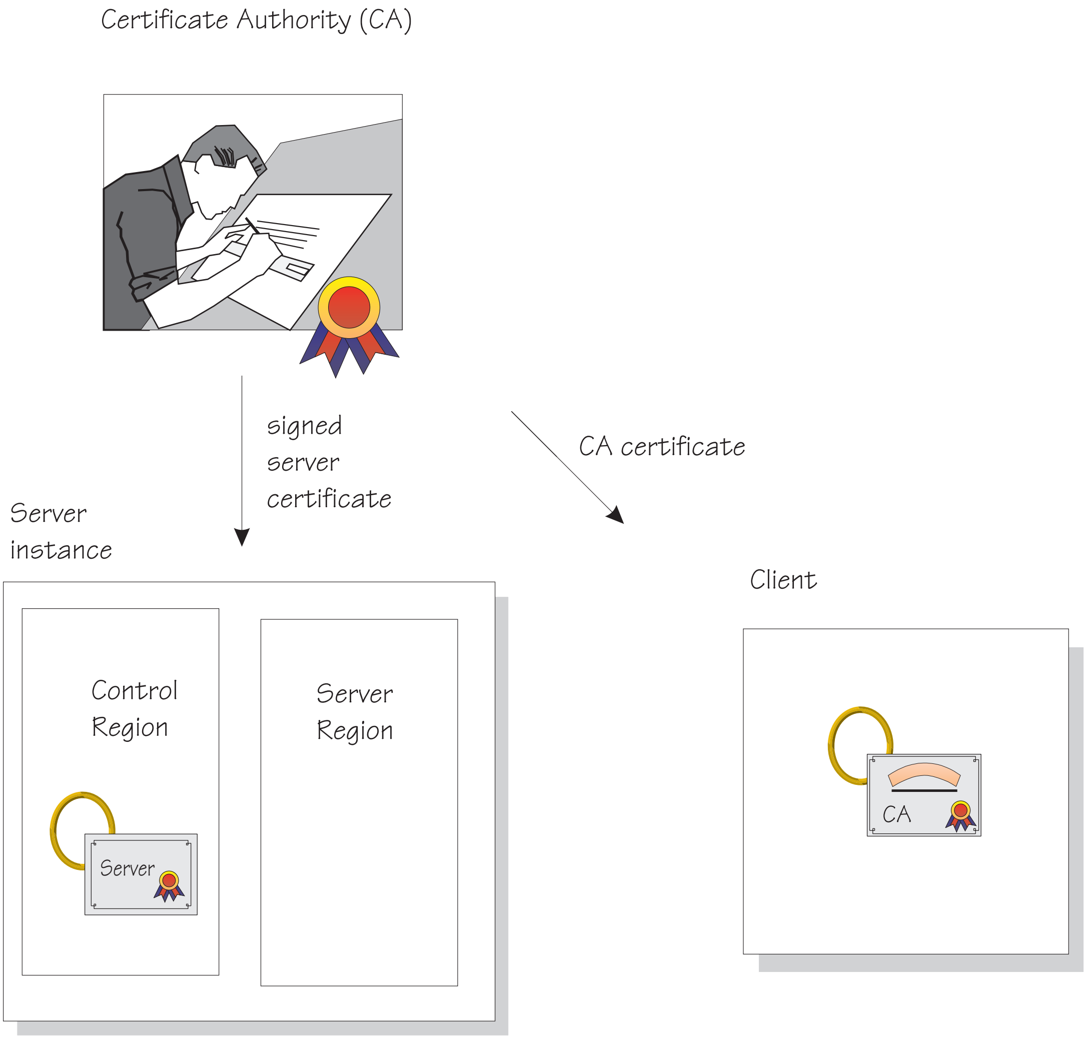


Figure 1. Certificate arrangement for SSL basic authorization

Rules:

- For Java clients on platforms other than OS/390, you must have WebSphere Application Server Enterprise Edition 3.5 to interoperate with a WebSphere for z/OS server and use SSL basic authentication. C++ clients on other platforms cannot use SSL basic authentication when interoperating with WebSphere for z/OS.
- For SSL basic authentication, clients are authenticated in the following ways:
  - An OS/390 client communicating with a remote OS/390 server uses the remote user ID and password (REM_USERID and REM_PASSWORD) environment variables in the client environment file to authenticate the client identity.
  - If an OS/390 client uses SSL with a Component Broker server on other platforms, the client must pass a DCE principal and password defined to the server by using the REM_DCEPRINCIPAL and REM_DCEPASSWORD environment variables.
  - An OS/390 client must also identify its key ring through the SSL_KEYRING environment variable.
  - A client on a WebSphere Application Server distributed platform communicating with an OS/390 server uses a user dialog supplied by the ORB, in which the user supplies a user ID and password.

The following table shows the subtasks and associated procedures for defining SSL basic authentication security:

Subtask	**Associated procedure (See . . .)**
Requesting a server certificate and a certificate authority (CA) certificate	*z/OS System Secure Sockets Layer Programming*, SC24-5901
Setting up SSL basic authentication security for servers	“Steps for using RACF to authorize the server to use digital certificates” on page 61 “Steps for defining server security properties for SSL security” on page 63
Setting up SSL basic authentication security for clients	“Steps for setting up SSL security for clients” on page 64

Overview of SSL client certificate security for your application server and clients

To define SSL client certificate security, you must first request signed certificates for your server and clients and certificate authority (CA) certificates from the certificate authority that signed those certificates. The process of requesting certificates is beyond the scope of this manual. For more information about requesting a certificate, see *z/OS System Secure Sockets Layer Programming*, SC24-5901.

After you have received signed certificates and CA certificates from the certificate authority, you must use RACF to authorize the use of digital certificates, store certificates and key rings in RACF, and define SSL security properties for your server through the Administration application.

Each client identified by a digital certificate must eventually be converted into a MVS user ID by the target WebSphere for z/OS server. If the client and server share the same RACF database, then you do not have to do any additional configuration for this mapping. If the client and server do not share the same RACF database, you can configure the mapping by:

- Adding client certificates to the RACF database of the target server. This may be impractical in most cases.
- Mapping groups of clients into RACF identities using RACF certificate name filtering.
- Using a combination of the two.

Figure 2 on page 60 shows the certificate arrangement involved in SSL client certificate authentication.

- **For the client to authenticate the server**, the server (actually, the control region user ID) must possess a signed certificate created by a certificate authority (CA). The server passes the signed certificate to prove its identity to the client. The client must possess the CA certificate from the same certificate authority that issued the server's certificate. The client uses the CA certificate to verify that the server's certificate is authentic. Once verified, the client can be sure that messages are truly coming from that server, not someone else.
- **For the server to authenticate the client**, the client must possess a signed certificate created by a certificate authority (CA2). (In Figure 2 on page 60 we show two different certificate authorities for clarification; it is possible that the same certificate authority supplies signed certificates to both the server and client.) The server must possess the CA2 certificate from the same certificate authority that issued the client's certificate. The server uses the CA2 certificate to verify that the client's certificate is authentic. Once verified, the server can be sure that messages are truly coming from that client, not someone else.

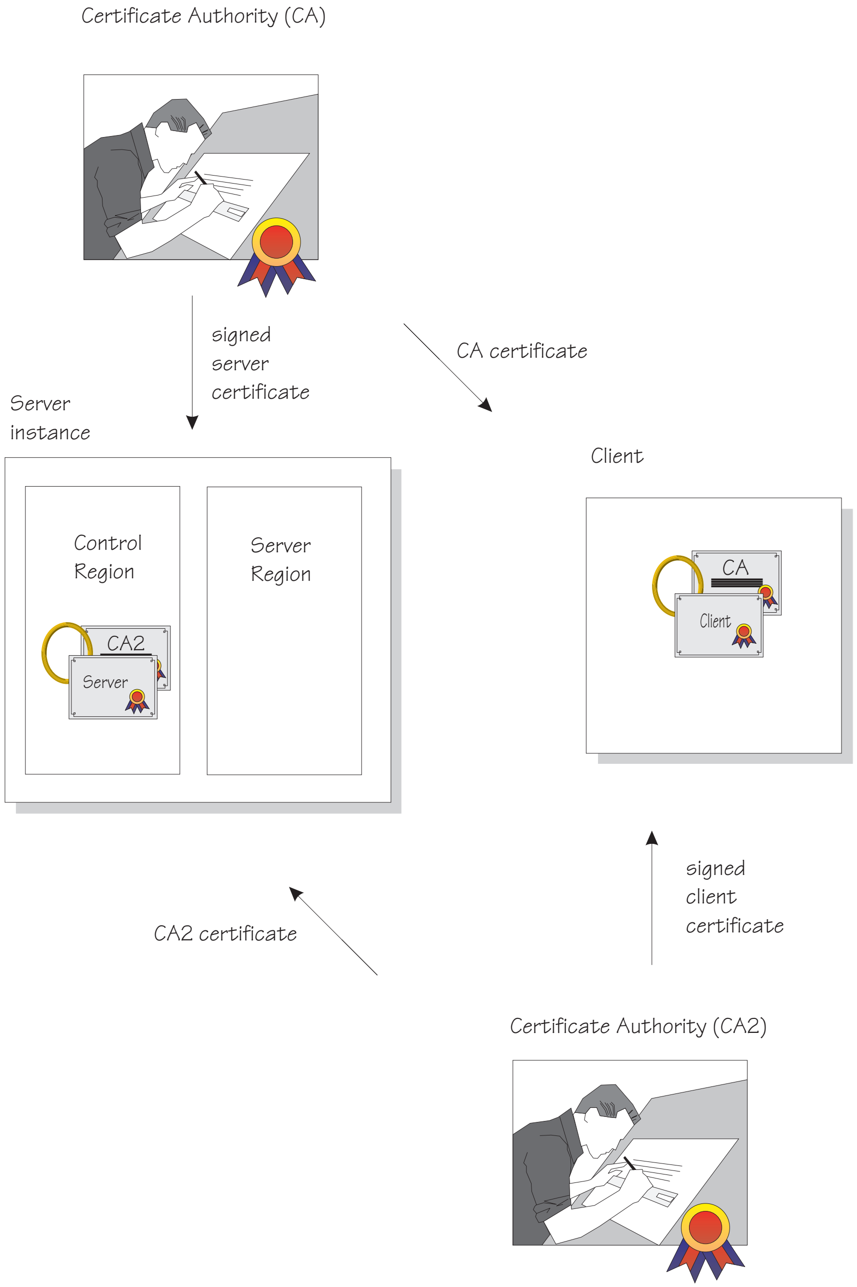


Figure 2. Certificate arrangement for SSL client certificate security

The following table shows the subtasks and associated procedures for defining SSL client certificate security:

Subtask	Associated procedure (See . . .)
Requesting a server certificate and a certificate authority (CA) certificate	*z/OS System Secure Sockets Layer Programming*, SC24-5901
Setting up SSL client certificate security for servers	“Steps for using RACF to authorize the server to use digital certificates” “Steps for defining server security properties for SSL security” on page 63
Setting up SSL client certificate security for clients	“Steps for setting up SSL security for clients” on page 64
Mapping client digital certificates to MVS user IDs on your server’s system	“Steps for mapping client digital certificates to MVS user IDs on your server’s system” on page 65

Defining SSL security for clients and servers

This section includes the procedures you must follow to implement all SSL–based authentication mechanisms.

Steps for using RACF to authorize the server to use digital certificates: SSL uses digital certificates and public/private keys. If your application server uses SSL, you must use RACF to store digital certificates and public/private keys for the user identities under which the server control regions run.

Before you begin: You need to request a certificate authority (CA) certificate and a signed certificate for your server.

If you plan to implement SSL client certificate support, you must also have certificate authority (CA) certificates from each certificate authority that verifies your client certificates. See *z/OS System Secure Sockets Layer Programming*, SC24-5901.

You must have a user ID with the authority to use the RACDCERT command in RACF (for example, SPECIAL authority). For details about RACDCERT, see *z/OS SecureWay Security Server RACF Command Language Reference*, SA22-7687, and *z/OS SecureWay Security Server RACF Security Administrator’s Guide*, SA22-7683.

Perform the following steps authorizing the use of digital certificates:

1. For each server that uses SSL, create a key ring for that server’s control region user ID.

   Example: Your control region is associated with the user ID called CBACRU1. Issue:

   ```
   RACDCERT ADDRING(ACRRING) ID(CBACRU1)
   ```

2. Receive the certificate for your application server from the certificate authority.

   Example: You requested a certificate and the certificate authority returned the signed certificate to you, which you stored in a file called CBACRU1.CA. Issue:

   ```
   RACDCERT ID (CBACRU1) ADD('CBACRU1.CA') WITHLABEL('ACRCERT') PASSWORD('password')
   ```

3. Connect the signed certificate to the control region user ID's key ring and make the certificate the default certificate.

   Example: Connect the certificate labelled ACRCERT to the key ring ACRRING owned by CBACRU1. Issue:

   ```
   RACDCERT ID(CBACRU1) CONNECT (ID(CBACRU1) LABEL('ACRCERT') RING(ACRRING) DEFAULT)
   ```

4. If you plan to have the server authenticate clients (SSL client certificate support):

   - Receive each certificate authority (CA) certificate that verifies your client certificates. Give each CA certificate the CERTAUTH attribute.

     Example: Receive the CA certificate that will verify a client with user ID CLIENT1. That certificate is in a file called USER.CLIENT1.CA. Issue:

     ```
     RACDCERT ADD('USER.CLIENT1.CA') WITHLABEL('CLIENT1 CA') CERTAUTH
     ```

   - Connect each client's certificate authority (CA) certificate to the control region user ID's key ring.

     Example: Connect the CLIENT1 CA certificate to the ring ACRRING owned by CBACRU1.

     ```
     RACDCERT ID(CBACRU1) CONNECT(CERTAUTH LABEL('CLIENT1 CA') RING(ACRRING))
     ```

5. Give read access for IRR.DIGTCERT.LIST and IRR.DIGTCERT.LISTRING in the RACF FACILITY class to the control region user ID.

   Example: Your control region user ID is CBACRU1. Issue:

   ```
   PERMIT IRR.DIGTCERT.LIST     CLASS(FACILITY) ID(CBACRU1) ACC(READ)
   PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(CBACRU1) ACC(READ)
   ```

You are done with the RACF phase when the RACF commands succeed. Continue on to “Steps for defining server security properties for SSL security” on page 63.

Steps for defining server security properties for SSL security: This procedure tells you how to specify that a server use SSL client certificate security through the Administration application.

Before you begin: You need to start the Administration application, log on, and create a new conversation. For more information, see *WebSphere Application Server V4.0 for z/OS and OS/390: System Management User Interface*, SA22-7838.

Perform the following steps to define security characteristics for the server:

1. Expand Servers in the Conversations tree.

2. Create a new server, or click the name of your existing server.

3. In the properties form:
   - If you are implementing SSL basic authentication, click the SSL Type 1 (basic authentication) check box.
   - If you are implementing SSL client certificates, click the SSL Client Certificates check box.
   - If you are implementing Kerberos, click the Kerberos check box.
   - If you are implementing asserted identities, click the Asserted identity check box. Be sure to also click the SSL client certificates check box.

4. Specify the SSL RACF key ring. This is the key ring you defined in step 1 in "Steps for using RACF to authorize the server to use digital certificates" on page 61.

   Note: If you specify the wrong RACF key ring, the server gets an error message at run time.

5. Specify the SSL V2 timeout value, which is the length of time, in seconds, that the system holds session keys. The range is 0-100 seconds. The default is 100 seconds.

6. Specify the SSL V3 timeout value, which is the length of time, in seconds, that the system holds session keys. The range is 0-86400 (1 day). The default is 600 seconds.

7. Order the security preference list. For more information about the security preference list, see *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834.

   ___

8. Complete all other specifications for the server, then validate, commit, complete all tasks, and activate the conversation.

   ___

You know you are done when the system tells you the conversation is activated.

Steps for setting up SSL security for clients: All clients must have access to the server's certificate authority (CA) certificate so they can authenticate the server during the SSL handshake. If you plan to implement SSL client certificate support, clients additionally must have their own certificates as the default certificate on their key rings.

- If your clients are connecting to WebSphere for z/OS from WebSphere on workstations, you must import SSL certificates into the workstation system. For more information and instructions, see *WebSphere Application Server Enterprise Edition Component Broker System Administration Guide*, SC09-4445.
- On OS/390, clients must have certificates attached to their keyrings in RACF.

This procedure explains how to attach certificates to OS/390 clients.

Before you begin: For SSL basic authentication, you must request a CA certificate from the same certificate authority that issued signed certificates for your application servers. If you plan to implement SSL client certificate support, you must additionally request a signed certificate for the client from a certificate authority.

You must have a user ID with the authority to use the RACDCERT command in RACF (for example, SPECIAL authority). For details about RACDCERT, see *z/OS SecureWay Security Server RACF Command Language Reference*, SA22-7687, and *z/OS SecureWay Security Server RACF Security Administrator's Guide*, SA22-7683.

Perform the following steps to authorize use of digital certificates by OS/390 clients:

1. Create a key ring for the OS/390 client.

   Example: Your client user ID is CLIENT1. Issue:

   ```
   RACDCERT ADDRING(C1RING) ID(CLIENT1)
   ```

   ___

2. Receive the server's certificate authority (CA) certificate and give it the CERTAUTH attribute.

   Example: You requested a CA certificate and the certificate authority returned its certificate to you, which you stored in a file called USER.CBSERVER.CA. Issue this command:

   ```
   RACDCERT ADD('USER.CBSERVER.CA') WITHLABEL('VERI CA') CERTAUTH
   ```

3. Connect the server's CA certificate to the client key ring.

   Example: Connect the VERI CA certificate to the C1RING key ring owned by CLIENT1.

   ```
   RACDCERT ID(CLIENT1) CONNECT(CERTAUTH LABEL('VERI CA') RING(C1RING))
   ```

4. In the client's environment file, code the SSL_KEYRING environment variable to correspond to the client's key ring.

   For more information about environment variables, see *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834.

5. If you are implementing SSL client certificate support:
   - Receive the certificate for your client from the certificate authority.

     Example: You requested a certificate and the certificate authority returned a signed certificate which you stored in CLIENT1.SIGNED.CERT. Issue:

     ```
     RACDCERT ID (CLIENT1) ADD('CLIENT1.SIGNED.CERT') WITHLABEL('CLIENT1 CERT') PASSWORD('password')
     ```
   - Connect the client's signed certificate to the client user ID's key ring and make the certificate the default certificate.

     Example: Connect the certificate labelled CLIENT1 to the key ring C1RING owned by CLIENT1. Issue:

     ```
     RACDCERT ID(CLIENT1) CONNECT (ID(CLIENT1) LABEL('CLIENT1 CERT') RING(C1RING) DEFAULT)
     ```

You are done when the RACF commands succeed and you save your environment file.

Steps for mapping client digital certificates to MVS user IDs on your server's system: Each Component Broker client who has presented a digital certificate to authenticate its identity, but does not have an individual certificate registered with RACF on the target server's system or sysplex, must have a mapping to a valid MVS user ID. You can create this mapping by using RACF certificate name filters.

You can create RACF certificate name filters based on either the client's or certificate issuer's distinguished name, as contained in the X.509 digital certificates.

Before you begin: You should know how you want to organize sets of clients that will be presenting digital certificates, and what sort of access those clients need.

You need to have the authority to issue the RACDCERT MAP command.

Perform the following steps to set up certificate name filtering:

1. Define a MVS user ID for each user ID you associate with a certificate name filter. Consider assigning the PROTECTED and RESTRICTED attributes to each one. The PROTECTED attribute protects the user ID from being used to log on directly to the system and from being revoked through incorrect password attempts. The RESTRICTED attribute ensures that the user ID will not be used to access protected resources it is not explicitly authorized to access. **Example:**

   ```
   ALTUSER WEBUSER NOPASSWORD RESTRICTED
   ```

2. Activate certificate name filtering. **Example:**

   ```
   SETROPTS CLASSACT(DIGTNMAP) RACLIST(DIGTNMAP)
   ```

3. Create a certificate name filter. **Example:** The following filter associates the user ID WEBUSER to any user presenting a certificate issued by VeriSign Class 1, who does not have an individual certificate registered with RACF on your system:

   ```
   RACDCERT ID(WEBUSER) MAP WITHLABEL('INTERNET OTHERS') +
           IDNFILTER('OU=VeriSign Class 1 Individual Subscriber.O=VeriSign, Inc.L=Internet')
   ```

   This filter is based on the issuer's name. You can create other filters based on the subject's name, or on combinations of the issuer's and subject's names. For more information about certificate name filtering, see *z/OS SecureWay Security Server RACF Security Administrator's Guide*, SA22-7683.

4. Refresh the DIGTNMAP class. **Example:**

   ```
   SETROPTS RACLIST(DIGTNMAP) REFRESH
   ```

You are done when the SETROPTS command completes.

Setting up Kerberos security for WebSphere for z/OS

On WebSphere for z/OS, Kerberos works with SSL to provide a complete authentication mechanism:

- SSL secures the transportation layer to protect messages. SSL also provides the mechanism whereby the client authenticates the server.
- Kerberos provides the mechanism whereby the server authenticates the client. That is, the client sends the server a Kerberos Generic Security Service Application Program Interface (GSS_API) token, which is used by the server to authenticate the identity of the client.
- Through the GSS_API token, a server is able to pass the client's identity to another server in order to satisfy a client's request. This is called delegation.

The following describes how a Kerberos over SSL connection works:

Stage	Description
Negotiation	After the client locates the server, the client and server negotiate the type of security for communications. If Kerberos is to be used, the client is told to connect to a special SSL port.
Handshake	The client connects to the SSL port and the SSL handshake occurs. If successful, SSL message protection begins. The client authenticates the server by inspecting the server's digital certificate.
Client authentication	After the SSL handshake occurs, the client establishes its Kerberos identity and obtains a Kerberos GSS_API token based on this identity and the server's Kerberos principal. The client sends this token to the server along with a unique SSL connection identifier. The server uses the GSS_API token to authenticate the Kerberos principal that represents the client. Once the client has been authenticated, the system uses RACF to obtain the z/OS user ID that has been mapped to the client's Kerberos principal. This z/OS user identity is used in future authorization checks. By default the client constructs the GSS_API token so that delegation is enabled. This will allow the server to impersonate the client on requests made on its behalf. The z/OS user ID, the Kerberos delegated credentials, and the unique SSL connection identifier are stored for use on future requests made over this SSL Kerberos connection. If the Kerberos client authentication, or the mapping of the authenticated principal fails, communication stops.

Stage	Description
Ongoing communication	Communication between the client and server use SSL services for message protection. Each message includes the unique SSL connection identifier, which allows the server to match a request to its stored z/OS user ID and Kerberos delegated credentials.

This support requires SSL security to be set up. In addition to SSL requirements, Kerberos requires the following to be installed and configured on your OS/390 system:

- OS/390 SecureWay Security Server Network Authentication and Privacy Service for OS/390. For OS/390 V2R8 and V2R9, this support is available through the following Web site:

  `http://www.software.ibm.com`

  For OS/390 V2R10 and z/OS, this support is part of SecureWay Security Server.
- The PTFs for your OS/390 system. Consult the PSP bucket for more information.
- The Kerberos security server must be active on the client and server systems where this support is used.
- All OS/390 user IDs (for clients and servers) that participate in Kerberos authentication must have a Kerberos RACF segment that defines their Kerberos principal.
- The Kerberos server is not required to have a file that contains its Kerberos secret key. Kerberos on OS/390 has eliminated this requirement and can use the Kerberos principal associated with the current system identity to decrypt the service ticket. WebSphere for z/OS servers must use this feature.
- The WebSphere for z/OS server must have READ access to the IRR.RUSERMAP resource in the RACF FACILITY class.
- Kerberos security relies on time coordination among its participants. The Kerberos security administrator should select a time provider and ensure that participants in Kerberos security use that time source to maintain their system time.

The following table shows the subtasks and associated procedures for defining Kerberos security:

Subtask	Associated procedure (See . . .)
Setting up SSL for basic authorization	"Setting up SSL security for WebSphere for z/OS" on page 53

Subtask	Associated procedure (See . . .)
Enabling the Kerberos server	*z/OS SecureWay Security Server Network Authentication Service Administration*, SC24-5926
Associating the server identity with a Kerberos principal.	“Step associating a server identity with a Kerberos principal”
Defining server attributes for Kerberos	“Steps for defining server security attributes for Kerberos”
Setting up a client to use Kerberos	“Steps for setting up a client to use Kerberos” on page 70

Step associating a server identity with a Kerberos principal

Before you begin: You need to have a RACF user ID established for the server’s control region.

Perform the following step to associated the server identity with a Kerberos principal:

⇔ Issue the ALTUSER command to make the association. **Example:**

```
ALTUSER ctl_ID PASSWORD(new_password) NOEXPIRED
        KERB(KERBNAME(kerberos_principal))
```

where

ctl_ID
Is the user ID assigned to the server’s control region through the STARTED class.

new_password
Is the shared OS/390 or z/OS and Kerberos password.

kerberos_principal
Is the Kerberos principal name associated with this OS/390 or z/OS user ID.

You know you are done when the RACF command succeeds.

Steps for defining server security attributes for Kerberos

This procedure tells you how to specify that a server use Kerberos security through the Administration application.

Before you begin: You need to start the Administration application, log on, and create a new conversation. For more information, see *WebSphere Application Server V4.0 for z/OS and OS/390: System Management User Interface*, SA22-7838.

Perform the following steps to define security characteristics for the server:

1. Expand Servers in the Conversations tree.

2. Create a new server, or click the name of your existing server.

3. In the properties form, click the Kerberos allowed checkbox.

4. Specify the SSL RACF key ring. This is the key ring you defined in step 1 in “Steps for using RACF to authorize the server to use digital certificates” on page 61.

   Note: If you specify the wrong RACF key ring, the server gets an error message at run time.

5. Specify the SSL V2 timeout value, which is the length of time, in seconds, that the system holds session keys. The range is 0-100 seconds. The default is 100 seconds.

6. Specify the SSL V3 timeout value, which is the length of time, in seconds, that the system holds session keys. The range is 0-86400 (1 day). The default is 600 seconds.

7. Order the security preference list. For more information about the security preference list, see *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834.

8. Complete all other specifications for the server, then validate, commit, complete all tasks, and activate the conversation.

You know you are done when the system tells you the conversation is activated.

Steps for setting up a client to use Kerberos

Before you begin: You must have SSL basic authentication set up.

You need to install and configure OS/390 SecureWay Security Server Network Authentication and Privacy Service for OS/390 (Kerberos). Enable a SecureWay Security Server (KDC) on each OS/390 or z/OS image where clients will use Kerberos. For more information, see *z/OS SecureWay Security Server Network Authentication Service Administration*, SC24-5926.

Perform the following steps to set up a client to use Kerberos.

1. Use RACF to map each OS/390 or z/OS user that will participate as a Kerberos client to a Kerberos principal on the local realm. **Example:**

   ```
   ALTUSER client_ID PASSWORD(CBIVP) NOEXPIRED KERB(KERBNAME(kerberos_principal))
   ```

   where

   client_ID
   Is the client's user ID.

   kerberos_principal
   Is the Kerberos principal name that will be associated with this OS/390 or z/OS user ID.

   Tip: You can use a utility to help a security adminstrator migrate a OS/390 or z/OS RACF registry to Kerberos. The utility is located at the following Web site:

   ```
   http://sandbox.s390.ibm.com/products/racf/kmigrate.html
   ```

   ---

2. Use RACF to set up cross-realm trust relationships between the realms where the target servers reside and the clients reside. **Example:** A client is in Kerberos realm CLIENTREALM and the server is in SERVERREALM:

   ```
   RDEFINE REALM /.../CLIENTREALM/krbtgt/SERVERREALM KERB(PASSWORD(password1))
   RDEFINE REALM /.../SERVERREALM/krbtgt/CLIENTREALM KERB(PASSWORD(password2))
   ```

   where *password1* and *password2* are passwords. These two commands must be issued to each RACF database.

   ---

3. Use RACF to set up foreign user mapping in server realms. **Examples:**
   a. To map all principals from a foreign-realm to a single user ID, issue:

      ```
      RDEFINE KERBLINK /.../foreign_realm APPLDATA('user_ID')
      ```

   b. To map an individual principal from a foreign-realm to a user ID, issue:

      ```
      RDEFINE KERBLINK /.../foreign_realm/principal APPLDATA('user_ID')
      ```

   where

   foreign_realm
   Is the foreign realm.

   user_ID
   Is the MVS user ID.

   principal
   Is the principal.

You know you are done when the RACF commands succeed.

Adding a new administrator for the Administration application

The default administrator for the Administration application is CBADMIN. If you want to add an administrator, you must perform the following tasks:

Subtask	Associated procedure (See . . .)
Creating an MVS user ID or using a current one **Note:** Give the new administrator user ID the same RACF authorizations as CBADMIN.	*z/OS TSO/E Administration*, SA22-7780, or *z/OS SecureWay Security Server RACF Security Administrator's Guide*, SA22-7683
Updating the access control list for LDAP	"Steps for updating the access control list for LDAP"
Defining the new administrator to the Administration application	*WebSphere Application Server V4.0 for z/OS and OS/390: System Management User Interface*, SA22-7838
Granting the administrator user ID System Management database authority	"Step for granting the new administrator database authorities" on page 74

Steps for updating the access control list for LDAP

If you add an administrator for the Administration application, you must add that administrator to the access control list in LDAP.

Before you begin: You need to set up the LDAP server. We assume you have already set up an exclusive LDAP server for WebSphere for z/OS administrative purposes. For more information about setting up the LDAP server, see *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834.

You also need the bboslapd.conf file currently in use by the LDAP server.

Perform the following steps to change the access control list for LDAP:

1. View the bboslapd.conf file and note the following:
   a. Administrator distinguished name. **Example:**
   ```
   adminDN        "cn=CBAdmin"
   ```
   b. Administrator password. **Example:**
   ```
   adminPW        mypass
   ```
   c. Root naming context (RDN) for the WebSphere for z/OS name space structure. **Example:**

```
   suffix        "o=BOSS,c=US"
```

2. Start the LDAP server:

```
S BBOLDAP
```

3. Extract the current access control list with the `ldapcp` command. **Example:**

```
/u/myself-> ldapcp -p 1389
GLD4005I Environment variable file not found.  Environment variables not set.
GLD6009I No DN entered.  Enter DN now.
ldapcp> cn=CBAdmin
GLD6010I No password entered.  Enter password now.
ldapcp>

GLD6019I Communicating with server on port 1389.
ldapcp> acl q ob "o=boss,c=us"
 object = o=boss,c=us
 aclSource = O=BOSS,C=US
 aclPropagate = TRUE

 acl = access-id:CBADMIN:object:ad:normal:rwsc

 acl = access-id:CBSYMCR1:object:ad:normal:rwsc

 acl = group:CN=ANYBODY:normal:rsc

 acl = access-id:CN=BOSSAdmin,O=BOSS,C=US:object:ad:normal:rwsc

ldapcp>quit
```

4. Create a new file in your home directory (for example, acl_update.txt). Add these lines to the file:

```
dn: o=boss, c=us
changetype:modify
replace:x
```

5. Following the first three lines you added to the file, add aclentry statements for each of the acl lines you extracted in step 3. Add a new aclentry statement for USER1.

   Notes:

   a. It is important to add the dash ('-') at the end.
   b. The output format of the ldapcp command is not the same as the input aclentry lines ("`acl=`" must change to "`aclentry`:", for example).

c. The aclentry for USER1 in the example gives USER1 the same authority as CBADMIN.

Example:

```
aclentry: access-id:cn=BOSSAdmin, o=boss, c=us:normal:rwsc:object:ad
aclentry: access-id:USER1:normal:rwsc:object:ad
aclentry: access-id:CBADMIN:normal:rwsc:object:ad
aclentry: access-id:CBSYMCR1:normal:rwsc:object:ad
aclentry: group:CN=ANYBODY:normal:rsc
-
```

6. Save the update file and issue the following ldapmodify command:

```
u/myself-> ldapmodify -v -p 1389 -D "cn=CBAdmin" -w mypass -f acl_update.txt
```

Result: ldapmodify responds with:

```
modifying entry o=BOSS, c=US
```

7. Repeat step 3 on page 73 to verify that you have added a new user to the access control list.

You know you are done when you see the new user in the access control list.

Step for granting the new administrator database authorities

Your new administrator requires execute authority for CBSYSMGT_PKG and select, update, insert, and delete authority for the tables required for an administrator to deploy a J2EE application in the system management database.

Before you begin: You need to have a user ID with DB2 for z/OS or OS/390 SYSADM authority.

Perform the following step to grant the new administrator database authorities.

⇔ Issue the following commands:

```
GRANT EXECUTE ON PACKAGE CBSYSMGT_PKG.*    TO user_ID

GRANT SELECT,UPDATE,INSERT,DELETE ON TABLE
BBO.BBOMT80_J2EEAPP TO user_ID;

GRANT SELECT,UPDATE,INSERT,DELETE ON TABLE
BBO.BBOMT81_MODULE TO user_ID;

GRANT SELECT,UPDATE,INSERT,DELETE ON TABLE
```

```
BBO.BBOMT82_COMPONENT TO user_ID;

GRANT SELECT,UPDATE,INSERT,DELETE ON TABLE
BBO.BBOMT83_METHOD TO user_ID;

GRANT SELECT,UPDATE,INSERT,DELETE ON TABLE
BBO.BBOMT86_DATASI TO user_ID;

GRANT SELECT,UPDATE,INSERT,DELETE ON TABLE
BBO.BBOMT87_COMP_DS TO user_ID;
```

where *user_ID* is the administrator user ID you defined.

You know you are done when the GRANT commands succeed.

Logging messages and trace data for Java server applications

By using the WebSphere for z/OS support for logging application messages and trace data, you can improve the reliability, availability, and serviceability of any Java application that runs in a WebSphere for z/OS server. Through this support, your Java application’s messages can appear on the MVS master console, in the error log stream, or in the component trace (CTRACE) data set for WebSphere for z/OS. Your application’s trace entries can appear in the same CTRACE data set.

Determining where to issue the messages

You might want to issue messages to the MVS master console to report serious error conditions for mission-critical applications. Through the master console, an operator can receive and, if necessary, take action in response to a message that indicates the status of an application. In addition, by directing messages to the master console, you can trigger automation packages to take action for specific conditions or events related to your application’s processing.

Any messages that your application issues to the console also appear in either the error log stream or the CTRACE data set for WebSphere for z/OS, depending on the message type. Logging the messages in these system resources can help you more easily diagnose errors related to your application’s processing. Similarly, issuing requests to log trace data in the CTRACE data set is another method of recording error conditions or collecting application data for diagnostic purposes.

System performance when logging messages and trace data

You can select the amount and types of trace data to be collected, which provides you with the ability to either run your application with minimal tracing when performance is a priority, or run your application with detailed tracing when you need to recreate a problem and collect additional diagnostic information.

The error log stream, the CTRACE data set for WebSphere for z/OS, and the master console are primarily intended for monitoring or recording diagnostic data for system components and critical applications. Depending on your installation's configuration, directing application messages and data to these resources might have an adverse affect on system performance. For example, if you send application data to the CTRACE data set, trace entries in that data set might wrap more quickly, which means you might lose some critical diagnostic data because the system writes new entries over existing ones when wrapping occurs. Use this logging support judiciously.

Note: You can use the WebSphere for z/OS support for logging messages and trace data only for Java applications, not for Java applets. See *WebSphere Application Server V4.0 for z/OS and OS/390: Assembling J2EE Applications*, SA22-7836, for more information about logging messages and trace data for Java server applications.

Issuing application messages to the MVS master console

With the WebSphere for z/OS reliability, availability, and servicability support for Java (JRAS), you can issue messages from your Java application to the MVS master console. You might want to issue messages to the master console to report serious error conditions for mission-critical applications, or to trigger automation packages.

The messages your application issues also appear in either the error log stream or the component trace (CTRACE) data set that WebSphere for z/OS uses.

Logging the messages is another method of recording error conditions or collecting application data for diagnostic purposes.

Using a message logger

WebSphere for z/OS provides code that creates and manages a message logger, which processes your application's messages. WebSphere for z/OS creates only one message logger for each unique organization, product, or component, so that you can more easily identify the messages recorded in the error log stream or CTRACE data set for a specific application. The message logger runs in the Java Virtual Machine (JVM) for the WebSphere for z/OS server in which your Java application will run.

To use a message logger, all you need to do in your Java application is:

1. Define the message logger.
2. Drive the method to instruct WebSphere for z/OS to create the message logger.
3. Code messages at appropriate points in your application.

See:

- *WebSphere Application Server V4.0 for z/OS and OS/390: Assembling J2EE Applications*, SA22-7836, for more general information.
- *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834, for information on how to set up the error log stream.

Cold starting WebSphere Application Server

See:

- *WebSphere Application Server V4.0 for z/OS and OS/390: System Management User Interface*, SA22-7838, for information on preparing for a cold start.
- *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834, for the cold start procedure.

If this is not an initial cold start, use your current configuration file that you saved for the cold start rather than the initial file sent with WebSphere for z/OS.

Chapter 8. WebSphere for z/OS tuning and performance monitoring

This chapter describes general WebSphere for z/OS tuning guidelines and performance monitoring procedures for EJBs and the servlet EJB integrated runtime. Tuning considerations specific to MOFW objects (the predecessor to EJBs) are included at the end of the chapter.

Note: This chapter does NOT cover the standalone servlet runtime or the standalone Webserver tuning considerations. For more information on that, please go to this Web site: `http://www.ibm.com/s390/ebusiness/perform.html`

One of the goals of the WebSphere for z/OS programming model and runtime is to significantly simplify the work required for application developers to write and deploy applications. Sometimes we say that WebSphere for z/OS relieves the application programmer of many of the plumbing tasks involved in developing applications.

For example, application code in WebSphere for z/OS does not concern itself directly with remote communication–it locates objects which may be local or remote and drives methods. Therefore, you won't see any direct use of socket calls or TCP/IP programming in a WebSphere for z/OS application.

This separation of what you want to do from where you do it is one aspect of removing the application programmers from plumbing tasks. Other considerations are not having to deal with data calls for some types of beans, potentially user authentication, and threading. There are generally no calls from the application code to touch sockets, RACF calls, or management of threading. Removing this from the application programmer doesn't mean this work won't get done. Rather, it means that there may be more work for the DBA, the network administrator, the security administrator, and the performance analyst.

This chapter will focus on the performance tuning aspects of WebSphere for z/OS. This becomes a complex exercise because the nature of the runtime involves many different components of the operating system and middleware.

Before you read a description of WebSphere for z/OS tuning guidelines, it is important to note that, no matter how well the middleware is tuned, it cannot make up for poorly designed and coded applications. Focusing on the application code can help improve performance. Often, poorly written or designed application code changes will make the most dramatic improvements to overall performance.

Tuning WebSphere for z/OS runtime

Diagnostics

The first thing to do is review the WebSphere for z/OS configuration. One simple way to do this is to look in your application control and server regions in SDSF. When each server starts, the runtime prints out the current configuration data in the joblog.

Note: There is an environment variable called `SHOW_SERVER_SETTINGS=YES` which will ensure all configuration values are printed out.

Starting with the basics, you should ensure that you are not collecting more diagnostic data than you need. You should check your WebSphere for z/OS tracing options to ensure that `TRACEALL=0` or 1, and that `TRACEBASIC` and `TRACEDETAIL` are not set.

Note: `TRACEALL=2` can cut performance by 10x (and `TRACEALL=3` even more than that), so, unless you are debugging a problem with the IBM support team, we advise that you never set `TRACEALL` above 1.

`TRACEBASIC` and `TRACEDETAIL` allow WebSphere for z/OS runtime component-specific level of tracing corresponding to `TRACEALL=2` and `3` respectively. Specifying these incrementally will add overhead as you trace more components. Even the simplest level of tracing will slow down your application by 2x.

If you use any level of tracing, including `TRACEALL=1`, ensure that you set `TRACEBUFFLOC` to `BUFFER`. `TRACEALL=1` will write exceptions to the TRACE log as well as to the ERROR log. CTRACE is a much more efficient collection mechanism than SYSPRINT and may improve performance.

To reduce memory requirements, you can set the `TRACEBUFFERNUMBER=4` and `TRACEBUFFERSIZE=128`, which will get 512KB of storage for the trace buffers (the minimum allowed).

Disable JRAS tracing. To do this, look for the following lines in the trace settings file:

```
com.ibm.ejs.*=all=enable
com.ibm.ws390.orb=all=enable
```

and either change ″=enable″ on both lines to ″=disable″ or delete the two lines altogether. For more information, see ″Chapter 4. Tracing Java server applications″ in *WebSphere Application Server V4.0 for z/OS and OS/390: Messages and Diagnosis*.

Disable DEBUG for each server on the SM GUI server specification. Unless you are using On-Line Trace (OLT) or distributed debugger, you should set the DEBUG allowed field to NO. This will prevent the WebSphere for z/OS runtime from calling the online trace/debug interfaces for each method.

Program locations

The next thing to review in the configuration is where your program code is located. IBM recommends that you install as much of the WebSphere for z/OS code itself in LPA as is reasonable, and the remainder in the linklist. This ensures that you have eliminated any unnecessary steplibs which can adversely affect performance. At this time, we have not measured performance of having the runtime located in the HFS, nor can we comment on USS system and user shared libraries with respect to performance. However, it is recommended that for simplicity, any C/C++ application code be located in the HFS. There is a performance cost to doing this, however. You should ensure that only server regions have visibility to your application code. The control region usually runs with no steplibs since all the code required is located in system locations. Verify that the STEPLIB DD in the control region and sever region procs do not point to anything unnecessary.

Review the PATH statement to ensure that only required programs are in the PATH and that the order of the PATH places frequently-referenced programs in the front. If you are using Java, refer to “JVM” on page 83.

Storage

Ensure that you don’t underestimate the amount of virtual storage applied to the WebSphere for z/OS servers. Generally, they use significantly more virtual memory than traditional application servers on z/OS or OS/390. The setting of REGION on the JCL for the proc should be large (at least 256MB to run), and much larger if high throughput is required. You can get an idea of the virtual storage usage through RMF or other performance monitors. It would not be unreasonable for the server region procs to specify REGION=0M, which tells the operating system to give all the available region (close to 2GB).

Note: For more information on REGION=0M and IEFUSI, please see ″Chapter 2. Preparing the base OS/390 or z/OS environment″ (specifically, the section ″Recommendations for using memory″) in *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834.

If you choose not to put most of the runtime in LPA, as described in the program locations section, you may find that your paging subsystem gets a bigger workout as the load increases. At a minimum, WebSphere for z/OS will start seven address spaces so that any code that is not shared will load seven copies rather than one. As the load increases, many more server regions may start and will contribute additional load on the paging subsystem.

LE Heap

The LE Heap is the next level of storage management to be concerned with. For servers, IBM has compiled default values for HEAP and HEAPPOOL into the server main programs. These are good starting points for simple applications. You can use the LE function RPTSTG(ON) on the PARM= in JCL to get a report on storage utilization for your application servers, and then override the values using the PARM= in the procs. Make sure that you remove RPTSTG since it does incur a small performance penalty to collect the storage use information. For your client programs that run on z/OS or OS/390, we recommend that you at least specify HEAPP(ON) on the proc of your client to get the default LE heappools.

Note: Ensure that if you use LE HEAPCHECK, you turn it off once you have ensured that your code doesn't include any uninitialized storage. HEAPCHECK can be very expensive.

Garbage collection and JVM_HEAPSIZE

Specifying a sufficient JVM_HEAPSIZE is important to Java performance. The JVM has thresholds it uses to manage the JVM's storage. When the thresholds are reached, the garbage collector (GC) gets invoked to free up unused storage. GC can cause significant degradation of Java performance.

In order to get it to run less frequently, you can give the JVM more memory. This is done by specifying a larger value for JVM_HEAPSIZE. The default of 256M is a good starting point and may need to be raised for larger applications. By default, the servers run with JVM_HEAPSIZE=256M and JVM_MINHEAPSIZE=256M.

Notes:

1. It is good for the JVM_MINHEAPSIZE to equal the JVM_HEAPSIZE because it allows the allocated storage to be completely filled before GC kicks in. Otherwise, GC would be running constantly, trying to maintain small chunks of storage, and performance would be compromised.
2. Make sure the region is large enough to hold the specified JVM heap.
3. Beware of making the JVM_HEAPSIZE *too* large. While it initially improves performance by delaying garbage collection, it ultimately compromises performance when garbage collection eventually kicks in (because it runs for a longer time).

To determine if you are being affected by garbage collection, you can specify JVM_ENABLE_VERBOSE_GC. This will write a report to the output stream each time the garbage collector runs. This is not a very human-friendly report, but you will get an idea of what is going on with Java GC.

Server region recycling

WebSphere for z/OS has a feature called server region recycling (previously named ″server region garbage collection″). This gives the installation a threshold for the number of transactions to execute in a specific server region before the server region is thrown away. This can be very helpful in improving the performance of applications which have storage leaks.

The default specification for server region recycling is 50 thousand transactions. What that means is that, after 50 thousand transactions, the server region will no longer pick up any new work. A new server region is started to pick up the new work while the old server region finishes up the work it already has. When it finishes, it terminates, and the process continues with the new server region.

When your application code is leaking storage, it can cause each storage obtain to get slower and slower. After the specified number of transactions, you get a new SR and storage obtains are fast again. This produces a somewhat saw-toothed performance curve where performance starts good and degrades until the SR is recycled and performance once again improves. Clearly the long term solution for application leaks is to fix the application, but server region recycling can help when you can’t fix the application immediately.

JVM

When using Java, you should ensure that you:

- Have the most recent version of JVM that is supported by WebSphere for z/OS. As of this writing, the JVM level for WebSphere Application Server V4.0 for z/OS and OS/390 is 1.3.0 PTF 7.
- Have the most recent PTFs, since almost every PTF level has improved performance of the JVM.
- Have sufficient JVM_HEAPSIZE (mentioned previously).
- Run with the JIT (Just In Time) compiler active. This is done by omitting the ″JAVA_COMPILER=″ option from the environment file.″
- Have not specified the debug version of the JVM libjava_g in your libpath. The debug version will not perform as well as the non-debug version.
- Have CLASSPATH point to only the classes you need (the classes that are referenced most frequently should be located near the front of the path, if possible).
- Verify the CLASSPATH as part of the Java configuration.

Note: For more information about JVM performance on z/OS and OS/390, see `http://www.s390.ibm.com/java/perform.html`.

Performance information and accounting

WebSphere for z/OS relies on its use of WLM services to collect some of the accounting and performance data. This information gets presented back to the installation through RMF and RMF-written SMF records. In addition, WebSphere for z/OS has its own SMF records which collect additional domain-specific information for WebSphere for z/OS. First, unless you need the SMF records or RMF data, turn them off. Controlling these SMF records is done in the `SMFPRMxx` parmlib statement. If you do need SMF information, you should review the SMF parmlib to ensure you are collecting only the data that you need (both record types and detail). You can control the detail of the WebSphere for z/OS SMF records. Finally, verify your SMF dataset is allocated optimally by verifying the CI size is defined. It should be about 26K for best throughput.

Note: Please see "Chapter 9. Systems Management Facility (SMF) recording and monitoring" on page 93 for more information on SMF.

Setting up your workload manager goals and filtering criteria is probably beyond the scope of this section. However, you should recognize that you can classify work into service classes based on userid and server name. You should ensure that you classify the control regions as reasonably high-performing system tasks.

Topology

Single server or multiple servers?

WebSphere for z/OS gives you the ability to install your application either in a single server or spread it across multiple servers. There are many reasons for partitioning your application. However, for performance, placing your application all in the same server will always provide better performance than partitioning it. If you do choose to partition your application across servers, you will get better performance if there are at least replica servers on each system in the sysplex. The Application Server runtime will try to keep calls local to the system if it can, which will, for example, use local interprocess calls rather than sockets.

One tran or multiple trans?

You also have a choice of running server regions with an isolation policy of one tran per server region or multiple trans per server region. From a performance perspective, we have not been able to define a specific benefit of one over the other.

Local client or remote client?

The difference between a local client, where the client and the optimized communication are done on the same system, and remote client, where the client cost is not on the platform but replaced by the additional communication overhead of sockets, is almost equivalent. Latency is better for a local client than for a remote client, meaning you will get better response time with a local client.

One copy of a server or many replicas?

You can define more than one copy of a server on a system. These copies are called replicas. We have found slight improvements in performance when running with a couple of replicas as opposed to just one. While there is some benefit, IBM does not recommend, at this time, the creation of replicated control regions for the sole purpose of improving performance. We do, however, recommend them for eliminating a single point of failure and for handling rolling upgrades without introducing an outage.

Container configuration

In WebSphere for z/OS, there are several types of EJBs and several transaction policies supported. Selection of each type has performance implications. While we won’t be able to give an exhaustive treatise on this yet, we will give some rules of thumb.

EJBs

There are two basic bean types in WebSphere for z/OS: session and entity.

Session beans: Within a session bean in WebSphere for z/OS, there are stateless and stateful session beans.

Stateless session bean
The lowest overhead type of bean. They are cheap to create, do very little automatically and, if not cleaned up by the application, will go away when the server terminates.

Stateful session bean
Slightly more overhead than the stateless session bean.

Entity beans: In WebSphere Application Server V4.0 for z/OS and OS/390, entity beans come in two flavors: bean managed persistence (BMP) and container managed persistence (CMP).

Since managing persistence is the responsibility of the bean in BMP, it really depends on the way the load and store is implemented whether a BMP is faster than a CMP. CMP beans manage persistence. A well-implemented BMP bean will probably be faster than a typical CMP bean. However, over time, CMP will get more sophisticated and will be much easier for the average application programmer to maintain.

Transaction policies

There are seven transaction policies in WebSphere for z/OS:

- TRANSACTION_REQUIRES
- TRANSACTION_REQUIRES_NEW
- TRANSACTION_SUPPORTS
- TRANSACTION_NOT_SUPPORTED
- TRANSACTION_BEAN_MANAGED
- TRANSACTION_NEVER
- TRANSACTION_MANDATORY

Within this specification, we also have local transaction and global transactions. Generally, local transactions are the fastest.

MOFW considerations

For MOFW objects, there are several additional transaction policies: HYBRID_GLOBAL and SUPPORTS_HYBRID_GLOBAL. These policies are similar to local transactions in the EJB world. The overhead of a transaction is reduced since they do not assume a full two-phase OTS mediated transaction. This policy is not standard and should be used carefully since it does affect the behavior of the application and possibly makes it nonportable.

Transient objects in WebSphere for z/OS are about twice as fast as persistent objects when running in a HYBRID_GLOBAL transaction. This is primarily because there is no interaction with any other resource manager, which eliminates the need to coordinate a transaction. Also, there is no need to do any logging to disk, so the latency of the transaction is improved.

If possible, read-only persistent data can be configured in a container which has a pinned policy. This means that the data will be read once from the database per server and kept in memory rather than being retrieved for each transaction.

To improve MOFW query performance, the environment variable SOMOOSQL=1 is set by default in the server configuration. When this is set, we can easily push queries down to DB2. This has a significant improvement in query performance. IBM recommends that you set this variable to on. When set to on, you lose some NLS support in Query.

You should also ensure that your query calls are being pushed down to DB2. This is most easily determined by looking at DB2PM reports where you look at the detail of the calls to DB2. You can review the resulting query statement that was issued to DB2 and see how many fetches are done from DB2. See “Tuning tips for DB2” on page 89 for more information.

Finally, the definition of each home (or container), has an option for whether you need method level access checking. If you don’t need it, you can improve performance by shutting it off.

Security

As a general rule, two things happen when you increase security: the cost per transaction increases and throughput decreases.

By default, WebSphere for z/OS runs with security on. The runtime will always incur a small price to collect and carry the security credential information for users and the server. Since all the security authorization checks are done with SAF (RACF or equivalent), you can choose to enable and disable SAF classes to control security. A disabled class will cost a negligible amount of overhead.

When a class is active, the number of profiles in a class will affect the overall performance of the check. Placing these profiles in a (RACLISTed) memory table will improve the performance of the access checks. Audit controls on access checks also effect performance. Usually, you audit failures and not successes. Audit events are logged to DASD and will increase the overhead of the access check.

If you are using EJBROLEs, specifying more roles on a method will lead to more access checks that need to be executed and a slower overall method dispatch. If you are not using EJBROLEs, do not activate the class.

Authentication

You have several options when dealing with authentication:

- **Local authentication:** Local authentication the fastest type because it is highly optimized.
- **UserID and password authentication:** Authentication that utilizes a userid and password has a high first-call cost and a lower cost with each subsequent call.
- **Kerberos security authentication:** We have not adequately characterized the cost of kerberos security yet.
- **SSL security authentication:** SSL security is notorious in the industry for its performance overhead. Luckily, there is a lot of assists available from hardware to make this reasonable on z/OS. We can’t comment on specifics yet, but a starting point is to follow the configuration options for the Webserver’s SSL.

Note: For more information on security, see *WebSphere Application Server V4.0 for z/OS and OS/390: Installation and Customization*, GA22-7834.

Servlet / EJB integrated runtime

While we can't comment with specifics on the integrated runtime performance, there are some general statements. If you are running just a servlet, the integrated runtime may not initially show an improvement in performance. However, when a servlet is calling on an EJB, it will benefit greatly from the integrated runtime. Essentially, the integrated runtime will convert the remote method calls to local in-process EJB invocations (which should be much faster). The servlet relies on the EJBs to speed up the performance.

Collecting performance diagnosis information

When you suspect that you have a performance problem in the runtime of WebSphere for z/OS, there are a couple of standard things that we ask for you to do. The first is to run your application and take a 15-minute sample of your application's performance. The sample data should be gathered with RMF monitor I. The following are the parameters that we ask that you set for the RMF collection:

```
CPU
CHAN
CYCLE(1000)
DEVICE(NOCHRDR)
DEVICE(COMM)
DEVICE(DASD)
DEVICE(NOGRAPH)
DEVICE(NOTAPE)
DEVICE(NOUNITR)
ENQ(SUMMARY)
INTERVAL(15M)
IOQ(DASD)
IOQ(COMM)
NOVSTOR
OPTIONS
PAGING
PAGESP
RECORD
REPORT(REALTIME)
NOSTOP
SYSOUT(H)
WKLD(PERIOD,SYSTEM)
TRACE(CCVUTILP)
```

Note: We need to also know which WLM service classes represent the Application Server workload so that we know where to start.

If you suspect that you are having throughput problems in a particular address space, for example by looking at some other real-time performance data, we may need to see a dump of one or more address spaces. This is done using the following parameters:

```
JOBNAME=(<jobname list>)
SDATA=(LSQA,PSA,SQA,SUM,SWA,TRT,WLM,CSA,RGN)
```

Tuning tips for z/OS or OS/390

- The first place to review is your CTRACE configuration. Ensure that all components are either set to `MIN` or `OFF`. This will eliminate any unnecessary overhead of collecting trace information that is not being used. Often during debug, CTRACE is turned on for a component and not shut off when the problem is debugged.
- Ensure that LE and C++ runtimes are loaded into LPA for best performance.
- Throughput above 200 transactions a second will benefit from moving the RRS logs in logger to a CF logstream. Generally, transactions that complete quickly will not require any DASD I/O. If you don't need the archive log, we recommend that you eliminate it since it can introduce extra DASD I/Os. The archive log contains the results of completed transactions. Normally, the archive log is not needed.
- Ensure that you are not collecting more SMF data than you need, and that the SMF dataset CI sizes are large to ensure the most efficient writing of SMF data to the dataset.

Tuning tips for DB2

Performance tuning for DB2 is usually critical to the overall performance of a WebSphere for z/OS application. DB2 is often the preferred datastore for a session or EJB. There are many books that cover DB2 tuning–we can't possibly provide as thorough a treatment of DB2 here as we would like. Listed here are just some basic performance guidelines to follow. (Guidelines) template)

- First, ensure that your DB2 logs are large enough and are allocated on the fastest volumes you have. Make sure DASD fastwrite is enabled if you have it. In our runs, the difference was about 2x (based on an I/O difference of 35msec). We also found that the RMF reports do not seem to show the real I/O difference. They showed .1msec before and after the change, but a GTF I/O trace definitely showed the problem was an SSCH and two I/O interrupts—one quick and one later. The RRS CTRACE showed that the time for exits was rather high during commit. It will help to make sure that the database tables are defined in multiples of cylinders.
- Next ensure that you have tuned your bufferpools so that the most often-read data is in memory as much as possible. The setting-up of buffer pool size is a balancing act between defining enough memory to hold everything and not defining more than 2G.
- We recommend that you ensure indexes defined on all your object primary keys. Failure to do so will result in costly tablespace scans.
- Ensure that, once your tables are sufficiently populated, you do a re-org to compact the tables. Executing Runstats will ensure that the DB2 catalog

statistics about table and column sizes and accesses are most current so that the best access patterns are chosen by the optimizer.

- You many want to consider pre-formatting tables that are going to be heavily used. This avoids formatting at runtime.
- You will have to define more connections called threads in DB2. The Application Server uses a lot of threads. Sometimes this is the source of throughput bottlenecks since the server will wait at the create thread until one is available.
- As a bean developer, you have the choice of JDBC or SQLJ. JDBC makes use of dynamic SQL whereas SQLJ generally is static and uses pre-prepared plans. SQLJ requires an extra step to create and bind the plan whereas JDBC does not. SQLJ, as a general rule, is faster than JDBC.
- If you do use JDBC, we recommend that you enable dynamic statement caching in DB2. To do this, modify your ZPARMS to say `CACHEDYN(YES) MAXKEEPD(16K)`. Depending on the application, this can make a very significant improvement in DB2 performance. Specifically, it can help JDBC, LDAP, and MOFW query.
- When coding an iterator, you have a choice of named or positioned. For performance, we recommend positioned iterators.
- With JDBC and SQLJ, you are better off writing specific calls that retrieve just what you want rather than generic calls that retrieve the entire row. There is a very high per-field cost.
- Allocate tables in cylinders (multiple of 720)
- Large CI sizes for logs

Tuning tips for RACF

- As is always the case, don't turn things on unless you need them. In general, the cost of security has been highly optimized. However, if you don't need EJBROLEs then don't enable the class in RACF.
- You should ensure that you place into memory, by the `RACLIST` command, those items that will improve performance. Specifically, ensure that you `RACLIST` (if used):
  - `ACEE`
  - `GTS`
  - `UID/GID`
  - `CBIND`
  - `EJBROLE`
- Use of things like SSL come at a price. If you are a heavy SSL user, ensure that you have appropriate hardware, such as PCI crypto cards, to speed up the handshake process.

Tuning tips for the system logger

- Use CF logs, if possible.

- If it’s not possible to use CF logs, use fastwrite DASD and make sure the logs are allocated with large CI sizes.
- Before OS/390 R10 (which had DB2 6.1), you could write as many as four log records to a logger. The archive log was not required, so you could get up to three records per tran. DB2 7.1 allowed you get up to two log writes per transaction with a PTF, and OS/390 R10 with DB2 7.1 PTF now allows you to get up to one log record per transaction.
- In any case, you should monitor the logger to ensure that there is a sufficient size in the CF and that offloading is not impacting the overall throughput. The transaction logs are one of the only shared I/O intensive resources in the mainline and can affect throughput dramatically if they are mis-tuned.
- DASD logger is limited to 450 I/O’s a second. This is a RAMAC III statement on a G4. Since WebSphere for z/OS writes 3 log records to the logger per transaction (in z/OS or OS/390 R7), we are limited to about 150 trans/sec.
- CF logger was around 2700 I/O’s per second on G4, RAMAC 3. Which means 900 trans/sec/CEC. Supposedly, as we get to G6, this will be even higher.

Tuning tips for TCP/IP

TCP/IP can be the source of some significant remote method delays.

1. First, ensure that you have defined enough sockets to your system and that the default socket time-out of 180 seconds is not too high.
2. Next check the specification of the port in TCPIP profile dataset to ensure that NODELAYACKS is specified as follows:

   ```
   PORT 8082 TCP NODELAYACKS
   ```

   In your runs, changing this could improve throughput by as much as 50% (this is particularly useful when dealing with trivial workloads).
3. You should ensure that your DNS configuration is optimized so that lookups for frequently-used servers and clients are being cached. Sometimes this is related to the name server’s Time To Live (TTL) value. On the one hand, setting the TTL high will ensure good cache hits. However setting it high also means that, if the daemon goes down, it will take a while for everyone in the network to be aware of it.

Chapter 9. Systems Management Facility (SMF) recording and monitoring

This chapter, along with *WebSphere Application Server V4.0 for z/OS and OS/390: System Management User Interface*, SA22-7838, and *z/OS MVS System Management Facilities (SMF)*, SA22-7630, describes how to enable and use the System Management Facilities to collect and record system and job-related information on the WebSphere for z/OS system. This information can be used to bill users, report system reliability, analyze your configuration, schedule work, identify system resource usage, and perform other performance-related tasks that your organization may require.

You can enable SMF recording for:

- **Capacity planning:**

  To determine:
  - How many transactions have run?
  - What is the average and maximum completion time for methods running on each server?
  - How many clients are attached to each server instance? Of these clients, how many are active?
- **Application profiling:**
  - To show an application broken down into its component parts.
  - To provide timing information on the application's component parts.
- **Error reporting:**
  - To detect and record soft failures (those that are generated through an exception or those that are performance-related).
  - To use this error information to trigger an event that will cause an action to occur once a threshold has been reached.

WebSphere for z/OS produces the appropriate SMF records that will allow your installation to perform these functions.

SMF record types

Two types of SMF records can be produced: *activity records* and *interval records*.

Activity records

Gathered as each activity within a server is completed. An activity is a logical unit of business function. It can be a server or user-initiated transaction.

Interval records

Consist of data gathered at installation-specified intervals and provide capacity planning and reliability information.

Four records can be produced: the *server activity record*, *container activity record*, *server interval record*, and the *container interval record*. Each record is described below. For more information about how to activate these records, see *WebSphere Application Server V4.0 for z/OS and OS/390: System Management User Interface*, SA22-7838.

Server activity record

The server activity SMF record is used to record activity that is running inside a WebSphere for z/OS Application Server. This record can be used to perform basic charge-back accounting and to profile your applications to determine, in detail, what is happening inside the WebSphere transaction server.

A single record is created for each activity that is run inside a server or server instance. If the activity runs in multiple servers, then a record is written for each server.

You can activate this record through the server definition of the Systems Management User Interface by checking the checkbox: ″Write Server Activity SMF Records.″

Container activity record

The purpose of the container activity SMF record is to record activity that is running inside a container located inside the WebSphere transaction server. This record can be used to perform basic charge-back accounting, application profiling, problem determination, and capacity planning.

A single record is created for each activity that is run inside a container located in a WebSphere transaction server. If the activity runs in multiple servers, then multiple records are written for the activity.

You can activate this record through the server definition of the Systems Management User Interface by checking the checkbox: ″Write Container Activity SMF Records.″

Server interval record

The purpose of the server interval SMF record is to record activity that is running inside a WebSphere for z/OS application server. This record is produced at regular intervals and is an aggregate of the work that ran inside the server instance during the interval.

A single record is created for each server instance that has interval recording active during the interval. If a server has multiple server instances, then a record for each server instance is written and the records must be merged after processing to get a complete view of the work that ran inside the server.

You can activate this record through the server definition of the Systems Management User Interface by checking the checkbox: ″Write Server Interval SMF Records.″

Container interval record

The purpose of the container interval SMF record is to record activity that is running inside a container located inside the WebSphere transaction server. This record is produced at regular intervals and is an aggregate of the activities running inside a container during the interval. This record can be used to perform application profiling, problem determination, and capacity planning.

A single record is created for each active container located in a WebSphere transaction server within the interval being recorded. If there is more than one server instance associated with a server, there will be a record for the container from each server instance. To get a common view of the work running in the container during the interval, you must merge the records after processing.

You can activate this record through the server definition of the Systems Management User Interface by checking the checkbox: ″Write Container Interval SMF Records.″

Setting up SMF recording

The following section describes what you must do to enable SMF recording, format the output data set, and disable SMF recording for WebSphere Application Server.

Steps for enabling SMF recording

Perform the following steps to enable SMF recording:

1. Enable SMF recording for WebSphere Application Server through the Server Definition of the System Management User Interface Administrator application (see *WebSphere Application Server V4.0 for z/OS and OS/390: System Management User Interface*, SA22-7838). The available choices are:

   Server activity record
   Check the checkbox: ″Write Server Activity SMF Records.″

   Container activity record
   Check the checkbox: ″Write Container Activity SMF Records.″

   Server interval record
   Check the checkbox: ″Write Server Interval SMF Records.″

Container interval record
Check the checkbox: "Write Container Interval SMF Records."

---

2. Edit the SMFPRMxx parmlib member.
   a. Insert an 'ACTIVE' statement to indicate SMF recording. See *z/OS MVS Initialization and Tuning Guide*, SA22-7591.
   b. Insert a SYS statement to indicate the types of SMF records you want the system to create. For example, use `SYS(TYPE(120:120))` to select WebSphere Application Server type 120 records only. Keep the number of selected record types small to minimize the performance impact.

   The server and container interval records will use either:
   - The value specified in the server/container definition as specified in the SM User Interface
   - The interval specified in the SMF parmlib member (from the SMF product settings) if you specify a length of 0.

   You can specify the interval in which you want the Server and Container interval records created in the SMFPRMxx parmlib member (if no interval was specified by the SM EUI for the server or container definition). The default SMF recording interval is 30 minutes. See *WebSphere Application Server V4.0 for z/OS and OS/390: System Management User Interface*, SA22-7838, for details.

---

3. To start writing records to DASD, issue the following command:

```
t smf=xx
```

   where xx is the suffix of the SMF parmlib member (SMFPRMxx). See *z/OS MVS System Management Facilities (SMF)*, SA22-7630, for more information.

   When you activate writing to DASD, the data is recorded in a data set (specified in SMFPRMxx).

Steps for formatting the output data set

Perform the following steps to format the SMF recording output data set into a readable format for printing to the screen or other output device.

Note: For detailed information, see *z/OS MVS System Management Facilities (SMF)*, SA22-7630, the SMF Dump program. Below is a short summary of this information.

1. Enter "`i smf`" from the MVS console to switch the SMF data sets.

2. Run the SMF Dump program (IFASMFDP) to create a sequential data set from the raw dump. A sample JCL is shown in *z/OS MVS System Management Facilities (SMF)*, SA22-7630.

3. View the data set using a program that can display record type 120.

SMF Record Interpreter for record type 120

The WebSphere for z/OS SMF Record Interpreter is a tool which enables the interpretation of complete output datasets from the IBM z/OS utility program IFASMFDP. It writes a header line for all record types and a detailed dump for record type 120.

Note: See the section "Steps for formatting the output data set" on page 96 about how to use IFASMFDP to extract specified subsets of SMF data from the system SMF datasets into a sequential dataset.

The WebSphere for z/OS SMF Record Interpreter dumps all the WebSphere for z/OS relevant data into a printable output file. It is a Java utility, so it needs to be interpreted and executed by a Java Virtual Machine (JVM) under the z/OS or OS/390 UNIX environment.

The printable output of the WebSphere for z/OS SMF Record Interpreter will resemble the following:

```
SMF file analysis starts ...

--------------------------------------------------------------------------------
Record #2,  Type: 120,  Size: 372,  Date: Mon Apr 23 09:13:39 EDT 2001
  SystemID: SY1,  SubsystemID: null,  Flag: 94
  Subtype: 1 (SERVER ACTIVITY)
#Triplets: 3
Triplet:    offset: 64     length: 32     count: 1
Triplet:    offset: 96     length: 192     count: 1
Triplet:    offset: 288     length: 84     count: 1

Triplet #1
ProductSection
  Version: 2, Codeset: IBM-1047
  Endian: 1,  TimeStampFormat: 1 (S390STCK64)
  IndexOfThisRecord: 1, Total#OfRecords: 1, Total#OfTriplets: 3

Triplet #2
ServerActivitySection
  HostName:           PLEX1
  ServerName:         BBOASR1
  ServerInstanceName: BBOASR1A
  ServerType:         MOFW Server
  #OfServerRegions:   1
  ASID1: 49, ASID2: 0, ASID3: 0, ASID4: 0, ASID5: 0
  UserCredentials:    IBMUSER
  ActivityType:       1 (method request)
  ActivityID:         * b5bad234 1f950520 000000dc 00000006 * &sup1;&#141;K..n.....
                            &#245;.... *
                      * 09263048 xxxxxxxx xxxxxxxx xxxxxxxx * ...&#171;............ *
  WlmEnclaveToken:    * 00000020 000001e0 xxxxxxxx xxxxxxxx * .......\........ *
  ActivityStartTime:  * b5bad234 1f950520 xxxxxxxx xxxxxxxx * &sup1;&#141;K..
                            n.......... *
  ActivityStopTime:   * b5bad28b 219e3847 xxxxxxxx xxxxxxxx * &sup1;&#141;K&#164;.
                            &#241;.&#226;........ *
  #InputMethods: 1, #GlobalTransactions: 1, #LocalTransactions: 0

Triplet #3
CommSessionSection
  CommSessionHandle:  * 25a48320 00000001 xxxxxxxx xxxxxxxx * .uc............. *
  CommSessionAddress: jobname=BBOAX8   asid=0035
  CommSessionOptimization: 1 (local optimization)
  DataReceived: 402, DataTransferred: 1688

(...)

SMF file analysis complete.
```

Data from the sequential file is produced record by record. Each record contains a number of triplets which are first described in the record's header section (the first part of a record). The description is then followed by the triplet contents which are presented by the tool in the sequence of their appearance within the record.

Note: See "Appendix A. SMF record type 120 (WebSphere for z/OS)" on page 101 for more information.

Each triplet contains data for a section. Several types of sections are defined, such as:

- ProductSection
- ActivitySection
- CommSessionSection
- etc.

The WebSphere for z/OS SMF Record Interpreter interprets each section in its specific way and prints the interpreteted data into the output file.

Note: Some sections may contain subsections that are also organized by the means of triplets.

SMF ViewTool installation and invocation

The Java SMF Record Interpreter is provided in the form of a jar file named `bbomsmfv.jar`. To use it from the z/OS or OS/390 UNIX environment:

1. Verify that the JAVA_HOME environment variable refers to the current java installation, eg. JAVA_HOME=../usr/bin/java/J1.3.

   Note: This should be at least Java 1.3 since this release is the first to implicitly contain the neccessary record support needed by the interpreter.
2. Copy the file ″bbomsmfv.jar″ to your tools directory.

   Note: Be sure that any edits made to the file in the future are made to both copies of the file, or just execute from the installation directory in the first place.
3. To interpret SMF data from a cataloged z/OS or OS/390 sequential file named ″USER.SMFDATA″ (which was previously created using the IFASMFDP utility as described above), execute:

   ```
   java -cp bbomsmfv.jar com.ibm.ws390.sm.smfview.Interpreter "USER.SMFDATA"
   ```

   Note: It is implicit in the java command parameterization that your current working directory is the tools directory. If this is not the case, you will receive a NoClassDefFoundError on com.ibm.ws390.sm.smfview.Interpreter--Java doesn't generate a diagnostic when it doesn't find bbomsmfv.jar in the current directory.

Steps for disabling SMF recording

Perform the following steps to disable SMF recording:

1. Disable SMF recording for WebSphere Application Server through the server definition of the System Management User Interface Administrator application (see *WebSphere Application Server V4.0 for z/OS and OS/390: System Management User Interface*, SA22-7838).

2. If you want to disable SMF recording for the whole MVS system, edit the SMFPRMxx parmlib member. Set SMFPRMxx to ′NOACTIVE′ to disable the writing of SMF records to DASD.

SMF record type 120 (78) — WebSphere for z/OS

See "Appendix A. SMF record type 120 (WebSphere for z/OS)" on page 101 for details on WebSphere for z/OS SMF record type 120. Also see *z/OS MVS System Management Facilities (SMF)*, SA22-7630.

Appendix A. SMF record type 120 (WebSphere for z/OS)

This appendix describes the layout of SMF records as created by WebSphere for z/OS.

Information resulting from the SMF data gathering process is typically presented with the help of an SMF data viewing tool. This record format description is intended to enable your tool providers to design an SMF data viewing tool. Your system administrators will use an SMF data viewing tool with a description presented by your tool provider, since it requires them to make proper selections that limit the amount of presentation data. For example, they might want to view a specific time frame and only specific containers, classes, and methods. They may also occasionally need to refer to the record descriptions.

For additional information about using SMF records, see *z/OS MVS System Management Facilities (SMF)*, SA22-7630.

Record Type 120 (78) - WebSphere for z/OS Performance Statistics

The following section defines the SMF Record Type 120 (78) - WebSphere for z/OS Performance Statistics. WebSphere for z/OS writes record type 120 to collect WebSphere for z/OS performance statistics. For more information about SMF record types, see *z/OS MVS System Management Facilities (SMF)*, SA22-7630.

Record Type 120 (78) - WebSphere for z/OS Performance Statistics

All subtypes of the record type 120 have the following format:

- Standard header section
- Individual header extension for subtype x
- Product section
- Subtype-specific sections listed below.

Record type 120 has the following subtypes:

- **Subtype 1: Server Activity Record (Version 2)**
  - **Server activity section** (one section per record):

    Contains information about each activity that occurred within one server
  - **Communication session section** (zero, one, or multiple sections per record):

Contains information about each communication session

- **Subtype 2: Container Activity Record (Version 2)**
  - **Container activity section** (one section per record):

    Contains information about each activity that occurred within one container
  - **Class section** (multiple sections per record):

    Contains information about all classes involved in this activity
  - **Method section** (multiple sections per class section):

    Contains information about all methods of this class involved in this activity
- **Subtype 3: Server Interval Record (Version 2)**
  - **Server interval section** (one section per record):

    Contains aggregated information about all activities that occurred within the specified server interval
- **Subtype 4: Container Interval Record (Version 2)**
  - **Container interval section** (one section per record):

    Contains aggregated information about all activities that occurred within one container in the specified interval
  - **Class section** (multiple sections per record):

    Contains information about all classes involved in this activity in the specified interval
  - **Method section** (multiple sections per class section):

    Contains information about all methods of this class involved in this activity in the specified interval

Record environment

The following conditions exist for the generation of this record:

- **Record environment**

  Macro SMFWTM (record exit: IEFU83)

  Mode Task

  Storage Residency
  31–bit

Record mapping

This section includes the header/self-defining and product sections.

Header/Self-defining Section

This section contains the common SMF record header fields and the triplet fields (offset/length/number), if applicable, that locate the other sections on the record. For a description of triplets, see “Triplets and splitting SMF records” on page 115 and *z/OS MVS System Management Facilities (SMF)*, SA22-7630.

Offset	Offset	Name	Length	Format	Description
0	0	SM120LEN	2	binary	Record length. This field and the next field (total of four bytes) form the RDW (record descriptor word). See ″Standard SMF Record Header″ in *WebSphere Application Server V4.0 for z/OS and OS/390: System Management User Interface*, SA22-7838 ,for a detailed description.
2	2	SM120SEG	2	binary	Segment descriptor (see record length field)
4	4	SM120FLG	1	binary	**Bit Meaning When Set** 0: New SMF record format 1: Subtypes used 2: Reserved 3-6: Version indicators* 7: Reserved *See ″Standard SMF Record Header″ in *WebSphere Application Server V4.0 for z/OS and OS/390: System Management User Interface*, SA22-7838, for a detailed description.
5	5	SM120RTY	1	binary	Record type 120(X’78’)
6	6	SM120TME	4	binary	Time since midnight, in hundredths of a second, that the record was moved into the SMF buffer

10	A	SM120DTE	4	packed	Date when the record was moved into the SMF buffer, in the form 0*cyydddF*. See ″Standard SMF Record Header″ in *WebSphere Application Server V4.0 for z/OS and OS/390: System Management User Interface*, SA22-7838 , for a detailed description.
14	E	SM120SID	4	EBCDIC	System identification (from the SMFPRMxx SID parameter)
18	12	SM120SSI	4	EBCDIC	Subsystem identification from SUBSYS parameter
22	16	SM120STY	2	binary	Record subtype: 1: Server activity 2: Container activity 3: Server interval 4: Container interval.
24	18	SM120TRN	4	binary	Number of triplets in this record. A triplet is a set of three SMF fields (offset/length/number values) that defines a section of the record. The offset is the offset from the RDW. Subtypes: 1: Value is equal to the number of sessions +2 2 and 4: Value is equal to the number of classes +2.
28	1C	SM120PRS	4	binary	Offset to product section from RDW.
32	20	SM120PRL	4	binary	Length of product section.
36	24	SM120PRN	4	binary	Number of product sections.

Individual header extension for subtype 1					
40	28	SM120SAS	4	binary	Offset to server activity section from RDW

44	2C	SM120SAL	4	binary	Length of server activity section
48	30	SM120SAN	4	binary	Number of server activity sections
52	34	SM120CSS	4	binary	Offset to communication session section from RDW
56	38	SM120CSL	4	binary	Length of communication session section
60	3C	SM120CSN	4	binary	Number of communication session sections

Individual header extension for subtype 2					
40	28	SM120CAS	4	binary	Offset to container activity section from RDW
44	2C	SM120CAL	4	binary	Length of container activity section
48	30	SM120CAN	4	binary	Number of container activity sections

The following triplet appears 0–n times; once for each class section.					
52	34	SM120CLS	4	binary	Offset to class section from RDW
56	38	SM120CLL	4	binary	Length of class section
60	3C	SM120CLA	4	binary	Number of class sections

Individual header extension for subtype 3					
40	28	SM120SIS	4	binary	Offset to server interval section from RDW
44	2C	SM120SIL	4	binary	Length of server interval section
48	30	SM120SIN	4	binary	Number of server interval sections

Individual header extension for subtype 4					
40	28	SM120CIS	4	binary	Offset to container interval section from RDW
44	2C	SM120CIL	4	binary	Length of container interval section
48	30	SM120CIN	4	binary	Number of container interval sections

The following triplet appears 0–n times; once for each class section.					
52	34	SM120CLS	4	binary	Offset to class section from RDW
56	38	SM120CLL	4	binary	Length of class section
60	3C	SM120CLN	4	binary	Number of class sections

Product Section

Offset	Offset	Name	Length	Format	Description
0	0	SM120MFV	4	binary	CB SMF version
4	4	SM120COD	8	EBCDIC	Character codeset in which strings in the SMF record are encoded
12	C	SM120END	4	binary	Encode of numbers in the SMF record
16	10	SM120TSF	4	binary	Encoding of timestamps: 1: S390STCK64: The time values are encoded in 64-bit S/390 Store Clock format.

Reassembly information.					
20	14	SM120IXR	4	binary	Index of this record
24	18	SM120NRC	4	binary	Total number of records
28	1C	SM120NTR	4	binary	Total number of triplets

Subtype 1: Server Activity Record(Version 2)

1. **Server Activity Section** (one section per record):

   Contains information about each activity that occurred within one server
2. **Communications Session Section** (zero, one, or multiple sections per record):

   Contains information about each communication session

Server Activity Section

Offset	Offset	Name	Length	Format	Description
0	0	SM120HNM	64	EBCDIC	WebSphere transaction server host name
64	40	SM120SNM	8	EBCDIC	WebSphere transaction server name

72	48	SM120SIN	8	EBCDIC	WebSphere transaction server instance name
80	50	SM120SNM	4	binary	Total number of server regions that were involved to process this activity. If applicable, up to the first five server region address space IDs are listed within the next five fields.
84	54	SM120SR1	4	binary	The specific WebSphere transaction server instance server region where the request ran
88	58	SM120SR2	4	binary	The specific WebSphere transaction server instance server region where the request ran
92	5C	SM120SR3	4	binary	The specific WebSphere transaction server instance server region where the request ran
96	60	SM120SR4	4	binary	The specific WebSphere Transaction Server Instance Server Region where the request ran
100	64	SM120SR5	4	binary	The specific WebSphere transaction server instance server region where the request ran
104	68	SM120CRE	8	EBCDIC	The user credentials under which the activity began.
112	70	SM120ATY	4	binary	Type of activity that this record references: 1: Method request: This record refers to a method request that is not part of a global transaction. 2: Transaction: This record refers to a transaction.
116	74	SM120AID	20	EBCDIC	Identity of the activity
136	88	SM120WLM	8	HEX	WLM enclave token
144	90	SM120AST	16	S390STCK	Activity start time
160	A0	SM120AET	16	S390STCK	Activity stop time

176	B0	SM120NIM	4	binary	Number of input methods
180	B4	SM120NGT	4	binary	Number of global transactions that were started in the server region
184	B8	SM120NLT	4	binary	Number of local transactions that were started in the server region
188	BC	SM120STY	4	binary	The WebSphere server type. 0: MOFW server. 1: J2EE server. (Not supported yet)

Communications Session Section

Offset	Offset	Name	Length	Format	Description
0	0	SM120CSH	8	HEX	Communications session handle
8	8	SM120CSA	64	EBCDIC	Communications session address
72	48	SM120CSO	4	binary	Communications session optimization 1: Local communications session: The session is a local OS/390 optimized communications session. 2: Remote communications session: The session is a remote communications session. 3: Remote encrypted (SSL) 4: Remote within sysplex.
76	4C	SM120SDR	4	binary	Data received; the number of bytes received by the server
80	50	SM120SDT	4	binary	Data transferred; the number of bytes transferred from the server back to the client.

Subtype 2: Container Activity Record(Version 2)

1. **Container Activity Section** (one section per record):

   Contains information about each activity that occurred within one container

2. **Class Section** (multiple sections per record):

   Contains information about all classes involved in this activity

3. **Method Section** (multiple sections per class section):

   Contains information about all methods of classes involved in this activity

Container Activity Section

Offset	Offset	Name	Length	Format	Description
0	0	SM120HNM	64	EBCDIC	WebSphere transaction server host name
64	40	SM120SNM	8	EBCDIC	WebSphere transaction server name
72	48	SM120SIN	8	EBCDIC	WebSphere transaction server instance name
80	50	SM120ASR	4	binary	The specific WebSphere transaction server instance server region where the request ran
84	54	SM120CNM	256	EBCDIC	WebSphere container name
340	154	SM120CTP	4	binary	Container transaction policy Values: • 1: Transaction required • 2: Same-Server Hybrid Global • 3: Hybrid Global • 4: Supports Same-Server Hybrid Global.

344	158	SM120CSP	4	binary	Container security policy: • 000000001: DCE • 000000010: Userid password • 000000100: Userid passticket • 000001000: SSL type 1 • 000010000: non-authenticated clients • 000100000: SSL client certificates • 001000000: Kerberos • 010000000: SendAssertedId • 100000000: AcceptAssertedId
348	15C	SM120WLM	8	HEX	WLM enclave token
356	165	SM120ATY	4	binary	Type of activity that this record references: 1: Method request: This record refers to a method request that is not part of a global transaction. 2: Transaction: This record refers to a transaction.
360	168	SM120AID	20	HEX	Identity of the activity

Class Section

Offset	Offset	Name	Length	Format	Description
0	0	SM120CLN	256	EBCDIC	Name of a class activa by the container
256	100	SM120NIC	4	binary	Number of instances c class that were createc
260	104	SM120NIA	4	binary	Number of instances c class that were activat
264	108	SM120NIR	4	binary	Number of instances c class that were remov (deleted)
268	10C	SM120NIP	4	binary	Number of instances c class that were passiva

272	110	SM120RMR	4	binary	Reserved
276	114	SM120RMW	4	binary	Reserved
280	118	SM120MN	4	binary	Number of method in this class section
The following triplet appears 0–n times; once for each method section.					
284	11C	SM120MS	4	binary	Offset to method se from the beginning class section
288	120	SM120ML	4	binary	Length of method s
292	124	SM120MN	4	binary	Number of method sections

Method Section

Note: When a client invokes a method on an object instance in a WebSphere for z/OS server region and that method, in turn, invokes other methods within the same object instance, only the first method invoked by the client will be recorded to SMF. The subsequent methods it invokes within the same object instance will not be recorded.

Offset	Offset	Name	Length	Format	Description
0	0	SM120MNM	256	EBCDIC	Name of the method **Note:** The only way SMF recording distinguishes methods is by method name. SMF recording does not evaluate the complete method signature.
256	100	SM120NMI	4	binary	Number of times the method was invoked during the activity
260	104	SM120NEX	4	binary	Number of non-framework exceptions that were detected by the container

264	108	SM120ART	4	binary	Average response time: The response time is measured in micro seconds. If the value exceeds 2**31 micro seconds, this field becomes negative and the accuracy is changed to seconds. A positive value is the time in micro seconds. A negative value is the time in seconds.
268	10C	SM120MRT	4	binary	Maximum response time: The response time is measured in micro seconds. If the value exceeds 2**31 micro seconds, this field becomes negative and the accuracy is changed to seconds. A positive value is the time in micro seconds. A negative value is the time in seconds.

Subtype 3: Server Interval Record(Version 2)

1. **Server Interval Section** (one section per record):

   Contains information about each activity that occurred within the specified server interval

Server Interval Section

Offset	Offset	Name	Length	Format	Description
0	0	SM120HNM	64	EBCDIC	WebSphere transaction server host name
64	40	SM120SNM	8	EBCDIC	WebSphere transaction server name
72	48	SM120SIN	8	EBCDIC	WebSphere transaction server instance name
80	50	SM120SST	16	S390STCK	Time that the sample began in the server
96	60	SM120SET	16	S390STCK	Time that the sample ended

112	70	SM120NGT	4	binary	Number of global transactions that have run through the server instance during the interval that have been initiated by the server instance during the interval
116	74	SM120NLT	4	binary	Number of local transactions that have been initiated by the server instance during the interval
120	78	SM120NCS	4	binary	Number of communications sessions that exist at the end of the interval
124	7C	SM120NCA	4	binary	The number of communications sessions that have been active during the interval
128	80	SM120NLS	4	binary	Number of local communication sessions that exist at the end of the interval
132	84	SM120NLA	4	binary	Number of active local communication sessions that have been attached and active within the server instance during the interval
136	88	SM120NRS	4	binary	Number of remote communication sessions that exist at the end of the interval
140	8C	SM120NRA	4	binary	Number of active remote communication sessions that have been attached and active within the server instance during the interval
144	90	SM120BTS	4	binary	Number of bytes that have been transferred to the server from all attached clients
148	94	SM120BFS	4	binary	Number of bytes that have been sent from the server to all attached clients

152	98	SM120BTL	4	binary	Number of bytes that have been transferred to the server from all locally attached clients
156	9C	SM120BFL	4	binary	Number of bytes that have been transferred from the server to all locally attached clients
160	A0	SM120BTR	4	binary	Number of bytes that have been transferred to the server from all remotely attached clients
164	A4	SM120BFR	4	binary	Number of bytes that have been transferred from the server to all remotely attached clients
168	A8	SM120STY	4	binary	The WebSphere server type. 0: MOFW server. 1: J2EE server. (Not supported yet)

Subtype 4: Container Interval Record(Version 2)

1. **Container Interval Section** (one section per record):

   Contains information about each activity that occurred within one container in the specified interval
2. **Class Section** (multiple sections per record):

   Contains information about all classes involved in this activity in the specified interval
3. **Method Section** (multiple sections per class section):

   Contains information about all methods of all classes involved in this activity in the specified interval.

Container Interval Section

Offsets	Offsets	Name	Length	Format	Description
0	0	SM120HNM	64	EBCDIC	WebSphere transaction server host name
64	40	SM120SNM	8	EBCDIC	WebSphere transaction server name
72	48	SM120SIN	8	EBCDIC	WebSphere transaction server instance name
80	50	SM120CNM	256	EBCDIC	WebSphere container name

336	150	SM120CTP	4	binary	Cntainer transaction policy Values: • 1: Transaction required • 2: Same-Server Hybrid G • 3: Hybrid Global • 4: Supports Same-Server Hybrid Global.
340	154	SM120CSP	4	binary	The container security poli • 000000001: DCE • 000000010: Userid passw • 000000100: Userid passti • 000001000: SSL type 1 • 000010000: non-authentic clients • 000100000: SSL client certificates • 001000000: Kerberos • 010000000: SendAsserted • 100000000: AcceptAssert
344	158	SM120SST	16	S390STCK	The time that the sample b in the server.
360	168	SM120SET	16	S390STCK	The time that the sample e

Class Section: (See Subtype 2: Class Section)

Method Section: (See Subtype 2: Method Section)

Triplets and splitting SMF records

Triplets

You can use triplets to build self-describing SMF records that contain various types of data sections and a varying number of each of these sections. All data sections are described by triplets that consist of:

1. An offset that specifies the start position of the data
2. A length that describes the length of the section
3. A count that describes how many instances of the section are included in this record.

The two triplets that describe the product section and the general record information section (for example, the section describing the container itself in a container activity record) are located at fixed positions within the record. This allows one to start evaluating the record right after having evaluated the record header.

Splitting SMF Records

Since most of the WebSphere Application Server SMF records are used to describe variable-length data structures (for example, there might be hundreds of classes by container and hundreds of methods by class), the SMF records may be larger than the maximum record size supported by SMF (32KB). In this case, the logical records need to be split into several physical records.

Each of those physical records needs to be self-describing and self-contained. *Self-describing* indicates what we described in the paragraph on triplets before; hich is a purely mechanical structure to help read a record. *Self-contained* indicates that,even if we have only a subset of the physical records at hand that together,describe the original logical record, we need to be able to evaluate these records, combine the information stored in them, and set an ’incomplete’ flag. This is required since,as we break up a logical record into physical records and write them to SMF one after the other, SMF might decide that only the first few physical records fit into the primary SMF dump dataset, whereas the remaining physical records are written into an alternate SMF dump dataset. At the time when a formatted SMF dump dataset is evaluated, we may not assume that all physical records that make up one logical record are present. For example, self-containedness of a physical container activity record means that it contains the description of the container,but not necessarily all of its classes.

We use a similar splitting mechanism like the one that is currently used in the RMF product. Note that,in the case of container records (subtype 2 and 4), we cannot assume that records will be split at a class boundary, but we must consider the case when the methods that belong to one class also need to be split over multiple physical records, as shown in the diagrams below.

Note: The section length numbers used throughout the following diagrams are only for demonstrative purposes. In particular, the arrows indicating 32K boundaries or the total length of the records are placed at random. You can fit many more classes and methods into a physical record than suggested by the diagrams.

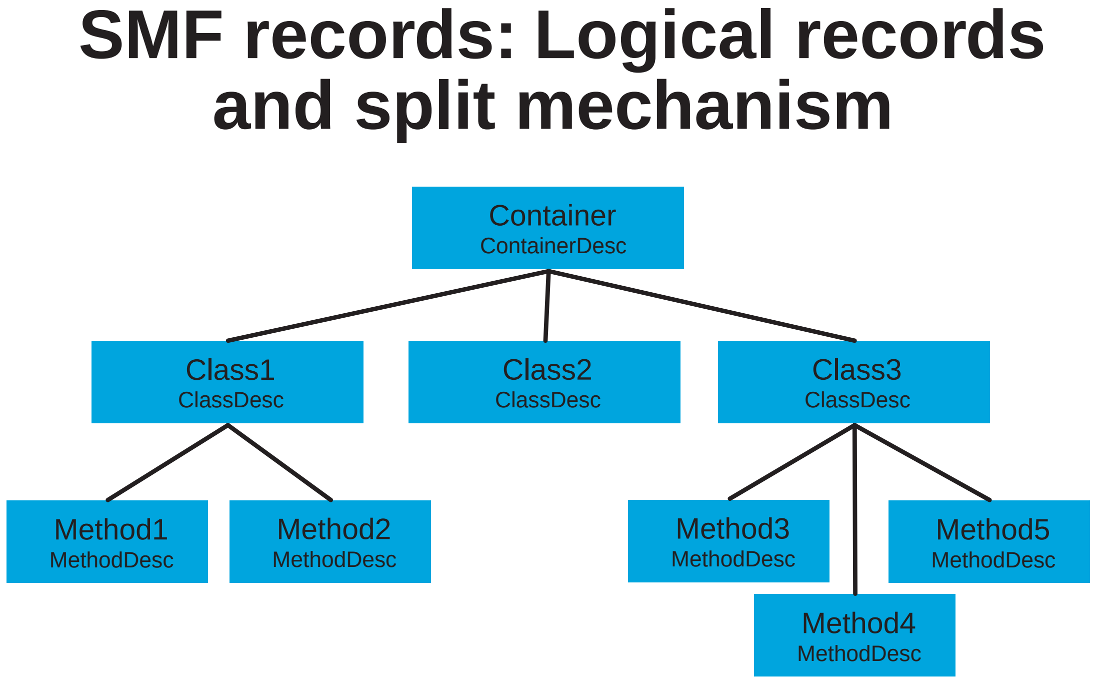
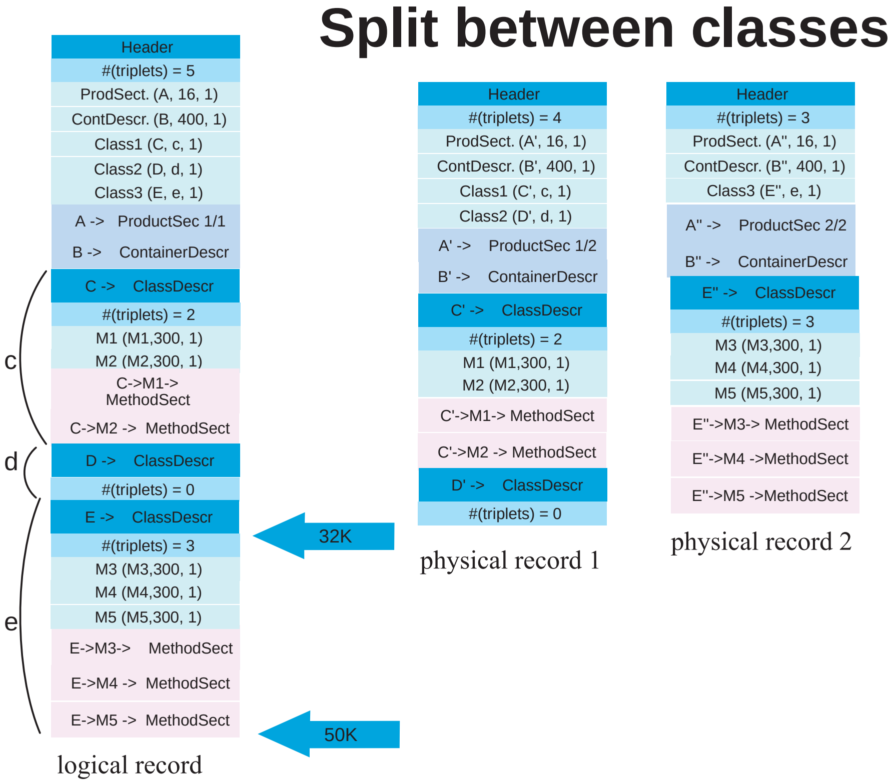


Figure 3. SMF records: logical records and split mechanism



Split between classes

Header
#(triplets) = 5
ProdSect. (A, 16, 1)
ContDescr. (B, 400, 1)
Class1 (C, c, 1)
Class2 (D, d, 1)
Class3 (E, e, 1)
A -> ProductSec 1/1
B -> ContainerDescr
C -> ClassDescr
#(triplets) = 2
M1 (M1,300, 1)
M2 (M2,300, 1)
C->M1-> MethodSect
C->M2 -> MethodSect
D -> ClassDescr
#(triplets) = 0
E -> ClassDescr
#(triplets) = 3
M3 (M3,300, 1)
M4 (M4,300, 1)
M5 (M5,300, 1)
E->M3-> MethodSect
E->M4 -> MethodSect
E->M5 -> MethodSect

logical record

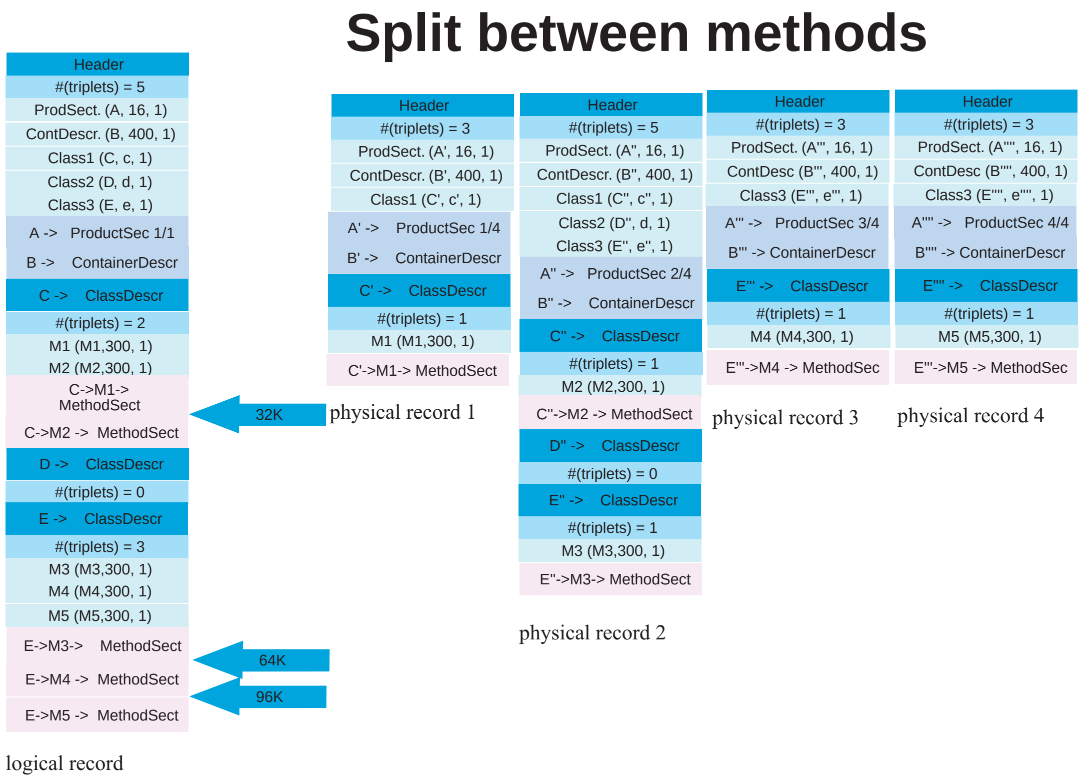
Header
#(triplets) = 4
ProdSect. (A', 16, 1)
ContDescr. (B', 400, 1)
Class1 (C', c, 1)
Class2 (D', d, 1)
A' -> ProductSec 1/2
B' -> ContainerDescr
C' -> ClassDescr
#(triplets) = 2
M1 (M1,300, 1)
M2 (M2,300, 1)
C'->M1-> MethodSect
C'->M2 -> MethodSect
D' -> ClassDescr
#(triplets) = 0

physical record 1

Header
#(triplets) = 3
ProdSect. (A", 16, 1)
ContDescr. (B", 400, 1)
Class3 (E", e, 1)
A" -> ProductSec 2/2
B" -> ContainerDescr
E" -> ClassDescr
#(triplets) = 3
M3 (M3,300, 1)
M4 (M4,300, 1)
M5 (M5,300, 1)
E"->M3-> MethodSect
E"->M4 ->MethodSect
E"->M5 ->MethodSect

physical record 2

Figure 4. SMF record: split between classes



Split between methods

Header
#(triplets) = 5
ProdSect. (A, 16, 1)
ContDescr. (B, 400, 1)
Class1 (C, c, 1)
Class2 (D, d, 1)
Class3 (E, e, 1)
A -> ProductSec 1/1
B -> ContainerDescr
C -> ClassDescr
#(triplets) = 2
M1 (M1,300, 1)
M2 (M2,300, 1)
C->M1-> MethodSect
C->M2 -> MethodSect
D -> ClassDescr
#(triplets) = 0
E -> ClassDescr
#(triplets) = 3
M3 (M3,300, 1)
M4 (M4,300, 1)
M5 (M5,300, 1)
E->M3-> MethodSect
E->M4 -> MethodSect
E->M5 -> MethodSect

logical record

Header
#(triplets) = 3
ProdSect. (A', 16, 1)
ContDescr. (B', 400, 1)
Class1 (C', c', 1)
A' -> ProductSec 1/4
B' -> ContainerDescr
C' -> ClassDescr
#(triplets) = 1
M1 (M1,300, 1)
C'->M1-> MethodSect

physical record 1

Header
#(triplets) = 5
ProdSect. (A'', 16, 1)
ContDescr. (B'', 400, 1)
Class1 (C'', c'', 1)
Class2 (D'', d, 1)
Class3 (E'', e'', 1)
A'' -> ProductSec 2/4
B'' -> ContainerDescr
C'' -> ClassDescr
#(triplets) = 1
M2 (M2,300, 1)
C''->M2 -> MethodSect
D'' -> ClassDescr
#(triplets) = 0
E'' -> ClassDescr
#(triplets) = 1
M3 (M3,300, 1)
E''->M3-> MethodSect

physical record 2

Header
#(triplets) = 3
ProdSect. (A''', 16, 1)
ContDesc (B''', 400, 1)
Class3 (E''', e''', 1)
A''' -> ProductSec 3/4
B''' -> ContainerDescr
E''' -> ClassDescr
#(triplets) = 1
M4 (M4,300, 1)
E'''->M4 -> MethodSec

physical record 3

Header
#(triplets) = 3
ProdSect. (A'''', 16, 1)
ContDesc (B'''', 400, 1)
Class3 (E'''', e'''', 1)
A'''' -> ProductSec 4/4
B'''' -> ContainerDescr
E'''' -> ClassDescr
#(triplets) = 1
M5 (M5,300, 1)
E'''->M5 -> MethodSec

physical record 4

Figure 5. SMF record: split between methods

Appendix B. Naming conventions for application servers

This chapter contains guidelines for establishing a solid naming convention for your application servers.

Understanding the need for application server naming conventions

There are a number of reasons why you need to establish a naming convention for application servers:

1. **Because WebSphere Application Server servers are like IMS or CICS regions.**
   - They contain tailored procedures for the control and server regions.
   - They contain tailored environmental variables for each instance of a server.
   - They contain environmental variables for each instance of a server.
   - Servers may be self-contained or dependent on other servers.
2. **For security.**
   - Regions have user IDs associated with them.
   - Users are allowed access to servers and objects within.
3. **For Workload Manager (WLM).**
   - Classification of regions and work within the regions
   - Application environments.

A WebSphere for z/OS application server consists of a number of address spaces which require the installation to manage configuration files, security profiles, workload classification constructs, and so forth. To create, manage, and recognize application servers, a template is needed for stamping out servers and server instances. The template needs to apply to the following:

- **Server names:**
  - Control region PROC names
  - Server region PROC names
  - Application Environment names
  - Instance names
- **Security:**
  - User/group/uid/gid
  - Control regions
  - Server regions

  - Instance names
- **Procedures:**
  - Environmental files
  - Library names
- **Other:**
  - DB2 collection and package names
  - Log stream names

Here is a system for creating servers based on a 4–character application naming scheme, which we refer to as XXXX. Since multiple instances of a server may exist on one or more systems in the WebSphere for z/OS environment, there is also a requirement to distinguish between servers. You can use a system that looks like the following:

Everything is determined by 4 characters: XXXX (and Y).

```
CBserver name                    = CBXXXX
- APPLENV name                     = CBXXXX
CBserver instance name           = CBXXXXAY
- ORBsrvname default value       = CBXXXXAY
Userid for control region        = CBXXXXC
- PROC for control region          = CBXXXXC
Group id for control region      = CBXXXXG
Userid for server  region        = CBXXXXS
- PROC for server  region          = CBXXXXS
Group id for server  region      = CBXXXX
Default remote userid            = CBXXXXI
Default local  userid            = CBXXXXD
Group id for default ids         = CBXXXXP
```

Here are the user IDs. Change as desired.

```
CBXXXXC   0     - do not change.
CBXXXXS 1100
CBXXXXD 1101
CBXXXXI 1102
```

Here are the groups/GIDS. Change as desired.

```
CBXXXXG 1000
CBXXXXR 1001
CBXXXXP 1002
```

The naming convention is also applied to:

Server specific log streams = CBXXXX.ERROR.LOG
LRMs, = CBXXXX_LRM_DB2
LRMIs, = CBXXXXAY_LRMI_DB2
DB2 collections = CBXXXX_PK
HFS File system names = /WSCapps/CBXXXX/bin and
/WSapps/CBXXXX/lib
OS File names = hlq.CBXXXX.LOADLIB
hlq.CBXXXX.HFS
hlq.CBXXXXAY.PARMS
and so forth.

The part of the naming scheme which breaks down is the management of the UID/GID associated with RACF identities. There appears to be no easy mechanism to automate the assignment or association of these entities with userids.

For example, below you will see one way of defining the procs for the control and server regions associated with application server APP1. Notice that each server instance has its own unique data set containing environmental settings. You could easily change this scheme so that there is one PDS for the entire sysplex specifying different members. The important limitation to remember is that there is minimal capability to pass symbolic parameter overrides to the server regions.

Also notice that data set names indicate whether the data set is unique to the server or common across the sysplex. In our naming scheme, the second level qualifier indicates whether the data set is to be used:

- sysplex -wide,
- only for servers running on a specific system,
- server-wide,
- only for a given server instance.

Control Region Proc:

```
//BBOASR1 PROC SRVNAME='BBOASR1A',
//       PARMS='',
//       CBCONFIG='/WebSphere390/CB390'
//* See instructions at the bottom of this file
//  SET BBOLIB='BBO'
//  SET LELIB='CEE'
//  SET DB2='DB2'
//  SET RELPATH='controlinfo/envfile'
//BBOASR1 EXEC PGM=BBOCTL,REGION=0M,
// PARM='/ -ORBsrvname &amp;SRVNAME &amp;PARMS'
//*STEPLIB  DD DSN=&BBOLIB..SBBOLD2,DISP=SHR
//*         DD DSN=&BBOLIB..SBBOLOAD,DISP=SHR
//*         DD DSN=&LELIB..SCEERUN,DISP=SHR
//*         DD DSN=&DB2..SDSNLOAD,DISP=SHR
```

```
//BBOENV    DD PATH='&CBCONFIG/&RELPATH/&SYSPLEX/&SRVNAME/current.env'
//CEEDUMP   DD SYSOUT=*,SPIN=UNALLOC,FREE=CLOSE
//SYSOUT    DD SYSOUT=*,SPIN=UNALLOC,FREE=CLOSE
//SYSPRINT  DD SYSOUT=*,SPIN=UNALLOC,FREE=CLOSE
```

Server Region Proc:

```
//BBOASR1S PROC IWMSSNM='BBOASR1A',PARMS='-ORBsrvname ',
//      CBCONFIG='/WebSphere390/CB390'
//* See instructions at the bottom of this file
//  SET BBOLIB='BBO'
//  SET LELIB='CEE'
//  SET DB2='DB2'
//  SET RELPATH='controlinfo/envfile'
//BBOASR1S EXEC PGM=BBOSR,REGION=0M,TIME=NOLIMIT,
// PARM='/ &PARMS &IWMSSNM'
//STEPLIB  DD DSN=&BBOLIB..SBBOULIB,DISP=SHR
//*         DD DSN=&BBOLIB..SBBOLD2,DISP=SHR
//*         DD DSN=&BBOLIB..SBBOLOAD,DISP=SHR
//*         DD DSN=&LELIB..SCEERUN,DISP=SHR
//*         DD DSN=&DB2..SDSNLOAD,DISP=SHR
//BBOENV   DD PATH='&CBCONFIG/&RELPATH/&SYSPLEX/&IWMSSNM/current.env'
//CEEDUMP  DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
```

Appendix C. Notices

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

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

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

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

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

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

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

Examples in this book

The examples in this book are samples only, created by IBM Corporation. These examples are not part of any standard or IBM product and are provided to you solely for the purpose of assisting you in the development of your applications. The examples are provided ″as is.″ IBM makes no warranties express or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose, regarding the function or performance of these examples. IBM shall not be liable for any damages arising out of your use of the examples, even if they have been advised of the possibility of such damages.

These examples can be freely distributed, copied, altered, and incorporated into other software, provided that it bears the above disclaimer intact.

Disclaimer - Programming Interface information

This publication documents information that is NOT intended to be used as Programming Interfaces of WebSphere for z/OS.

Trademarks

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

CICS
DB2
IBM
IMS
IMS/ESA
MVS
OS/390
RACF
RAMAC
RMF
SecureWay
S/390
VTAM
WebSphere
z/OS

Lotus, Notes, Domino, and Lotus Go Webserver, are trademarks of the Lotus Development Corporation in the United States, other countries, or both.

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

Microsoft, ActiveX, Visual Basic, Visual C++, Windows, Windows NT, and the Windows logo are trademarks or registered trademarks of Microsoft Corporation in the United States, other countries, or both.

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

Glossary

For more information on terms used in this book, refer to one of the following sources:

- *WebSphere Application Server V4.0 for z/OS and OS/390 Glossary*, SC09-4450, located on the Internet at:

  `http://www.ibm.com/software/webservers/appserv/`
- Sun Microsystems Glossary of Java Technology-Related Terms, located on the Internet at:

  `http://java.sun.com/docs/glossary.html`

If you do not find the term you are looking for, refer to *IBM Glossary of Computing Terms*, located on the Internet at:

`http://www.ibm.com/ibm/terminology/`

or the Sun Web site, located on the Internet at:

`http://www.sun.com/`

Program Number: 5655–F31

Printed in the United States of America

SA22-7835-01