WebSphere® Application Server for z/OS V5.0

Diagnosis

GA22-7914-00

WebSphere® Application Server for z/OS V5.0

Diagnosis

GA22-7914-00

Note

Before using this information and the product it supports, be sure to read the general information under Appendix C, "Notices", on page 51.

First Edition (May 2003)

This edition applies to WebSphere Application Server V4.0.1 for z/OS and OS/390 (5655-I35), and to all subsequent releases and modifications until otherwise indicated in new editions.

The most current versions of the WebSphere Application Server V4.0.1 for z/OS and OS/390 publications and articles are at this Web site:

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

Contents

Tables

About this book

This book is intended to help customers understand the different aspects of problem determination for WebSphere Application Server V4.0.1 for z/OS and OS/390. This book contains information on diagnosing and debugging problems you might encounter when running applications in theWebSphere Application Server for z/OS environment. Topics covered include:

- Communicating with IBM when a problem occurs;
- The skills that are needed for problem determination;
- The tools needed (by systems programmers and by applications programmers) for problem determination, including those shipped with the product and z/OS tools; and
- Where to find information about problem determination-related topics.

Note: The full product name is ″WebSphere Application Server V4.0.1 for z/OS and OS/390,″ hereafter referred to in this text as WebSphere Application Server for z/OS.″

Who should read this book

This book is intended for systems programmers and applications programmers who need to diagnose and debug problems with WebSphere Application Server for z/OS. Because WebSphere Application Server for z/OS uses most advanced features and functions of the operating system, diagnosing and fixing problems might require systems programming skills in a variety of areas. This book documents the skills required for diagnosis in the z/OS environment, diagnostic controls and tools available for use, and guidelines for diagnostic procedures.

How this book is organized

The following is an overview of the chapter order and contents.

- Chapter 1, “Introduction”, on page 1 briefly introduces diagnosis terms, skills, and tools for use with WebSphere Application Server for z/OS.
- Chapter 2, “Preparing for the unexpected”, on page 5 suggests preventative measures that customers can take to avoid problems, along with suggestions for controls and tools that can be used for first-failure data capture and initial diagnosis.
- Chapter 3, “Doing your own detective work”, on page 17 provides suggestions for using tools and controls to diagnose specific errors so that you can solve problems without the help of IBM service.
- Chapter 4, “Working with IBM service”, on page 37 briefly describes tasks and tools that you might be asked to perform or use when you call IBM service to resolve a problem.
- Appendix A, “WebSphere variable definitions”, on page 43 describes WebSphere variable definitions, including default values and examples.
- Appendix B, “The error dump and cleanup interface”, on page 49 describes a WebSphere-specific tool that you can use to request a dump.
- Appendix C, “Notices”, on page 51 contains legal notices about the contents, examples, and trademarks used in this book.

Where to find related information, tools, and supplements

The WebSphere for z/OS library Web site also includes the following books in PDF format:

- *WebSphere Application Server for z/OS V5.0: License Information*, GA22-7908, which describes the license information for WebSphere for z/OS.
- *WebSphere Application Server for z/OS V5.0: Program Directory*, GI11-2825, which describes the elements of and the installation instructions for WebSphere for z/OS.
- *WebSphere Application Server for z/OS V5.0: Installation and Customization*, GA22-7909, which describes the planning, installation, and customization tasks and guidelines for WebSphere for z/OS.
- *WebSphere Application Server for z/OS V5.0: Operations and Administration*, SA22-7912, which describes z/OS system operations and administration tasks for WebSphere for z/OS and other z/OS subsystems that are configured in the WebSphere for z/OS environment. This book also includes information about improving the performance of WebSphere for z/OS and the applications it hosts.
- *WebSphere Application Server for z/OS V5.0: Messages and Codes*, GA22-7915, which describes messages and codes associated with WebSphere for z/OS.
- *WebSphere Application Server for z/OS V5.0: Diagnosis*, GA22-7915, which provides diagnosis information associated with WebSphere for z/OS.

For additional WebSphere for z/OS tools and supplements, go to the following Web site and select the download link:

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

You also might 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/`

How to send your comments

Your feedback is important in helping to provide the most accurate and highest quality information.

- To send comments on PDF books, 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 you are using, 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.

Chapter 1. Introduction

Diagnosis is a general term for problem determination and problem source identification (PD/PSI), which is the process of determining a problem and its underlying source from specific symptoms. In the multifaceted WebSphere Application Server environment, problem determination might involve not only debugging applications, but also diagnosing system problems by investigating product configurations, and by verifying the means by which the system's components interact.

A large-scale enterprise system can exhibit many symptoms that you might classify as problems. General types of problems include:

- Application or system components that immediately fail without providing any services.
- Applications or system components that have been running effectively but then fail to respond (in other words, "hang").
- Applications or system components that have been running effectively but then stop running (in other words, "crash").
- Applications or system components that have been running effectively but do not respond as expected.

These problems present themselves through various symptoms, such as an error message, wrong output, an abend, an error state, none or bad response times, or a message returned by the browser.

Another symptom that you might consider a problem is poor performance of an application or system component. Although poor performance is a problem, it is not necessarily a result of an error in design or configuration that needs to be fixed. Performance problems are solved through tuning, which is the process of adjusting applications or system components to more efficiently exploit their operating environment. For information about tuning applications and system components in the WebSphere Application Server for z/OS environment, see *WebSphere Application Server for z/OS V5.0: Operations and Administration*, SA22-7912.

Acquiring skills for problem determination

In a large-scale enterprise system such as the WebSphere Application Server for z/OS environment, diagnosis might require a variety of skills to progress from a symptom to fixing the underlying cause of that symptom. Because WebSphere Application Server for z/OS exploits many of the qualities and services that are unique to the z/OS operating system, diagnosing system-related problems might require skills in the following areas:

- Parallel sysplex
- TCP/IP
- Security Server (RACF) or the equivalent
- Database systems such as DB2 Universal Database for z/OS and OS/390
- UNIX Systems Services

You can find information for many of these topics in the publications available through the z/OS library Web site:

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

Similarly, diagnosing application-related problems might require a variety of skills because of the variety of application components that WebSphere Application Server for z/OS supports. Programmers who diagnose application problems in the WebSphere Application Server for z/OS environment need some familiarity with the following:

- The programming models and specifications for application components (Enterprise beans, Web applications, and client programs).
- The process of assembling, deploying, installing, and running server applications and clients in the WebSphere Application Server for z/OS environment.
- Various tools such as the WebSphere Application Server for z/OS error log, and the job logs for programs running on z/OS.

Working with diagnostic tools and controls

This book describes diagnostic information and controls that are specific to WebSphere Application Server for z/OS, including:

- Messages and codes for WebSphere components
- Configuration variables for routing diagnostic output
- Configuration variables for collecting trace data
- Configuration variables for setting timeout values
- z/OS modify command for dynamically changing configuration variable settings

To access and work with diagnostic information, you will use z/OS tools and controls as well. The following list summarizes z/OS tools:

- **z/OS console**

  The console displays configuration errors that cause the termination of the WebSphere Application Server for z/OS address spaces. Whatever goes to the console also goes to SYSLOG.

- **System log (SYSLOG)**

  SYSLOG is the repository for all messages that have appeared on the operator console. It also contains warning and informational messages that might be helpful after a failure has occurred.

- **Job log**

  The job log contains errors and warnings (non-termination) that are related to configuration. Anything that goes to the console and SYSLOG automatically goes to the job log.

- **System output (SYSOUT)**

  SYSOUT is a batch log that usually contains diagnostic data from the Java Virtual Machine (JVM) that runs in the servant (region). Any messages written to CError will end up in SYSOUT. In addition, SYSOUT might contain error messages that usually appear in the log stream, but were redirected to SYSOUT becasue the log stream was not available.

- **Error log**

  The error log contains messages issued through JRas support, if any. In addition, the error log usually contains messages intended for IBM use only; these are messages that support actions, problems, or issues that are usually externalized through additional messages in other sources. When you work with IBM service, you might be asked to supply the error log so that service personnel can use these support messages to diagnose the problem.

- **SYSPRINT**

  SYSPRINT contains component trace (CTRACE) output for clients, and for servants when WebSphere Application Server for z/OS is configured to use SYSPRINT rather than CTRACE buffers and data sets.

- **Component trace (CTRACE) data set**

  CTRACE data sets contain diagnostic trace entries for various processes, depending on the trace options configured for WebSphere Application Server for z/OS.
- **Logrec**

  When an error occurs, the system records information about the error in the logrec data set or the logrec log stream. The information provides you with a history of all hardware failures, selected software errors, and selected system conditions.

To find additional information about these tools, and about the process of diagnosing problems on z/OS, use the following sources:

- *z/OS MVS Diagnosis: Procedures*, GA22-7587 helps you diagnose problems in the MVS operating system, its subsystems, its components, and in applications running under the system.
- *z/OS MVS Diagnosis: Tools and Service Aids*, GA22-7589 provides detailed information about tools and service aids that can help you diagnose problems. This book contains a guide on how to select the appropriate tool or service aid for your purposes, and also provides an overview of all the tools and service aids available.
- *WebSphere Application Server for z/OS V5.0: Messages and Codes*, GA22-7915, which describes messages and codes associated with WebSphere Application Server for z/OS.

Chapter 2. Preparing for the unexpected

The purpose of this chapter is to:

- Suggest preventative measures that you can take to prevent problems from occurring,
- Suggest settings or tools that you can use so that first-failure data can be as complete as possible without affecting system performance; and
- Introduce the tools and controls that you can use for diagnosis, when a problem does occur.

Guidelines for maintaining the run-time environment

Use the following guidelines to make sure that WebSphere Application Server for z/OS is customized and maintained correctly, to support your installation's application workload. Checking these basic software and hardware requirements can help you avoid problems with the run-time environment.

- **Check that you have the necessary prerequisite software up and running.** Check that they have the proper authorizations and that the definitions are correct.
- **Check for messages that signal potential problems.** Look for warning and error messages in the following sources:
  - SYSLOG from other z/OS subsystems and products, such as TCP/IP (especially the DNS, if in use), RACF, and so on
  - WebSphere Application Server for z/OS error log
  - SYSPRINT of the WebSphere Application Server for z/OS error log
  - Component trace (CTRACE) output for the server
- **Ensure that z/OS has enough DASD space for SVC dumps.** You might have to adjust the amount of space, because it depends on the size of your applications, on the configured Java virtual machine (JVM) heap size, and on the number of servant regions that might be included in one dump, and so on. For an SVC dump of one controller and one servant, you can start with a minimum of 512, but might have to increase the MAXSPACE to 1024 or higher, given the factors listed above.
- **Check your general environment.** Does your system have enough memory? Insufficient memory problems can show up as AUX shortages, abends, or exceptions from the WebSphere Application Server for z/OS run-time. Sometimes the heap size for Language Environment (LE) and for the Java virtual machine (JVM) needs to be increased.
- **Make sure all prerequisite fixes have been installed**; a quick check for a fix can save hours of debugging.

  For the most current information on fixes and service updates, see:

  - The Preventive Service Planning (PSP) buckets for both WebSphere Application Server for z/OS and JAVA subsets of the WebSphere Application Server for z/OS Upgrade. To obtain a copy of the most current versions of these PSP buckets, you can either contact the IBM Support Center, use S/390 SoftwareXcel or link to IBMLink at the following website:

    `http://www.ibmlink.ibm.com/`
  - The Support page of the WebSphere Application Server for z/OS Web site, which contains a table of the latest authorized program analysis reports (APARs). The Support page is available through the following URL:

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

With the latest service information, check the following:

- Ensure that all prerequisite PTFs (fixes) have been applied to the system.
- Verify that all PTFs were actually present in the executables that were used at the time of error. Often, SMP can indicate that a fix is present and installed on the system when, in reality, the executables that were used at the time of error did not contain the fix.

For more details, see the following sources:

- *WebSphere Application Server for z/OS V5.0: Installation and Customization*, GA22-7909 documents hardware and software requirements, instructions for customizing WebSphere Application Server for z/OS run-time, and considerations for installing new releases or service.
- *WebSphere Application Server for z/OS V5.0: Operations and Administration*, SA22-7912 contains WebSphere Application Server for z/OS performance tuning guidelines, which can help you determine appropriate heap sizes, and so on.

Guidelines for using system controls

- You have the option of using a z/OS system logger log stream as the WebSphere Application Server for z/OS error log. The WebSphere variable `ras_log_logstreamName` identifies which log stream you want to use for the error log; it has no default setting. If you do not use a log stream, however, messages that usually appear in the error log are directed to server's job log. See "Setting up the error log" for more information about using a log stream for the error log.
- You have the option of directing trace output to SYSPRINT or buffers. The WebSphere variable `ras_trace_outputLocation` controls the location of trace output; its default values are SYSPRINT for client applications, and buffers to all other processes. Although you can change the default for other processes from buffers to SYSPRINT, performance is better when you use buffers.
- You can use the Resource Measurement Facility (RMF) to view status information that might indicate potential problems. WebSphere Application Server for z/OS uses Workload Manager (WLM) services to report transaction begin-to-end response times and execution delay times, which might indicate that changes are required for timeout values or tuning controls. For additional details, see the WLM delay monitoring topic in *WebSphere Application Server for z/OS V5.0: Operations and Administration*, SA22-7912.

Setting up the error log

WebSphere Application Server for z/OS uses an error log to record error information when an unexpected condition or failure is detected within the product's own code. Such unexpected conditions or failures include:

- Assertion failures
- Unrecoverable error conditions
- Failures related to vital resources, such as memory
- Operating system exceptions
- Programming defects in WebSphere Application Server for z/OS code.

Because WebSphere Application Server for z/OS is predefined as a z/OS system logger application, you can use a log stream as the product's error log. Doing so offers the following flexibility:

- You can direct error information to:
  - A coupling facility log stream, which provides sysplex-wide error logging, or
  - A DASD-only log stream, which provides single system-only error logging.
- You can set up a common log stream for all WebSphere Application Server for z/OS servers, or individual log streams for each application server. Local z/OS or OS/390 client ORBs can also log data in log streams. The system logger APIs are unauthorized, but logstream resources can be protected using security products such as RACF.
- You can use the WebSphere variable `ras_time_local` to control whether timestamps in the error log appear in local time or Greenwich Mean Time (GMT), which is the default.

When your installation first customizes and verifies WebSphere Application Server for z/OS installation, you have the option of defining the error log as a log stream. Using the ISPF customization dialog to configure a base application server node, you can specify log stream characteristics, including sizes. After verifying installation, you can change the log stream used for normal operations.

For additional information, see the following sources:

- *WebSphere Application Server for z/OS V5.0: Installation and Customization*, GA22-7909 contains instructions for setting up the error log as a log stream, as well as IBM defaults for the error log that appear in the ISPF customization dialog. These settings will help you determine what requirements might be necessary for additional log streams, if you decide to use anything other than the default log stream.
- "Setting output destinations and characteristics" on page 43 describes the WebSphere variables that you can use to identify the log stream for either server or client error logs. Other log-stream characteristics must be set manually.
- *z/OS MVS Setting Up a Sysplex*, SA22-7625 contains additional information about log stream requirements.
- "Viewing error log contents through the Log Browse Utility (BBORBLOG)" on page 23 describes the browse facility for viewing error log contents.

Setting up component trace (CTRACE)

WebSphere Application Server for z/OS uses z/OS component trace (CTRACE) facilities to manage the collection and storage of trace data. Unless you configure specific CTRACE controls, WebSphere Application Server for z/OS records its trace data in address-space buffers in private (pageable) storage. This data is not accessible unless a dump of the address space is taken.

Although CTRACE data is primarily output for IBM service personnel to use, exploiting CTRACE capabilities at your installation allows you to have additional trace data available when a problem first occurs. Because CTRACE efficiently uses system resources, you can collect valuable trace data with minimal impact on performance.

When your installation first customizes and verifies WebSphere Application Server for z/OS installation, you have the option of defining CTRACE controls and resources. Using the ISPF customization dialog to configure a base application server node, you can specify:

- Data sets to contain CTRACE data collected for WebSphere Application Server for z/OS.

- CTRACE writer parameters that control the writer through which trace data moves from address-space buffers into trace data sets.
- The parmlib member that connects WebSphere Application Server for z/OS address spaces to trace data sets, and optionally turns on the CTRACE writer.
- WebSphere variables that control the characteristics of trace data.

The ISPF customization dialog generates instructions for:

1. Starting the CTRACE writer
2. Starting the WebSphere Application Server for z/OS application server

Following the instructions in sequence is quite important; you can lose valuable trace data if you do not start the CTRACE writer before starting the server.

For additional information, see the following sources:

- "Steps for preparing CTRACE controls and resources".
- Procedures for starting CTRACE activity:
  - "Steps for starting CTRACE as part of WebSphere Application Server for z/OS customization" on page 10, or
  - "Steps for starting CTRACE while WebSphere Application Server for z/OS servers are active" on page 11
- "Setting trace controls" on page 44 describes the WebSphere variables that you can use to control tracing activity.
- "Viewing CTRACE and JRas data through IPCS" on page 19 describes how to use IPCS to view trace data.
- *WebSphere Application Server for z/OS V5.0: Installation and Customization*, GA22-7909 contains instructions for setting up CTRACE, as well as IBM defaults for the CTRACE writer, parmlib member, and trace data sets.
- *z/OS MVS Diagnosis: Tools and Service Aids*, GA22-7589 contains information about the advantages of using CTRACE facilities.

Steps for preparing CTRACE controls and resources

Before you start CTRACE activity for WebSphere Application Server for z/OS servers, you need to make some decisions about CTRACE controls and resources. You have the option of using default CTRACE values and resources, such as the IBM-supplied CTRACE parmlib member for WebSphere Application Server for z/OS, or you may alter default values and provide your own resources.

Perform the following steps to prepare CTRACE controls and resources:

1. Decide where you want to store CTRACE data. The location of trace data is set through the WebSphere variable `ras_trace_outputLocation`. Make a note of the value that you want to use; you will set the value later, when you start CTRACE activity.

   Tips:

   - See "Setting trace controls" on page 44 for an explanation of and default values for the `ras_trace_outputLocation` variable.
   - If you decide to use trace data sets, you can use existing or create new data sets now or later, as part of the WebSphere Application Server for z/OS customization process.
   - If you want the CTRACE data for each cell to go into separate data sets, use the `ras_trace_ctraceParms` variable described in "Setting trace controls" on page 44.

- When you are installing WebSphere Application Server for z/OS, sending trace data to SYSPRINT can be helpful; however, tracing to SYSPRINT in a production environment can cause spool space to fill up quickly. For production, you can specify a different trace output location through the WebSphere variable `ras_trace_outputLocation`.
- To separate trace data from other standard output messages, such as `System.output.println` from Java applications, you can direct CTRACE data to a separate data set. To accomplish this separation, you need to:
  - Specify a TRCFILE DD statement in the JCL start procedure (proc) for the server.
  - Set the WebSphere variable `ras_trace_outputLocation` to TRCFILE. Note that you may specify the TRCFILE value with or without additional variable values.
- Through modify commands, you have the ability to dynamically and temporarily direct trace data to SYSPRINT or TRCFILE. In other words:
  a. You can direct CTRACE data to buffers as part of normal operations.
  b. When an error occurs, you can use the modify command to send trace data to SYSPRINT or TRCFILE.
  c. Then you can use the modify command again, to dynamically reset the trace-output location

  You can complete this sequence of actions without stopping the server that is encountering the problem. For information about the modify command, see *WebSphere Application Server for z/OS V5.0: Operations and Administration*, SA22-7912.

---

2. Decide whether you want to accept defaults or provide other values for the following WebSphere variables. Make a note of the values that you want to use; you will set the values later, when you start CTRACE activity.
   - `ras_time_local`
   - `ras_trace_BufferCount`
   - `ras_trace_BufferSize`

   For defaults and valid values, see "Setting trace controls" on page 44.

---

3. Decide whether you want to use the default JCL start procedure for the CTRACE writer, or code your own JCL proc. WebSphere Application Server for z/OS provides a CTRACE writer proc named `bbowtr`. If you decide to provide your own CTRACE writer procedure, create the JCL start proc using the rules for source JCL for an external writer in *z/OS MVS Diagnosis: Tools and Service Aids*, GA22-7589. Place the start procedure in your system proclib.

---

4. Decide whether you want to use the default CTRACE parmlib member for WebSphere Application Server for z/OS, or provide your own. The WebSphere parmlib member is named CTIBBO00. If you decide to provide your own parmlib member, create one according to the following rules, and place it in your system parmlib.

   Rules:

   - You must follow the same naming convention; that is, the name must consist of the letters `CTIBBO`, but you may replace the two zeroes with any two alphanumeric characters.

- Your parmlib member must contain the following parameters:

```
TRACEOPTS
  WTRSTART(xxxxxx)
  ON
/*CONNECT TO CTRACE EXTERNAL WRITER: */
  WTR(xxxxxx)
```

Table 1. Parameters for the CTIBBOxx parmlib member

Parameter	Required or Optional?	Description
TRACEOPTS	Required and positional	This parameter must be specified first.
WTRSTART*(xxxxxx)*	Optional	This parameter automatically starts the CTRACE writer, using the specified JCL procedure. **Alternative:** You may use the operator command TRACE CT,WTRSTART=xxxxx to start the CTRACE writer. If you use this parameter in the parmlib member, and later issue the operator command, CTRACE issues an informational message to notify you that the writer already has been started.
ON	Required	This parameter must be specified before component options.
WTR*(xxxxxx)*	Required	This parameter identifies the JCL procedure to be used to start the CTRACE external writer. The specified value must match the value of the WTRSTART parameter to capture the WebSphere Application Server for z/OS trace data into a trace data set.

5. Ensure that the DLL named BBORTSS5 is in the link-pack area (LPA). For further details about loading modules into the LPA, see the recommendations on using memory in *WebSphere Application Server for z/OS V5.0: Installation and Customization*, GA22-7909.

After you have made these decisions and completed preparations, you are ready to start CTRACE activities using one of the following procedures:

- During customization of WebSphere Application Server for z/OS (see "Steps for starting CTRACE as part of WebSphere Application Server for z/OS customization"), or
- While WebSphere Application Server for z/OS servers already are running on z/OS or OS/390 (see "Steps for starting CTRACE while WebSphere Application Server for z/OS servers are active" on page 11).

Steps for starting CTRACE as part of WebSphere Application Server for z/OS customization

Before you begin: Make sure you have properly prepared CTRACE controls and resources as instructed in "Steps for preparing CTRACE controls and resources" on page 8.

Perform the following steps to start CTRACE as part of the customization process for WebSphere Application Server for z/OS:

1. Using the ISPF customization dialog to configure a base application server node, specify:
   - Trace data set characteristics
   - CTRACE writer parameters
   - The CTRACE parmlib member.
   - WebSphere variables, if you want values other than the defaults

   Result: The ISPF customization dialog generates:
   - Instructions for starting the CTRACE writer
   - Instructions for starting the WebSphere Application Server for z/OS application server

   _____

2. Start the CTRACE writer, using the generated instructions. You must start the writer first, or you might lose valuable trace data.

   _____

3. Start the WebSphere Application Server for z/OS application server, using the generated instructions.

   _____

4. When you need to collect trace data for analysis:
   a. Use the following operator command to disconnect WebSphere Application Server for z/OS from CTRACE:

      ```
      TRACE CT,ON,COMP=cell_short_name
      REPLY x,WTR=DISCONNECT,END
      ```

      where *cell_short_name* is the value specified through the ISPF customization dialog to identify the location of server configuration files. The name must be 8 or fewer characters and all uppercase.
   b. Use the following operator command to stop the CTRACE writer address space:

      ```
      TRACE CT,WTRSTOP=writer_name
      ```

      where *writer_name* is the name of the JCL start procedure for the CTRACE writer.

      _____

To view the CTRACE data, see "Viewing CTRACE and JRas data through IPCS" on page 19 for instructions.

Steps for starting CTRACE while WebSphere Application Server for z/OS servers are active

If you start a WebSphere Application Server for z/OS server before starting the CTRACE writer for WebSphere, the server still gathers data in its trace buffers. This trace data is not available for use unless you follow this procedure, or until a dump of the server address space is taken.

Before you begin: Make sure you have properly prepared CTRACE controls and resources as instructed in "Steps for preparing CTRACE controls and resources" on page 8.

Perform the following steps to start CTRACE when a WebSphere Application Server for z/OS server already is active:

1. Use the following operator command to start the CTRACE writer address space:

   ```
   TRACE CT,WTRSTART=writer_name
   ```

   where *writer_name* is the name of the JCL start procedure for the CTRACE writer that is specified in the WebSphere Application Server for z/OS CTIBBO*xx* parmlib member.

   Result: The CTRACE external writer begins writing the server's trace data to the location specified through the WebSphere variable `ras_trace_outputLocation`.

   Alternative: To connect WebSphere Application Server for z/OS to a CTRACE writer other than the one specified in the CTIBBO*xx* parmlib member, also enter these operator commands:

   ```
   TRACE CT,ON,COMP=cell_short_name
   REPLY x,WTR=writer_name,END
   ```

   where:

   - *cell_short_name* is the value specified through the ISPF customization dialog to identify the location of server configuration files. The name must be 8 or fewer characters and all uppercase.
   - *writer_name* is the name of a JCL start procedure for a CTRACE external writer. The JCL start procedure must reside in the system's proclib.

---

2. When you need to collect trace data for analysis:
   a. Use the following operator command to disconnect WebSphere Application Server for z/OS from CTRACE:

      ```
      TRACE CT,ON,COMP=cell_short_name
      REPLY x,WTR=DISCONNECT,END
      ```

   b. Use the following operator command to stop the CTRACE writer address space:

      ```
      TRACE CT,WTRSTOP=writer_name
      ```

      where *writer_name* is the name of the JCL start procedure for the CTRACE writer.

To view the CTRACE data, see "Viewing CTRACE and JRas data through IPCS" on page 19 for instructions.

Using CTRACE to collect trace data for Java server applications

Applications that run in WebSphere Application Server for z/OS can use JRas to provide tracing support that is consistent with WebSphere tracing. Instrumented applications use the JRas interfaces and classes for logging and tracing; trace data is written to the same component trace data set as the internal traces issued by the WebSphere Application Server for z/OS runtime. So you can gather application trace data in the same locations, and use the same commands to start and stop

CTRACE for these JRas applications as you do for WebSphere Application Server for z/OS server in which the applications are running.

For more information about working with applications that use JRas, see:

- "Viewing CTRACE and JRas data through IPCS" on page 19 for instructions for viewing application trace data.

Configuring WebSphere variables

WebSphere Application Server for z/OS provides configuration variables that control server behavior. Specifically, these variables allow you to control:

- Output destinations and characteristics for the error log, and for CTRACE buffers, data sets and the external writer.
- Trace buffers, data sets, and the content of trace data.
- Types of dumps to be requested.
- Timeout values for system and application behavior.

Generally speaking, the default values are designed for normal operation in a production environment. Other circumstances might require different values:

- When you first customize and verify WebSphere Application Server for z/OS installation, or
- When you test application workloads in a test environment, or
- when you encounter a problem, and need to collect more diagnostic data.

For additional information, see the following topics:

- "Steps for configuring WebSphere variables" for instructions for changing variable values.
- Appendix A, "WebSphere variable definitions", on page 43 for variable names, descriptions, and valid values.
- "Mapping of V4.0.1 environment variables to V5 WebSphere variables" on page 42 for a comparison of V4.0.1 environment variables and V5 WebSphere variables.

Steps for configuring WebSphere variables

Depending on the types of problems you encounter, you might need to change the values set for WebSphere configuration variables that control server behavior. The following procedure explains how to use the WebSphere Administrative console to change configuration variable values.

Before you begin: You should know that:

- Configuration variables may be set on a cell, node, or server level.
  - Variable values set on a cell level apply to all servers in all nodes in the cell, unless a different value for the same variable is set on a node or server level. Variable settings on a node or server level override values for the same variable set at the cell level.
  - Variables set on a node level apply to all servers in the node, unless a different value for the same variable is set on the server level. Variable settings on a server level override values for the same variable set at the node or cell level.
  - Variables set on a server level apply only to the specific server, not to any other servers in the same node or cell.

When you are diagnosing particular problems, you are most likely to alter variable values on a server level, for a particular server. Specifying variable values on the server level affects both the controller and servant regions.

- You may use scripting interfaces, instead of the WebSphere Administrative console, to alter configuration variable values.

Perform the following steps to configure WebSphere variables through the WebSphere Administrative console:

1. Click **Environment → Manage WebSphere Variables** in the console navigation tree.
2. On the **WebSphere Variables** page, select **Server** as the scope of the variable setting, and click **Apply**.
3. On the **WebSphere Variables** page, click **New**.
4. On the **Variable** page, specify a name and value for the variable. So other people can understand what the variable is used for, also specify a description for the variable. Then click **OK**.
5. Verify that the variable is shown in the list of variables.
6. Save your configuration.
7. (Optional) To have the configuration take effect, stop the server and then start the server again.

Using the z/OS modify command

You may use either the WebSphere Administrative console or the z/OS MVS console to accomplish many operations tasks related to WebSphere Application Server for z/OS servers. Entering the z/OS display or modify commands through the MVS console can provide information or perform tasks that are useful for diagnosing problems.

In particular, the z/OS modify command provides parameters that not only allow you to control WebSphere Application Server for z/OS operations, but also to:

- Display information about WebSphere Application Server for z/OS servers or servants (regions), and
- Dynamically change values related to tracing activity for a server or servant.

For additional information, see the following topics:

- "Dynamically changing diagnostic controls through the modify command" for a mapping of modify command parameters and their equivalent WebSphere variables, and
- *WebSphere Application Server for z/OS V5.0: Operations and Administration*, SA22-7912 for the modify command syntax and parameter descriptions.

Dynamically changing diagnostic controls through the modify command

The z/OS modify command provides parameters that allow you to dynamically change values related to tracing activity for a server or servant. Table 2 on page 15 lists the modify command parameters and the WebSphere variable that provides equivalent function.

Table 2. z/OS modify command parameters and their equivalent WebSphere variables

z/OS modify command parameter	Equivalent WebSphere variable
TRACEALL	ras_trace_defaultTracingLevel
TRACEBASIC	ras_trace_basic **Note:** Do not change this variable unless directed by IBM service personnel.
TRACEDETAIL	ras_trace_detail **Note:** Do not change this variable unless directed by IBM service personnel.
TRACESPECIFIC	ras_trace_specific **Note:** Do not change this variable unless directed by IBM service personnel.
TRACE_EXCLUDE_SPECIFIC	ras_trace_exclude_specific **Note:** Do not change this variable unless directed by IBM service personnel.
TRACEINIT	n/a
TRACENONE	n/a
TRACETOSYSPRINT	ras_trace_outputLocation=SYSPRINT
TRACETOTRCFILE	ras_trace_outputLocation=TRCFILE
TRACEJAVA	n/a

Chapter 3. Doing your own detective work

The topics in this chapter provide suggestions for using tools and controls to diagnose specific errors so that you can solve problems without the help of IBM service.

The following table outlines the steps you can perform to diagnose and resolve errors. For each step, sources of additional details are listed.

Table 3. Overview of diagnostic procedure, with related information sources

Diagnosis task	Sources of related information
Record initial symptoms and observations	Reviewing various sources of diagnostic data can help identify symptoms and probable causes. "Viewing diagnostic information" on page 18 provides information about tools for viewing particular types of diagnostic data.
Collect additional data, if necessary	Altering trace and dump controls enables you to collect more diagnostic information than what was gathered during the initial occurrence of the problem. Use one or more of the following: • Use the Administrative console to alter WebSphere variable settings. See: – "Steps for configuring WebSphere variables" on page 13 for instructions for changing variable values. – Appendix A, "WebSphere variable definitions", on page 43 for names, descriptions, and valid values. – "Mapping of V4.0.1 environment variables to V5 WebSphere variables" on page 42 for a comparison of V4.0.1 environment variables and V5 WebSphere variables. • Use the modify command to dynamically change trace settings. See: – "Dynamically changing diagnostic controls through the modify command" on page 14 for a mapping of modify command parameters and their equivalent WebSphere variables, and – *WebSphere Application Server for z/OS V5.0: Operations and Administration*, SA22-7912 for the modify command syntax and parameter descriptions. • Use the WebSphere Application Server for z/OS error dump and cleanup interface to request an SDUMP. See Appendix B, "The error dump and cleanup interface", on page 49.
Identify the cause of the error	"Debugging specific types of problems" on page 26

If none of the actions above resolve the problem to your satisfaction, call IBM service. For information about preparing to call IBM service, see Chapter 4, "Working with IBM service", on page 37.

Viewing diagnostic information

The following topics provide information about specific sources of diagnostic data, and the tools or resources you might need to view or work with that data.

Type of diagnostic tools or data:	Notes and instructions for use appear in:
CEEDUMPs	"Viewing CEEDUMPs in the job log"
SVC dumps	"Viewing SVC dumps"
CTRACE and JRas data	"Viewing CTRACE and JRas data through IPCS" on page 19
Error log data	"Viewing error log contents through the Log Browse Utility (BBORBLOG)" on page 23
z/OS display command	"Using the z/OS display command" on page 26
Java minor codes	"Converting Java minor codes" on page 26

Viewing CEEDUMPs in the job log

An error caught by LE or the Java run-time can result in the production of a CEEDUMP, which is written to a separate CEEDUMP specification in the job log. To view the dump contents, select the CEEDUMP portion of the output for the address space. The 'Traceback' section at the beginning of the dump can be very helpful. For additional information, see "Setting dump controls" on page 45 for the WebSphere variables you can use to control CEEDUMP contents.

Viewing SVC dumps

An SVC dump is a core dump initiated by the operating system generally when a programming exception occurs. SVC dump processing stores data in dump data sets that you pre-allocate, or that the system allocates automatically as needed.

Alternatively, you can initiate an SVC dump through the MVS console, to gather diagnostic data for a 'hang' condition, for example. SVC dumps that you initiate this way are called console dumps.

One example of an abend that could occur is the EC3 abend. WebSphere Application Server for z/OS requests an SVC dump when a controller terminates a servant (region) with an EC3 abend when timeout conditions occur.

- Your installation can set parmlib options that determine what to dump, eliminate duplicate dumps, and so on. WebSphere Application Server for z/OS provides a dump parmlib sample in `SBBOJCL(BBODMCCB)`.
- The standard SDATA expected in a SVC dump:
  `SDATA=(ALLNUC,CSA,GRSQ,LPA,LSQA,PSA,RGN,SQA,SUM,SWA,TRT),end`
- If you cannot find an SVC dump for a specific abend, your installation might be using Dump analysis and elimination (DAE) to suppress the dump. If this is the case, you can change DAE to let the dump be taken or set a SLIP on the specific abend for a particular job name if the timeout is consistently happening. For further information, see:
  - *z/OS MVS Diagnosis: Tools and Service Aids*, GA22-7589 for details about using DAE.
  - *z/OS MVS System Commands*, SA22-7627 for details about the SLIP command, which controls SLIP (serviceability level indication processing), a diagnostic aid that intercepts or traps certain system events and specifies what action to take. Using the SLIP command, you can set, modify, and delete SLIP traps.

- When you initiate a console dump:
  - When you want an SVC dump of a servant region, also request a dump of the servant's controller region.
  - Unless you suspect a particular servant region as the source of a problem, dump the controller region and all of its servant regions.
- If syslog contains a message indicating that the maxspace limit was reached for this dump, the SVC dump might be a partial one that might not contain the data you need to diagnose the timeout. This limit means that the data set used for SVC dump is not large enough, and you have to change the size to capture a complete dump.
- To view CEEDUMP contents within the SVC dump, use the IPCS verbexit LEDATA, with the CEEDUMP or NTHREADS options, to format and analyze Language Environment control blocks. For additional information, see *z/OS Language Environment Debugging Guide*, GA22-7560 for instructions for using IPCS to format and analyze CEEDUMP contents.

See *z/OS MVS Diagnosis: Tools and Service Aids*, GA22-7589 for additional information about SVC dumps.

Viewing CTRACE and JRas data through IPCS

Once activated, the WebSphere Application Server for z/OS always writes trace data into memory buffers. The number and size of these buffers is controlled using WebSphere variables.

You can get this trace data from a dump, which may be taken by the system or requested by the operator through DUMP or SLIP commands. See *z/OS MVS System Commands*, SA22-7627, for more information about the DUMP and SLIP commands.

To view messages or application trace data from Component Trace, you must use the interactive problem control system (IPCS) to format the data. The source of the trace data can be a dump data set or a trace data set, and the command to use would be `IPCS CTRACE`. You can also use the `IPCS CTRACE` command to merge multiple trace entities together such as multiple WebSphere Application Server for z/OS address space traces, OMVS, and TCPIP.

z/OS MVS IPCS Commands, SA22-7594, describes how to use `IPCS CTRACE` and `IPCS MERGE`.

Steps for using the IPCS dialog to format CTRACE data

Before you begin: When setting up IPCS, your installation may customize IPCS for its users. IBM recommends providing access to the IPCS dialog through an ISPF panel. If your installation has not customized IPCS as recommended, you need to start the IPCS dialog. See *z/OS MVS IPCS User's Guide*, SA22-7596, to find out how to start the IPCS dialog.

Perform the following steps to use the IPCS dialog to format application trace data:

1. From the IPCS Primary Option Menu panel, select option 6 (`COMMAND`).

   ---

2. On the IPCS Subcommand Entry panel:
   a. (Optional) Issue the `SETDEF` subcommand to determine the default values for routing displays.
   b. Enter the `CTRACE` command, with the following required parameters:
      `CTRACE COMP(`*`cell_short_name`*`)`

where *cell_short_name* is the value specified through the ISPF customization dialog to identify the location of server configuration files. The name must be 8 or fewer characters and all uppercase.

Note: If you were interested in only JRAS data, you would enter the following:

```
CTRACE COMP(cell_short_name)USEREXIT(JRAS)
```

Specify additional parameters as necessary.

Example: To direct trace data to the terminal only, you would append the `NOPRINT` and `TERMINAL` parameters to the `CTRACE` command.

Tip: For a complete list of CTRACE command parameters, see *z/OS MVS IPCS Commands*, SA22-7594.

---

3. View your application's data, basing the method you choose on which one is appropriate for the location of the data:

If you directed output to the...	Then use the...
IPCS print data set (IPCSPRNT)	ISPF/PDF Browse option
Terminal	Dump Display Reporter panel

Tip: To navigate through the trace data on the Dump Display Reporter panel, use the commands and PF keys listed in *z/OS MVS IPCS User's Guide*, SA22-7596.

Finding the subname for IPCS CTRACE

If the trace data set is an SVC dump, the trace subname must also be specified. This subname is the aggregation of the address space's jobname with its ASID (address space identifier), in printable hexadecimal. An easy way to determine the subname is to query CTRACE for the data using the following IPCS subcommand:

```
CTRACE QUERY DSN('dump.data.set')
```

Once you get the subname you can view the WebSphere Application Server for z/OS trace data with the following IPCS subcommand:

```
CTRACE COMP(cell_short_name) SUB((subname)) FULL DSN('dump.data.set')
```

where *cell_short_name* is the value specified through the ISPF customization dialog to identify the location of server configuration files. The name must be 8 or fewer characters and all uppercase.

Note: The `subname` parameter is optional for only the trace data set. It is required when viewing the trace data using the dump data set.

Viewing multiple traces

CTRACE enables you to view multiple traces together with the trace data from the various sources intermixed based on the time stamp. See *z/OS MVS IPCS Commands*, SA22-7594, for specifics on using this `MERGE` subcommand.

Steps for using IPCS in batch mode to format CTRACE data

To view messages or application trace data from Component Trace, you must use the interactive problem control system (IPCS) to format the data. Using IPCS in batch mode is the easiest method of formatting data, especially if you do not have much experience with using IPCS, TSO/E and ISPF. Through batch mode, you can use IPCS to format trace data and write it to an MVS data set. Optionally, you may copy the contents of that data set into an HFS file for viewing.

Before you begin: You must create an IPCS dump directory before you can use IPCS in batch mode. When setting up IPCS, your installation may customize IPCS for its users. This customization can include modifying the IBM-supplied BLSCDDIR CLIST with default values for creating an IPCS dump directory.

If your installation has modified the BLSCDDIR CLIST, perform the following steps to create an IPCS dump directory:

1. Decide on a fully-qualified data set name for the directory.
2. From the TSO/E command prompt, enter the `BLSCDDIR` command, specifying the data set name. For example, to create a dump directory named IBMUSER.DDIR, enter:

   ```
   %blscddir dsn('ibmuser.ddir')
   ```

If your installation has not customized IPCS, you might need to alter other BLSCDDIR CLIST parameters. See *z/OS MVS IPCS User's Guide*, SA22-7596 and *z/OS MVS IPCS Commands*, SA22-7594 for more details about using the BLSCDDIR CLIST to create a dump directory.

Perform the following steps to use IPCS in batch mode to format application trace data:

1. Create a file and copy the following sample JCL into it. This JCL invokes IPCS to extract and format JRAS trace data and write it into an MVS data set, and then uses the `TSO/E OPUT` command to copy the formatted data from the MVS data set into an HFS file.

   ```
   //IBMUSERX   JOB ,
   // CLASS=J,NOTIFY=&SYSUID,MSGCLASS=H
   //IPCS       EXEC PGM=IKJEFT01,REGION=4096K,DYNAMNBR=50
   //IPCSDDIR   DD DSN=IBMUSER.DDIR,DISP=SHR
   //IPCSDOC    DD SYSOUT=H
   //JRASTRC    DD DSN=IBMUSER.CB390.CTRACE,DISP=SHR
   //IPCSPRNT   DD DSN=IBMUSER.IPCS.OUT,DISP=OLD
   //SYSTSPRT   DD SYSOUT=*
   //SYSTSIN    DD *
   IPCS
   DROPDUMP DDNAME(JRASTRC)
   PROFILE LINESIZE(80)PAGESIZE(99999999)
   SETDEF NOCONFIRM
   CTRACE COMP(SYSBBOSS) DDNAME(JRASTRC) FULL PRINT +
          NOTERMINAL
   DROPDUMP DDNAME(JRASTRC)
   END
   /*
   //OPUT       EXEC PGM=IKJEFT01,REGION=4096K,DYNAMNBR=50
   //SYSTSPRT   DD SYSOUT=*
   //SYSTSIN    DD *
   oput 'ibmuser.ipcs.out' '/u/ibmuser/ipcs/jrastrace.txt' TEXT
   /*
   ```

   ---

2. Edit the sample JCL to replace `IBMUSER.DDIR` with the data set name that you used for the IPCS dump directory you created.

Notes:

a. Use the PAGESIZE parameter on the PROFILE statement only if you do not want to print the output data set.

b. You may replace the HFS file name with the name of an existing HFS file, but you do not have to do so. The OPUT command processing will create a new HFS file, if the one specified does not exist, and grants read and write access to that file for your user ID only.

   If you do specify an existing HFS file, the OPUT command processing will write over any data that is already in that file. If you want to know more about the OPUT command, see *z/OS UNIX System Services Command Reference*, SA22-7802.

c. Change the data set name specified on the JRASTRC DD in the example to the name of the data set containing the CTRACE data.

d. Change the name of the MVS data set on both the JRASTRC DD statement and the OPUT command in the SYSTSIN stream, as necessary. The formatted output of the JRAS CTRACE data is first written to the MVS data set specified by the IPCSPRNT DD statement and then (optionally) copied to the HFS data set. You must either pre-allocate this data set, or change the sample JCL to allocate the data set. This data set should have a record format of VBA and a record length of 133.

---

3. Submit the JCL to start the IPCS batch job.

Once you are done you can use a UNIX editor, such as vi, to view your trace data in the HFS file. If you want to know more about the UNIX editors, see *z/OS UNIX System Services User's Guide*, SA22-7801.

Sample JCL to display WebSphere for z/OS trace data: The following sample shows JCL that displays WebSphere for z/OS trace data.

Note: The JCL uses an IPCS dump directory (in VSAM data set userid.DUMP.DIR) that must be allocated before you run the JCL. See *z/OS MVS IPCS Commands*, SA22-7594, for information about initializing a dump directory.

```
//SHOWTRC  JOB <job card info>
//JOBLIB   DD DISP=SHR,DSN=BBO.SBBOMIG
//       DD DISP=SHR,DSN=SYS1.MIGLIB
//PRINTIT EXEC PGM=IKJEFT01,REGION=0M
//IPCSDDIR DD DISP=(OLD,KEEP),DSN=userid.DUMP.DIR
//IPCSPARM DD DISP=SHR,DSN=SYS1.PARMLIB
//SYSTSPRT DD SYSOUT=*
//IPCSTOC  DD SYSOUT=*
//IPCSPRNT DD SYSOUT=*
//*-------------------------
//SYSTSIN DD *
IPCS NOPARM
  CTRACE COMP(SYSBBOSS) SUB((subname)) FULL DSN('dump.data.set')
/*
```

The following example shows JCL that displays WebSphere for z/OS trace data for multiple address spaces.

```
//SHOWTRC2 JOB <job card info>
//JOBLIB   DD DISP=SHR,DSN=BBO.SBBOMIG
//       DD DISP=SHR,DSN=SYS1.MIGLIB
//PRINTIT EXEC PGM=IKJEFT01,REGION=0M
//IPCSDDIR DD DISP=(OLD,KEEP),DSN=userid.DUMP.DIR
```

```
//IPCSPARM DD DISP=SHR,DSN=SYS1.PARMLIB
//SYSTSPRT DD SYSOUT=*
//IPCSTOC  DD SYSOUT=*
//IPCSPRNT DD SYSOUT=*
//*--------------------------
//SYSTSIN DD *
IPCS NOPARM
  MERGE
  CTRACE COMP(SYSBBOSS) SUB((subname)) FULL DSN('dump.data.set')
  CTRACE COMP(SYSBBOSS) SUB((subname2)) FULL DSN('dump.data.set')
  MERGEEND
/*
```

Viewing error log contents through the Log Browse Utility (BBORBLOG)

You can use the Log Browse Utility (BBORBLOG) to view the error log stream. If you need to look at the WebSphere Application Server for z/OS error logstream, use ISPF option 6 to enter the command:

```
ex 'BBO.SBBOEXEC(BBORBLOG)'  'BBO.BOSSXXXX '
```

the log-stream name is assumed to be `BBO.BOSSXXXX`

The space allocation and the unit for the allocation are contained within the rexx code. If you keep a large amount of trace data, the allocation must be made larger.

The WebSphere Application Server for z/OS provides an ISPF REXX EXEC named BBORBLOG, that allows you to browse the error log stream.

Notes:

1. By default, the macro formats the error records to fit a 3270 display.
2. Timestamps are in Greenwich Mean Time (GMT) unless changed by setting WebSphere variable `ras_time_local` to 1.
3. Message BBOJ0051I, which appears in the job output can help correlate error-log entries to the proper job output.

Using the log browse utility (BBORBLOG)

You can view the error log stream output using the BBORBLOG browser. To invoke the browser, go to ISPF option 6 and enter:

```
'BBO.SBBOEXEC(BBORBLOG)'  'BBO.BOSSXXXX   format option '
```

Note: In this example, `BBORBLOG` resides in `BBO.SBBOEXEC`.

The browser creates a browse data set named "`userid.stream_name`", which contains the contents of the log stream. When the browser is executed, it:

1. Allocates a data set called `userid.stream_name`, which overwrites any duplicate data sets.
2. Populates the data set with the contents of the log stream.
3. Puts the user in "browse" mode on the data set.

Note: Each time `BBORBLOG` is invoked a static file is created which overwrites the existing file. In order to refresh the file, it is necessary to re-issue `BBORBLOG`.

The browser takes two parameters:

log stream name
The name of the log stream. See the job messages for the name of the log stream.

format option

80 The default. The log stream record will be formatted on a lrecl length of 80 characters. Additional lines will be wrapped.

NOFORMAT
Turns off formatting. The error log message appears as one log message string in the browse file.

There are three valid ways (three separate commands to use) to invoke the browser. We will illustrate each of these using the following example:

Example: If the BBORBLOG member was in a data set named BBO.SBBOEXEC, then you would issue one of the following depending on your chosen format option:

```
ex 'BBO.SBBOEXEC(BBORBLOG)'  'BBO.BOSSXXXX '
ex 'BBO.SBBOEXEC(BBORBLOG)'  'BBO.BOSSXXXX  80'
ex 'BBO.SBBOEXEC(BBORBLOG)'  'BBO.BOSSXXXX   NOFORMAT '
```

Tip: (For using BBORBLOG): If the target library in the BBO.SBBOEXEC example above was added to the SYSEXEC concatenation of the user logon procedure during the WebSphere Application Server for z/OS installation, it would be easiest to invoke the browser. You would not have to specify the library containing the browser REXX EXEC—you would only need to specify BBORBLOG.

Error log stream record output

There are two error log stream records that we will look at:

- Server logstream
- CERR of a server.

Note: The numbers to the left of each sample were added to specify lines—they will not be in the actual output.

Sample output from a server logstream:

```
1|  2000/06/01 16:01:06.683 01 SYSTEM=SY1 SERVER=BBOASR1A JobName=BBOASR1S
2|            ASID=0X0033 PID=0X0100003C TID=0X24F858A0 0X000004 c=2.1010030
3|            ./bbooreq.cpp+4437 ... BBOU0013W The function
4|            make_user_exception(IIOP_protocolArea*)+4437 raised a user exception
5|            CosNaming::NamingContext::NotFound.
```

The log stream record output fields from stream BBO.BOSSXXXX are:

Table 4. Parts table for a server logstream record output

Component	Description
line 1: 2000/06/01 16:01:06.683 01	Date / timestamp / 2-digit record version number
line 1: SYSTEM=SY1	System name
line 1: SERVER=BBOASR1A	Server name
line 1: JobName=BBOASR1S	Jobname
line 2: ASID=0X0033	ASID (address space identifier)
line 2: PID=0X0100003C	PID (Process ID)

Table 4. Parts table for a server logstream record output (continued)

Component	Description
line 2: `TID=0X24F858A0 0X000004`	TID (Thread ID)
line 2: `c=2.1010030`	Request correlation information
line 3: `./bbooreq.cpp+4437`	File name & line
line 3: `BBOU0013W`	Log message number
line 3: `The function...`	Log message
lines 4–5: `make_user_exception... CosNaming::Naming...`	Continuation lines: Continuation of the Log Stream log message

Note: Each field is delimited by a blank.

Sample output from CERR of a server:

```
1  BossLog: { 0017} 2000/06/01 15:58:25.557 01 SYSTEM=SY1 SERVER=BBOASR1A
2     PID=0X0100003C TID=0X24F82920 00000000  c=3.C5D02
3     ./bboiroot.cpp+1195 ... BBOU0012W The function IRootHomeImpl::findHome(
4     const char*)+1195 received CORBA system exception CORBA::INTERNAL.
5     Error code is C9C21200.
```

The CERR job message output fields are:

Table 5. Parts table for a CERR record output

Component	Description
line 1: `BossLog: { 0017}`	`BossLog`: {entry number}
line 1: `2000/06/01 15:58:25.557 01`	Date / timestamp / 2-digit record version number
line 1: `SYSTEM=SY1`	System name
line 1: `SERVER=BBOASR1A`	Server name
line 2: `PID=0X0100003C`	PID (Process ID)
line 2: `TID=0X24F82920 00000000`	TID (Thread ID)
line 2: `c=3.C5D02`	Request correlation information
line 3: `./bboiroot.cpp+1195`	File name & line
line 3: `BBOU0012W`	Log message number
line 3: `The function IRootHomeImpl::find...`	Log message
lines 4–5: `const char*)+1195 received CORBA system exception CORBA::INTERNAL. Error code is C9C21200.`	Continuation lines: Continuation lines of the CERR job message

Notes:

1. Each field is delimited by a blank.
2. The CERR format is found in SYSOUT, not the logger.

Saving your BBORBLOG browser output

When you use the BBORBLOG browser, it creates a data set with your user ID followed by the log stream name. You should rename it if you wish to save your browser output. The contents of the current view of the log stream will remain until the stream reaches its retention date. The next time you invoke the browser, however, the current view of the log stream will be deleted (because it uses the same data set name). The previous data will exist in another record (not the current view) until its retention date.

Using the z/OS display command

You may use either the WebSphere Administrative console or the z/OS MVS console to accomplish many operations tasks related to WebSphere Application Server for z/OS servers. Entering the z/OS display or modify commands through the MVS console can provide information or perform tasks that are useful for diagnosing problems.

In particular, the z/OS display command provides parameters that allow you to display information about the following:

- Servers
- Servant regions
- Sessions
- Trace settings
- JVM heap statistics

For additional information, see *WebSphere Application Server for z/OS V5.0: Operations and Administration*, SA22-7912 for the display command syntax and parameter descriptions.

Converting Java minor codes

Occasionally, Java will take an WebSphere Application Server for z/OS error code (C9C2xxxx in hexadecimal) and convert it to a very large negative number. If you get a very large negative number, try converting it back to hexadecimal to find the correct code.

To convert the error codes back to hexadecimal:

- Add 2^{32} to the negative number and convert it into hexadecimal. This can be done using the OMVS command "bc".

  Example: Suppose you get the error code "910022649":

  1. Under OMVS, type the command:

     ```
     bc
     ```

  2. then type:

     ```
     obase=16
     2^32 - 910022649
     quit
     ```

- The bc program displays C9C22807, which is the hex value that you should look up.

Debugging specific types of problems

The following topics provide information for specific types of problems that you might encounter in the WebSphere Application Server for z/OS environment.

Debugging client exceptions

Start with the client and work your way backward to find the problem. When tracing exceptions back to the original problem, be aware that the RMI/IIOP protocol requires that some exceptions undergo conversion from one type to another as the exception passes through the runtime. Usually this transformation is between CORBA::SystemExceptions and RMI RemoteExceptions. Pay special attention to the CORBA::SystemException minor codes which indicate that a type transformation has occurred.

Caused by:	System exception (thrown by runtime)	User exception (thrown by application code)
Look for:	• CEEDUMPs in controller (region) or servant (region). These dumps indicate that the runtime had an error • JRAS error log entries, which can narrow the error down the exception to a specific function within the runtime	• CEEDUMPs • JRAS error log entries and traces
Actions:	• Look at the minor code that is listed. • Look for fixes that address similar symptoms or minor codes. • System exceptions usually represent the detection of an unexpected error, and therefore (unless directed by the documentation of the minor code) will often require IBM assistance to identify the problem.	• Look at your application for any sign of error. • Look for system failures, such as a system exception in the controller (region). If you find a system exception, follow the steps to the left for diagnosing a system exception.

Debugging applications that hang

Possible cause:	The WebSphere variable `transaction_defaultTimeout` might have a value too large.	
Caused by:	Loop in the application	JVM runs out of heap storage, if you are running Java in a servant (region)

Look for:	• Environment variable that handles how long the application runs before timeout • Timeout-related minor codes: – C9C21047 – C9C2110F – C9C21110 – C9C21111 – C9C21112 – C9C21113 – C9C21114 – C9C21190 – C9C21191 – C9C21192 – C9C21809 – C9C21892 – C9C21893 – C9C22013 • ABEND EC3, reason codes 0413002 through 04130007 • resource messages on the console **Example:** DB2 deadlock messages on the z/OS console • A wait beyond the timeout value length with no timeout	• Any error messages from JVM in the job log of the failed servant (region)
Actions:	• Analyze with IPCS to determine whether or not the servant (region) was looping (application code loop) or waiting (maybe the runtime failed). – Use the `DUMP` command to get a console dump of the servant and its controller. • If you were utilizing JRAS, look at the JRAS CTRACE entries: – If the application code was looping, you may see the same entry repeating. • Ensure that CTRACE writer is on and take a SVC dump at the approximate time of hang. • Use the display command to determine the state of the server. See "Using the z/OS display command" on page 26 for additional information.	• Through the Administrative console, set the WebSphere variable to debug the JVM; this setting passes information to the JVM and turns on the high-level messages for you to examine. • Look for error message or Java stack traces that might indicate an `OUT_OF_MEMORY` condition. • Use application monitoring tools, such as WebSphere Studio Application Monitor (WSAM) or Jinsight, to look for application memory leaks.

Resolving timeout conditions

In such a complex environment as WebSphere Application Server for z/OS, timeouts might occur for many different reasons. Although you can alter timeout values, you should not do so until you understand why the timeout occurs.

Depending on the timeout condition, you might be able to permanently fix the timeout condition by doing some system or application tuning. For example, if the diagnostic data indicates throughput problems, you can alter the number of server regions, the number of threads within each server region, or the use of replicated servers.*WebSphere Application Server for z/OS V5.0: Operations and Administration*, SA22-7912 contains performance tuning guidelines for:

- Applications running in the WebSphere Application Server for z/OS run-time,
- The WebSphere Application Server for z/OS run-time itself,
- The z/OS operating system, and
- z/OS subsystems or other products that might be involved in processing a particular application request.

Generally speaking, increasing the timeout values should be your last resort, or only a temporary action taken to prevent multiple timeout-abend dumps from causing system performance problems. If you increase timeout values without properly diagnosing the timeout condition, the only results you might see are less frequent abends and dumps for the same timeout condition, or slower system or application performance. The following topics provide background information that can help you resolve timeout conditions:

- "Understanding how timers work"
- "Guidelines for analyzing diagnostic data for timeout conditions" on page 30
- "Identifying possible causes of and fixes for timeout conditions" on page 32
- "Guidelines for altering timeout values" on page 33

Understanding how timers work

Timers define a limit to the amount of time required for a specific operation to complete. When the timer begins its countdown depends on type of operation it controls. The timers that WebSphere Application Server for z/OS uses can be classified into the general types described in Table 6 on page 30; the specific timers themselves are described in "Controlling behavior through timeout values" on page 46. Most of the timers have a default value that defines a reasonable limit for the particular operation to complete. When the timer pops (that is, reaches the time limit), WebSphere Application Server for z/OS takes one of the following actions:

- Sends a minor code to the client for timers that pop before the client request is dispatched to a servant region.
- Abnormally ends the servant region with an EC3 ABEND for timers that pop while the client request is being processed by an application component running in the servant region. All threads in the abending servant region will be terminated.

  WebSphere terminates the servant region to prevent the application from tying up resources, thus causing other requests to start backing up. Once the servant is terminated, WLM starts a new servant to take its place and continue processing requests from the controller.

Different types of timers might be counting down simultaneously, because the operations they control might overlap to a certain degree. For example, suppose the application server receives an IIOP client request that will be processed by an application component that uses transaction support. In this case, both of the following WebSphere timers can be counting down simultaneously:

- `control_region_wlm_dispatch_timeout`, which limits both the amount of time a client request waits on the WLM queue, as well as the time required for the application component to process the request; and

- `transaction_defaultTimeout`, which limits the amount of time the controller will wait for a transaction to be either committed or rolled back.

These timers overlap only for the time during which the application's transaction is being processed. To determine which timer cause the error, you can use the symptoms– specific minor codes or EC3 ABEND reason codes.

Table 6. General types of timers and the operations they control

General type	Timer processing	Timeout symptoms
Input	Input timers define limits for receiving a complete request; the countdown starts when a connection to the Java server is established. The communication protocol (HTTP, HTTPS) determines the timer used for the request.	The session is closed.
Session	Session timers define limits for the use of session connections. These timers start the countdown as soon as a session becomes idle.	The session is closed.
WLM dispatch	Dispatch timers control how long a complete client request waits to be dispatched in a servant region for processing. The countdown starts when the controller places the request on the WLM queue. Depending on the specific timer, the time limit can include not only wait time on the WLM queue, but also the time required for processing a response to the client request.	Message BBOO0232W and an EC3 ABEND in the servant (region), with one of these accompanying reason codes: 04130003 04130004 04130006
Transaction	Transaction timers define how long: • An application or controller will wait for one transaction to complete. The countdown starts when the container starts a transaction on behalf of the application component. • A controller will attempt to recover in-doubt transactions during peer restart and recovery mode.	• Message BBOT0003W or BBOO0232W • An EC3 ABEND in the servant (region), with one of these accompanying reason codes: 04130002 04130005
Output	Output timers define how long a controller will wait to receive output for a client request. The countdown starts when the client request is dispatched to the servant region for processing. The communication protocol (HTTP, HTTPS) determines the timer used for the request.	Message BBOO0232W and an EC3 ABEND in the servant (region), with reason code `04130007`

Guidelines for analyzing diagnostic data for timeout conditions

The following guidelines provide instructions for finding diagnostic data in an SVC dump that can help you determine what timeout condition occurred:

- Find the task with the EC3 abend:
  1. Format the TCB summary for the servant that was timed out by entering the following command:

     ```
     ip summ format asid(x'address')
     ```

     where *address* is the address space ID of the servant.

Find the TCB that had the EC3 completion code. Ignore any EC3 completion code on the ″main″ thread which is the 4th TCB listed in the summary format (the 1st one after the 3 MVS TCBs). The WebSphere main thread is the one that is waiting in BBO_BOA::impl_is_ready. No application requests are ever dispatched on this thread, therefore there is nothing to timeout. During timeout processing the main thread for the server region is also abended with EC3 as a mechanism of bringing the address space down. Thus the reason why the EC3 completion code may appear on the main thread. This is never the cause of a timeout though, only a result of timeout processing.

2. If there is no EC3 completion code in the TCB summary, look in systrace. Format the systrace in gmt time since the other timestamps you'll be comparing it to are in gmt time. To format in gmt time, enter the following command::

   ```
   ip systrace all time(gmt)
   ```

   You may not see the EC3 abend in systrace either as systrace can cover a small amount of time.
3. You can also try looking in ip verbx mtrace or in syslog to see when the EC3 abend occurred. You'll need this time to determine the 'end' time of the request which is the gmt time the timeout value was reached.

- Determine what timeout values are in effect by checking the reason code associated with the EC3 abend.

Table 7. Abend EC3 reason codes and their explanations

Reason code	Explanation
04130002	The controller issued an ABTERM for this servant region because a transaction timeout ocurred. Code under dispatch could have been in a tight loop.
04130003	The controller issued an ABTERM for this servant region because it was hung trying to move a controller request into the servant region. The target request was timed out, but the servant was currently copying the request. The controller checked the servant for progress at regular intervals, before taking action by issuing an ABTERM.
04130004	The controller issued a ABTERM for this servant region because the WLM queue timeout occurred. Code under dispatch could have been in a tight loop.
04130005	The controller issued an ABTERM for this servant region because a transaction timeout ocurred. The transaction has timed out, but no current request associated with the transaction was found. The servant associated with the transaction will be terminated.
04130006	A controller thread encountered a problem while processing a request. The request has been queued to WLM and associated with a servant region. The termination of the associated servant region is needed to complete cleanup for the request.
04130007	The controller issued a ABTERM for this servant region because the HTTP OUTPUT timeout occurred. Code under dispatch could have been in a tight loop.

- Find the method name to determine if it was `httpRequest`, `httpsRequest` or `DispatchbyURI` or some other method.

  If the request is not specifically a request that came through the HTTP or HTTPS transport handlers, the `protocol_http_output_timeout` (HTTP) and `protocol_https_timeout_output` (HTTPS) timeout values will not be a factor. In

other words, when the request is a `DispatchbyURI` method, the request is received through the RMI/IIOP protocol, so the `protocol_http*` variables have no affect.

- Obtain the callback stack for the request, using the IPCS verbexit LEDATA, with the CEEDUMP or NTHREADS option.

Identifying possible causes of and fixes for timeout conditions

The timer that expires first might not indicate the actual problem that needs to be fixed. To properly diagnose timeout conditions, you should know all of the timer values that might be in effect for a particular servant region.

Table 8. Possible causes of and solutions for timeout conditions

General type of timer	Possible causes	Possible solutions
Input	The client sent only part of the data and was delayed in sending the rest.	The application on the client side may want to consider having retry logic in place if it does receive a timeout minor code in return.
Session	The session is idle through lack of use.	If you consider losing idle sessions to be a problem, increase the values of the persistent-session timeouts, or use the session more frequently.
WLM dispatch	No threads are free to pick up the request because of one of the following conditions: • The threads are all busy processing requests. • The currently executing threads are waiting for a response from DB2, WebSphere MQ, another server, and so on. In this case, look for messages indicating contention for resources; for example, on the z/OS console, you might see messages about DB2 deadlocks. In either case, the request times out waiting in the WLM queue to be dispatched in a servant (region).	The case where the threads are all busy processing requests could indicate one of the following conditions: • The number of servant regions that WLM may start is set too low (the number is set through WebSphere variable `wlm_maximumSRCount`). • The number of threads allowed within a servant region is set too low (the number is controlled by the Isolation Policy setting in Administrative console or WebSphere variable `server_region_workload_profile`). • You need to replicate servers to handle the amount of incoming work. All of these conditions indicate that performance tuning might be necessary. See the performance topics in *WebSphere Application Server for z/OS V5.0: Operations and Administration*, SA22-7912.
Transaction	Possible causes of transaction timeouts include: • The same as those for WLM dispatch timeouts, or • Delays that prevent the transaction coordinator from committing or rolling back a transaction within the allotted time.	See the possible solutions for WLM dispatch timeouts. In addition, you can look for messages indicating contention for resources that are involved in the transaction that timed out.
Output	Possible causes of output timeouts are the same as those for WLM dispatch (dispatch is for IIOP, output is for HTTP).	See the possible solutions for WLM dispatch timeouts. In addition, you can use the WebSphere variable `protocol_accept_http_work_after_min_srs=1` to prevent the HTTP transport handler from dispatching requests until WLM starts a minimum number of servant regions.

Guidelines for altering timeout values

Generally speaking, increasing the timeout values should be your last resort, or only a temporary action taken to prevent multiple timeout-abend dumps from causing system performance problems. If you increase timeout values without properly diagnosing the timeout condition, the only results you might see are less frequent abends and dumps for the same timeout condition, or slower system or application performance. When you do have to alter timeout values, however, use the information in Table 9 to help you decide what new values might be appropriate.

Table 9. Common timer variables and tools for monitoring these timeout conditions

WebSphere variable and its relationship, if any, to other timers	How to monitor processing for this type of timeout condition:	If you want to adjust the value, consider the following:
control_region_wlm_dispatch_ timeout For HTTP work, the WLM timer is not set and only the `HTTP_OUTPUT_TIMEOUT` is in effect (covering the entire dispatch window)	SMF provides data on WLM queue time	How long work takes to get to a servant depends on the number of servants that WLM starts, how many you let it start (wlm_maximumSRCount), how many service classes the work is spread across, how much work you're getting, and so on
protocol_http_ timeout_ input None.	This behavior is not easily monitored. Turning on a trace point would indicate whether a client failed because of this input timeout setting, but tracing has performance consequences.	• How long are you willing to allow a control region worker thread to be blocked while it is waiting for a message? • How big are incoming HTTP requests? The larger they are, the longer it might take to get the whole request through the network.
protocol_http_timeout_ output If the application component starts transactions, then the transaction timers also might be involved.	This behavior is not easily monitored, but the controller will terminate the servant (region) with abend EC3 for this timeout condition.	• How long are you willing to let a client hang waiting for a response? • How long are you willing to let a thread in a servant (region) be tied up working on a response before concluding that the request has taken too long? • If you have multiple application threads in the servant (region), all of them will be terminated when only one of them times out. This loss of work might make you want to allow these timeouts to occur less frequently.
protocol_http_timeout_ persistentSession None. All the other timers relate to work processing, whereas this one relates to what happens when there is no work.	None.	How much time passes between requests vs. how much does it cost to establish a new session. You would want to keep idle sessions around for a while to avoid the startup cost of a new session, but don't want to keep them forever as resource usage accumulation will eventually be a problem.
protocol_https_timeout_ input	See the information for the `protocol_http_timeout_input` variable. This value applies in exactly the same way to work that comes in over the HTTPS port.	
protocol_https_timeout_output	See the information for the `protocol_http_timeout_output` variable. This value applies in exactly the same way to work that comes in over the HTTPS port.	

Table 9. Common timer variables and tools for monitoring these timeout conditions (continued)

WebSphere variable and its relationship, if any, to other timers	How to monitor processing for this type of timeout condition:	If you want to adjust the value, consider the following:
protocol_iiop_local_ timeout None. This variable is a client-side timeout, and IIOP only.	None, other than to observe the timeouts occuring on the client side.	How long are you willing to let the client block?
protocol_iiop_server_session_ keepalive **protocol_iiop_server_session_ keepalive_ssl** None. These variables relate to session activity during idle periods and only for IIOP, so these timers do not interact with the `protocol_http_timeout_ persistentSession` timer.	You should read TCP/IP APAR PQ18618 for information about the `SOCK_TCP_KEEPALIVE` values and their consequences.	Is it useful to have idle sessions timeout? They normally don't which can consume resources. However, detecting a timeout requires network traffic between TCP/IP stacks. Creating traffic on otherwise idle sessions may have network consequences you don't want.
transaction_ defaultTimeout This variable can be overriden by applications up to the maximum indicated by the `transaction_maximumTimeout` variable, which limits the amount of time an application can set for its transactions to complete. Output timers also might cause work to time out, but the transaction timers and output timers are not aware of each other.	The controller issues message BBOT0003W to indicate a timeout condition, and terminates the servant (region) with abend EC3 reason codes 04130002 or 04130005.	• How long are you willing to let a client hang waiting for a response? • How long are you willing to let a thread in a servant (region) be tied up working on a response before concluding that the request has taken too long? • If you have multiple application threads in the servant (region), all of them will be terminated when only one of them times out. This loss of work might make you want to allow these timeouts to occur less frequently.
transaction_ maximumTimeout If set, this variable limits the amount of time an application can set for its transactions to complete. If the `transaction_maximumTimeout` variable is not set, application transactions are controlled by the time limit set on the `transaction_ defaultTimeout` variable.	None.	Same considerations as for `transaction_ defaultTimeout`
transaction_ recoveryTimeout None	None.	Locks are held while one controller (region) waits for other controllers that are required to resolve in-doubt transactions. How long can you afford to have these resources held?

Debugging problems related to Java Message Service (JMS) support

You might encounter JMS-related errors in the WebSphere Application Server for z/OS environment. To debug these errors, use the following:

- WebSphere variables that control the type of trace data collected. Use the procedure in "Steps for configuring WebSphere variables" on page 13 to set these variables:

  com.ibm.ejs.*=all=enabled
  Turns on all container tracing

com.ibm.ejs.j2c.*=all=enabled
Turns on all tracing for connector support in WebSphere Application Server for z/OS

Messaging=all=enabled
Collects trace data for the JMS and Message-driven bean (MDB) components of WebSphere Application Server for z/OS

If your installation configures WebSphere MQ to provide Java Message Service (JMS) support, you might need to use specific MQ tools for diagnosis:

- WebSphere variable `JMSApi=all=enabled`, which turns on all tracing for applications that use the JMS application programming interface (API).
- MQSC commands, which allow you to display information and perform other operations
- The CSQUTIL utility program, which helps you to perform backup, restoration, and reorganization tasks, and to issue WebSphere MQ commands.

For more information about these diagnostic tools, refer to WebSphere MQ books, which are available through the Web site:

`http://www.ibm.com/software/ts/mqseries/library/`

The most useful for diagnostic information are:

- WebSphere MQ for z/OS Problem Determination, GC34-6054
- WebSphere MQ for z/OS Messages and Codes, SC34-6056
- WebSphere MQ Script (MQSC) Command Reference, SC34-6055
- WebSphere MQ for z/OS System Administration Guide, SC34-6053

Chapter 4. Working with IBM service

When you report a problem to IBM service, you will need to provide as much information as possible to help service personnel quickly resolve the problem. The information you might need to send depends, in part, on the type of problem you have encountered, and includes the following items:

- Job logs for affected address spaces; for example, the controller and any servant regions that the controller terminated
- Job output for affected address spaces, particularly WebSphere Application Server for z/OS messages that are written to the JESMSGLG data set
- The system log (SYSLOG), another source of WebSphere Application Server for z/OS messages
- WebSphere Application Server for z/OS error log
- The system logrec data set or log stream
- CTRACE external writer data sets
- SVC dumps, CEEDUMPs, or other types of dumps produced because of the problem.
- The affected server's environment file, WAS.env, which is located in the HFS:
  `AppServer/config/cells/cellname/nodes/nodename/servers/servername/was.env`

Additionally, IBM service might request you to:

- Provide a description of the circumstances or scenario under which the error occurs.
- Use the `VERBEXIT CBDATA` subcommand. See "Using the IPCS VERBEXIT subcommand to display diagnostic data".
- Reset WebSphere variables that are for use only when directed by IBM service:
  - "Setting trace controls for IBM service" on page 39
  - "Setting dump controls for IBM service" on page 41
- Set WebSphere variable values for the location service daemon address space (same as those for servers, with the prefix "DAEMON_"). Use the procedure in "Steps for configuring WebSphere variables" on page 13.

Using the IPCS VERBEXIT subcommand to display diagnostic data

The interactive problem control system (IPCS) is a tool that provides formatting and analysis support for dumps and traces produced by WebSphere Application Server for z/OS and the applications that it hosts. IBM service personnel might request that you use the IPCS subcommand `VERBEXIT` with the`CBDATA` verb name to display dump information for WebSphere Application Server for z/OS. The CBDATA formatters reside in the `SBBOMIG` data set, which must be in the link list or LPA; see *WebSphere Application Server for z/OS V5.0: Installation and Customization*, GA22-7909 for further details about configuring the `SBBOMIG` data set.

Entering `VERBEXIT CBDATA` results in a display of dump contents that can include the following WebSphere Application Server for z/OS structures:

- Global control blocks
- Address space control blocks
- Task control blocks (TCBs)

- ORB control block

Optional parameters control which of these structures are included in the dump display. If you enter `VERBEXIT CBDATA` without any optional parameters, the dump display includes only global control block contents.

To enter `VERBEXIT CBDATA`, you may use any of the methods for entering IPCS subcommands on z/OS, as described in *z/OS MVS IPCS User's Guide*, SA22-7596. Use the following syntax:

```
VERBEXIT CBDATA [ 'parameter [,parameter]...' ]
```

Valid parameters are:

GLOBAL(bgvt-address)
Formats and displays cell-level global vector data for the specified address space. This display includes the following formatted control blocks:

- BGVT address — z/OS Global Vector table
- ASR Table and ASR Table entries — Active Server Resposity information

ASID(asid-number)
Formats and displays address space information for the daemon, the controller (region), or the servant (region). This display includes the following formatted control blocks:

- BACB — z/OS address space control block
- BTRC,TBUFSET,TBUF — z/OS Component trace control blocks
- BOAM,BOAMX — z/OS BOA control blocks
- ACRW queue — Application Control Region Work element control blocks
- BTCB queues — z/OS control information

Along with `ASID(asid-number)`, IBM service personnel might direct you to specify one of the following parameters, to include additional information in the dump display:

BTCB(btcb_address)
Formats and displays the specified BTCB and ORB information for the WebSphere Application Server for z/OS TCB.

COMMDATA
Formats and displays session information.

CONFIG
Formats and displays configuration information for the address space.

OBJADDR(object_address) and OBJTYPE(object_type_ID)
Formats and displays information for the specified object of the specified type. IBM service personnel will provide the values for you to supply for these parameters.

ORBDATA
Formats and displays ORB information.

TCB(tcb_address)
Formats and displays request summary information for the specified task.

TRACEBACK
Formats and displays ORB information.

SUMMARY
Summarizes information from some of the other CBDATA optional parameters. For example, for the GLOBAL parameter, specifying SUMMARY produces a list of active servers.

Example: Output from the command `ip VERBEXIT CBDATA 'ASID(xx) TCB(yyyyyyyy)'`:

```
command ===> ip VERBX CBDATA 'ASID(xx) TCB(xxxxxxxx)'
******************************** TOP OF DATA ********************************
COMPON=WEBSPHERE Z/OS,COMPID=5655A9801,ISSUER=BBORMCDP,ERRNO=04006006

_____________________________________________________________________________
BBOR0012I Formatting Clssname
 Clssname: 2BE6947E
+0000 D9859496 A385E685 82C39695 A3818995 |RemoteWebContain|.................|
+0010 859900                              |er.             |...              |
BBOR0012I Formatting MethodNm
 MethodNm: 2BE69472
+0000 88A3A397 998598A4 85A2A300 00000000 |httprequest.....|.........?......|
BBOR0012I Formatting ComRtInf
 ComRtInf: 2BE69212
+0000 89974081 8484997E F94BF5F6 4BF4F24B |ip addr=9.56.42.|..@....~.K..K..K|
+0010 F1F6F840 979699A3 7EF1F0F8 F500     |168 port=1085.  |...@....~.....  |
BBOR0026I GMT Time Request was received into CTL region
 TODCLOCK: 00000000
    04/08/2003 12:58:02.926136
BBOR0026I GMT Time Request was Queued to WLM in CTL region
 TODCLOCK: 00000000
    04/08/2003 12:58:02.926263
BBOR0026I GMT Time Request will be Expired (approximated)
 TODCLOCK: 00000000
    04/08/2003 13:08:01.663032
BBOR0026I GMT Time Request was received into SR region
 TODCLOCK: 00000000
    04/08/2003 12:58:02.927729
```

Setting trace controls for IBM service

"Using the z/OS modify command" on page 14

ras_trace_defaultTracingLevel=*n*
Specifies the default tracing level for WebSphere Application Server for z/OS.

Valid values and their meanings are:

0 No tracing

1 Exception tracing

2 Basic and exception tracing

3 Detailed tracing, including basic and exception tracing

Use this variable together with the ras_trace_basic and ras_trace_detail variables to set tracing levels for WebSphere Application Server for z/OS subcomponents.

Default: 1

Example: `ras_trace_defaultTracingLevel=2`

ras_trace_basic=n | (n,...)
Specifies tracing overrides for particular WebSphere Application Server for z/OS subcomponents.

Subcomponents, specified by numbers, receive basic and exception traces. If IBM service directs you to specify more than one subcomponent, use parentheses and separate the numbers with commas. IBM service provides the subcomponent numbers and their meanings.

Other parts of WebSphere Application Server for z/OS receive tracing as specified on the `ras_trace_defaultTracingLevel` variable.

Default: (no default value)

Example: `ras_trace_basic=3`

ras_trace_detail=n | (n,...)
Specifies tracing overrides for particular WebSphere Application Server for z/OS subcomponents.

Subcomponents, specified by numbers, receive detailed traces. If IBM service directs you to specify more than one subcomponent, use parentheses and separate the numbers with commas. IBM service provides the subcomponent numbers and their meanings.

Other parts of WebSphere Application Server for z/OS receive tracing as specified on the `ras_trace_defaultTracingLevel` variable.

Default: (no default value)

Examples:

```
ras_trace_detail=3
ras_trace_detail=(3,4)
```

ras_trace_specific=n | (n,...)
Specifies tracing overrides for specific WebSphere Application Server for z/OS trace points.

Trace points are specified by 8-digit, hexadecimal numbers. If IBM service directs you to specify more than one trace point, use parentheses and separate the numbers with commas. You also can specify a WebSphere variable name by enclosing the name in single quotes. The value of the WebSphere variable will be handled as if you had specified that value on `ras_trace_specific`.

Default: (no default value)

Examples:

```
ras_trace_specific=03004020
ras_trace_specific=(03004020,04005010)
ras_trace_specific='xyz'
        [where xyz is an environment variable name]
ras_trace_specific=('xyz','abc',03004021)
        [where xyz and abc are environment variable names]
```

ras_trace_exclude_specific=n | (n,...)
Specifies WebSphere Application Server for z/OS trace points to exclude from tracing activity.

Trace points are specified by 8-digit, hexadecimal numbers. If IBM service directs you to specify more than one trace point, use parentheses and separate the numbers with commas. You also can specify a WebSphere variable name by enclosing the name in single quotes. The value of the WebSphere variable will be handled as if you had specified that value on `ras_trace_exclude_specific`.

Default: (no default value)

Examples:

```
ras_trace_exclude_specific=03004020
ras_trace_exclude_specific=(03004020,04005010)
ras_trace_exclude_specific='xyz'
        [where xyz is an environment variable name]
ras_trace_exclude_specific=('xyz','abc',03004021)
        [where xyz and abc are environment variable names]
```

Setting dump controls for IBM service

ras_minorcode_action=*value*

Determines the default behavior for gathering documentation about system exception minor codes.

CEEDUMP

Captures callback and offsets.

Tip: It takes time for the system to take CEEDUMPs and this may cause transaction timeouts. For instance, if the WebSphere variable `transaction_defaultTimeout` is set to 30 seconds, your application transaction may time out because processing a CEEDUMP can take longer than 30 seconds. To prevent this from happening, either:

- Increase the transaction timeout value, or
- Code `ras_minorcode_action=NODIAGNOSTICDATA` and make sure the `ras_trace_minorCodeTraceBacks` variable is not specified.

TRACEBACK

Captures Language Environment and z/OS UNIX traceback data.

SVCDUMP

Captures an MVS dump (but will not produce a dump in the client).

NODIAGNOSTICDATA

Specifies that no diagnostic data will be collected, even if CEEDUMP, TRACEBACK, or SVCDUMP processing occurs because of another WebSphere variable setting. For example, if you code both of the following variables, traceback processing occurs but none of the traceback data is collected:

```
ras_minorcode_action=NODIAGNOSTICDATA
ras_trace_minorCodeTraceBacks=ALL
```

Default: NODIAGNOSTICDATA

Example: `ras_minorcode_action=SVCDUMP`

ras_trace_minorCodeTraceBacks=*value*

Enables traceback of system exception minor codes. Values are:

ALL|all

Enables traceback for all system exception minor codes.

minor_code

Enables traceback for a specific minor code.

Example: Type `1234` for minor code `C9C21234`

(null value)

The default. This setting will not cause gathering of a traceback.

Default: (null value)

Example: `ras_trace_minorCodeTraceBacks=all`

Mapping of V4.0.1 environment variables to V5 WebSphere variables

The following table lists only V5 WebSphere variables related to diagnosis, along with their equivalent V4.0.1 environment variables. This information is provided only as an aid to IBM service personnel.

Warning: Do not use this information to manually modify the contents of a `was.env` file. The `was.env` file is managed by WebSphere, and its content is rewritten with each change made to the WebSphere configuration. Therefore, any hand-editing will be overwritten.

Table 10. V4.0.1 environment variables and their equivalent V5 WebSphere variables

V4.0.1 environment variables	Equivalent V5 WebSphere variables
BBOC_HTTP_INPUT_TIMEOUT	**protocol_http_timeout_input**
BBOC_HTTP_OUTPUT_TIMEOUT	**protocol_http_timeout_output**
BBOC_HTTP_PERSISTENT_SESSION_TIMEOUT	**protocol_http_timeout_persistentSession**
BBOC_HTTP_SSL_OUTPUT_TIMEOUT	**protocol_https_timeout_output**
CLIENT_TIMEOUT	**protocol_iiop_local_timeout**
IIOP_SERVER_SESSION_KEEPALIVE	**protocol_iiop_server_session_keepalive**
OTS_DEFAULT_TIMEOUT	**transaction_defaultTimeout**
	control_region_wlm_dispatch_timeout
OTS_MAXIMUM_TIMEOUT	**transaction_maximumTimeout**
RECOVERY_TIMEOUT	**transaction_recoveryTimeout**
SSLIIOP_SERVER_SESSION_KEEPALIVE	**protocol_iiop_server_session_keepalive_ssl**

Appendix A. WebSphere variable definitions

WebSphere Application Server for z/OS provides configuration variables that allow you to control:

- Output destinations and characteristics for the error log, and for CTRACE buffers, data sets and the external writer.
- Trace buffers, data sets, and the content of trace data.
- Types of dumps to be requested.
- Timeout values for system and application behavior.

Setting output destinations and characteristics

client_ras_logstreamname=*name*
Specifies the name of the log stream for an application client run-time to use for error information.

Default: If this variable is not specified, the client run-time uses STDERR.

Example: `client_ras_logstreamname=MY.CLIENT.ERROR.LOG`

Tip: Do not put the log stream name in quotes. Log stream names are not data set names.

ras_default_msg_dd=*DD_card_name*
Redirects write-to-operator (WTO) messages that use the default routing to hardcopy. These messages are redirected to the location identified through the DD card on the server's JCL start procedure. These WTO messages are primarily messages that WebSphere Application Server for z/OS issues during initialization.

Default: No default value is set; WTO messages that use default routing are sent to hardcopy.

Examples:

```
ras_default_msg_dd=MSGDD
ras_default_msg_dd=DFLTLOG
```

Example of the `DFLTLOG DD` card on the server's JCL start procedure:

```
//DFLTLOG    DD   SYSOUT=*
```

ras_hardcopy_msg_dd=*DD_card_name*
Redirects write-to-operator (WTO) messages that WebSphere Application Server for z/OS routes to hardcopy. These messages are redirected to the location identified through the DD card on the server's JCL start procedure. These WTO messages are primarily audit messages issued from Java code during initialization.

Default: No default value is set; WTO messages that use hardcopy routing are sent to hardcopy.

Example: `ras_hardcopy_msg_dd=MSGDD`

ras_log_logstreamName=*name*
Specifies the log stream for WebSphere Application Server for z/OS to use for

error information during bootstrap processing. If the specified log stream is not found or not accessible, a message is issued and errors are written to the server's job log.

Default: If this variable is not specified, WebSphere Application Server for z/OS uses STDERR.

Example: `ras_log_logstreamName=MY.CB.ERROR.LOG`

Tip: Do not put the log stream name in quotes. Log stream names are not data set names.

Setting trace controls

ras_trace_outputLocation=SYSPRINT | BUFFER | TRCFILE
Specifies where you want trace records to be sent:

- To SYSPRINT
- To a memory buffer (BUFFER), the contents of which are later written to a CTRACE data set
- To a trace data set (TRCFILE) specified on the `TRCFILE DD` statement in the server's start procedure.

For servers, you may specify one or more values, separated by a space. For clients, you may specify SYSPRINT only.

Defaults:

- For clients, SYSPRINT
- For all other processes, BUFFER

Example: `ras_trace_outputLocation=SYSPRINT BUFFER`

ras_time_local=0 | 1
Specifies whether timestamps in trace records use Greenwich Mean Time (GMT) or local time. This variable setting controls timestamp format in the error log, and in traces sent to SYSPRINT or TRCFILE DD.

Default: 0 (GMT)

Example: `ras_time_local=1` sets timestamps to local time.

ras_trace_ctraceParms=SUFFIX | MEMBER_NAME
Identifies the CTRACE PARMLIB member. The value can be either:

- A two-character suffix, which is added to the string `CTIBBO` to form the name of the PARMLIB member, or
- The fully specified name of the PARMLIB member. A fully specified name must conform to the naming requirements for a CTRACE PARMLIB member.

If this environment variable is specified and the PARMLIB member is not found, the default PARMLIB member, CTIBBO00, is used. If neither the specified nor the default PARMLIB member is found, tracing is defined to CTRACE, but there is no connection to a CTRACE external writer.

Note: The Daemon is the only server that recognizes this environment variable.

Default: 00 (the default PARMLIB member, CTIBBO00)

Example: `ras_trace_ctraceParms=01` identifies PARMLIB member CTIBBO01

ras_trace_BufferCount=*n*
Specifies the number of trace buffers to allocate. Valid values are 4 through 8.

Default: 4

Example: `ras_trace_BufferCount=6`

ras_trace_BufferSize=*n*
Specifies the size of a single trace buffer in bytes. You can use the letters K (for kilobytes) or M (for megabytes). Valid values are 128K through 4M.

Default: 1M

Example: `ras_trace_BufferSize=2M`

Setting dump controls

ras_dumpoptions_dumptype=*n*
Specifies the default dump used by the signal handler. Valid values and their meanings are:

0 No dump is generated.

1 A ctrace dump is taken.

2 A cdump dump is taken.

3 A csnap dump is taken.

4 A CEE3DMP dump is taken.

CEE3DMP generates a dump of Language Environment and the member language libraries. Sections of the dump are selectively included, depending on dump options specified, either by default or through the `ras_dumpoptions_ledumpoptons` variable. By default, this value passes `THREAD(ALL) BLOCKS` to CEE3DMP. You can override the default options for CEE3DMP through the `ras_dumpoptions_ledumpoptons` variable. For more information about CEE3DMP and its options, see *z/OS Language Environment Programming Reference*, SA22-7562.

Default: 3

Example: `ras_dumpoptions_dumptype=2`

ras_dumpoptions_ledumpoptons=*options*
Specifies dump options to be used with a CEE3DMP. If you want more than one option, separate each option with a blank.

This WebSphere variable is used only when you specify `ras_dumpoptions_dumptype=4`. For an explanation of CEE3DMP and valid dump options, see *z/OS Language Environment Programming Reference*, SA22-7562.

Rule: The maximum length of the option string on this environment variable is 255. If the option string is longer than 255, you receive message BBOM0011W and the CEE3DMP dump options are set to `THREAD(ALL) BLOCKS`.

Default: `THREAD(ALL) BLOCKS`

Example: `ras_dumpoptions_ledumpoptons=NOTRACEBACK NOFILES`

Controlling behavior through timeout values

control_region_wlm_dispatch_timeout
Specifies the maximum amount of time, in seconds, that WebSphere Application Server for z/OS will wait for IIOP requests to complete. This time limit includes:

- Time during which the IIOP request waits on the WLM queue until being dispatched to a servant (region), and
- Time during which an application component, running in the servant, processes the request and generates a response.

The server generates a failure response if this processing does not complete within the specified time.

Note: This variable setting does not apply for HTTP requests; for that type of work, the value specified through the `protocol_http_timeout_output` variable controls the time allowed for dispatching work to a servant (region).

Default: 300 seconds

Example: `control_region_wlm_dispatch_timeout=600`

protocol_http_timeout_input
Sets a maximum amount of time, in seconds, that the Java server will wait for the complete HTTP request to arrive. The Java server starts the timer after the connection has been established, and cancels the connection if a complete request does not arrive within the specified maximum time limit. Specifying a value of zero disables the time-out function.

Default: 10 seconds

Example: `protocol_http_timeout_input=15`

protocol_http_timeout_output
Sets a maximum amount of time, in seconds, that the Java server will wait for an application component to respond to an HTTP request. If the response is not received within the specified length of time, the servant (region) fails with ABENDEC3 and RC=04130007. Setting this timer prevents client applications from waiting for a response from an application component that might be deadlocked, looping, or encountering other processing problems related to:

- JSP compiles (javac / jit compiles)
- XML parser
- jaspr

Default: 120 seconds

Example: `protocol_http_timeout_output=150`

protocol_http_timeout_persistentSession
Specifies the time, in seconds, that the Java server will wait for a subsequent request from an HTTP client on a persistent connection. If another request is not received from the same client within this time limit, the connection is closed.

Default: 30 seconds

Example: `protocol_http_timeout_persistentSession=40`

protocol_https_timeout_input
Specifies the maximum amount of time, in seconds, that the Java server will allow for the complete HTTPS request to be received before cancelling the connection.

Default: 10 seconds

Example: `protocol_https_timeout_input=15`

protocol_https_timeout_output
Specifies the maximum amount of time, in seconds, that the Java server will wait from the time the complete HTTPS request is received until output is available to be sent to the client.

Default: 120 seconds

Example: `protocol_https_timeout_output =150`

protocol_https_timeout_persistentSession
Specifies the time, in seconds, that the Java server will wait between requests issued over a persistent connection from an HTTPS client. After the server sends a response, it uses the persistent timeout to determine how long it should wait for a subsequent request before cancelling the persistent connection.

Default: 30 seconds

Example: `protocol_https_timeout_persistentSession=40`

protocol_iiop_local_timeout
Specifies the maximum time, in tenths of seconds, that the client will wait for the response to a client request. This variable is the only time-out available for remote method dispatches made by clients only, not by application components within the servant region. Because the sysplex TCP/IP that runs through the coupling facility does not always tell the client when the other end of the socket is gone, clients can wait indefinitely for a response unless you set this variable. Setting `protocol_iiop_local_timeout` ensures that the client gets a response within the specified time, even if the response is a `COMM_FAILURE` exception.

Default: 0 (unlimited), which means no time-out value is set

Example: `protocol_iiop_local_timeout=20` sets the time to 2 seconds

protocol_iiop_server_session_keepalive
Defines the value, in seconds, provided to TCP/IP on the SOCK_TCP_KEEPALIVE option for the IIOP listener. The function of this option is to verify if idle sessions are still valid by polling the client TCP/IP stack. If the client does not respond, the session is closed. If the client goes away without notifying the server, it would unnecessarily leave the session active on the server side. Use this option to clean up these unnecessary sessions.

Notes:

1. If the environment variable is not set, the TCP/IP option is not set.
2. Setting the SOCK_TCP_KEEPALIVE option generates network traffic on idle sessions, which can be undesirable.

Default: 0

Example: `protocol_iiop_server_session_keepalive=3600`

protocol_iiop_server_session_keepalive_ssl

transaction_defaultTimeout
Specifies the maximum amount of time, in seconds, that the Java server will wait for an application transaction to complete. This default amount of time is given to the application transaction if it does not set its own time-out value through the `current→set_timeout` method. The timer in the controller (region) starts when the container starts a global transaction. If the application transaction is not committed or rolled back within the specified time, the controller abends servant (region) in which the application component is running, with abend EC3 reason code 04130002 or 04130005.

The maximum value is 2147483 seconds (24.85 days). You should not use a null or 0 value.

Default: 120 seconds

Example: `transaction_defaultTimeout=300`

transaction_maximumTimeout
Specifies the maximum amount of time, in seconds, that your installation will allow an application to specify for its transactions to complete. If an application assigns a greater amount of time through the `current→set_timeout` method, the Java server overrides the application setting to the value specified for the `transaction_maximumTimeout` variable. If the application does not set its own time-out value through the `current→set_timeout` method, the default value set through the `transaction_defaultTimeout` variable applies.

The maximum value is 2147483 seconds (24.85 days). You should not use a null or 0 value.

Default: 300 seconds

Example: `transaction_maximumTimeout=600`

transaction_recoveryTimeout
Specifies the time, in minutes, that this controller (region) uses to attempt to resolve in-doubt transactions before issuing a write-to-operator-with-reply (WTOR) message to the console, requesting whether it should:

- Stop trying to resolve in-doubt transactions,
- Write transaction-related information to the job log or hard copy log, and
- Terminate.

If the operator replies that recovery should continue, the controller (region) will attempt recovery for the specified amount of time before re-issuing the WTOR message. Once all the transactions are resolved, the control region terminates. This variable applies only to controllers in peer restart and recovery mode.

Default: 15 minutes

Example: `transaction_recoveryTimeout=7`

Appendix B. The error dump and cleanup interface

Note: This appendix contains Programming Interface and Associated Guidance Information.

The Error Dump and Cleanup (BBORLEXT) interface exists to call WebSphere Application Server for z/OS in a recovery environment to allow it to request a dump and to clean up its resources.

The interface will:

- Save the function and DLL names of the failing z/OS component into the SDWA.
- Determine whether or not to issue an SDUMP, if relevant to the time-of-failure environment.
- Clean up z/OS internal structures and connections.

Program requirements

This interface **must** be called from within a WebSphere Application Server for z/OS location service daemon, controller (region), or servant (region). There are no restrictions against in which recovery environment, such as an ESTAE or FRR routine, the caller must reside.

General information

Interface:	BALR to BBORLEXT
Address of routine:	(ECVT+'234'x)+'20'x
Address mode:	AMODE 31, RMODE any
State:	Allow problem program state, task mode
Cross memory mode:	PASN=HASN=SASN (non-cross memory)
Return codes:	No return codes
Function:	Clean-up various WebSphere for z/OS resources and possibly issue an SVC dump for the current address space

Input register information

The contents of the registers are as follows:

1	Contains the address of the SDWA
14	Contains the return address
15	Contains the entry point address of BBORLEXT

Output register information

When control returns to the caller, the contents of the registers are as follows:

0-1	Used as a work register by the system
2-14	Unchanged
15	Used as a work register by the system

Note: Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on

which the caller depends, the caller must save them before issuing the service and restore them after the system returns control.

Note: A dump will not occur for X22 abends or for certain reason codes from 0D6, 052, 067, CC3, and DC3 abends. There may also be other error conditions that will not create a dump.

Example:

```
Example
Here is an example of how to call this routine in assembler:

LA 1,SDWA       Load SDWA@ in Reg 1
L 15,(0,16)     Load CVT address
L 15,140(,15)   Load ECVT address
L 15,564(,15)   Load address of z/OS structure
L 15,32(,15)    Load address of z/OS routine
BALR 14,15      Invoke z/OS routine
```

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.

Trademarks

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

DB2
IBM
Language Environment
MVS
OS/390
RACF
S/390
WebSphere
z/OS

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.

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

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

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

Programming interface information

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

Glossary

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

- *IBM Glossary of Computing Terms*, located on the Internet at:

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

  `http://java.sun.com/docs/glossary.html`
- The Sun Web site, located on the Internet at:

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

Program Number: 5655–I35

Printed in the United States of America

GA22-7914-00