WebSphere Application Server for z/OS V5.0.2:

Troubleshooting

GA22-7964-01

Note

Before using this information, be sure to read the general information under “Notices” on page 181.

Compilation date: December 2, 2003

Contents

How to send your comments . . . vii

Chapter 1. Diagnosing and fixing problems . . . 1
Acquiring skills for problem determination . . . 1
Working with diagnostic tools and controls . . . 1

Chapter 2. Preparing for diagnosis . . . 3
Best practices for using system controls . . . 3
Updating the CFRM policy . . . 4
Configuring WebSphere Application Server for z/OS variables . . . 5
  Steps for configuring WebSphere variables . . . 5
Setting up component trace (CTRACE) . . . 6
  Steps for preparing CTRACE controls and resources . . . 6
  Steps for starting CTRACE as part of WebSphere Application Server for z/OS customization . . . 8
  Steps for starting CTRACE while WebSphere Application Server for z/OS servers are active . . . 8
  Using CTRACE to collect trace data for Java server applications . . . 9
Setting up the error log . . . 9
Using the z/OS modify command. . . . 10
Types of configuration variables . . . 11
  Setting output destinations and characteristics . . . 11
  Setting trace controls . . . 12
  Setting dump controls . . . 13
  Controlling behavior through timeout values . . . 14
Debugging WebSphere Application Server applications . . . 16
  Attaching WebSphere Studio Application Developer to a remote debug session . . . 18
  Unit testing with DB2 . . . 18
Adding logging and tracing to your application . . . 18
  Programming with the JRas framework . . . 19
  Logging messages and trace data for Java server applications . . . 36
Collecting job-related information with Systems Management Facility (SMF) . . . 37
  Setting up SMF . . . 38
  SMF record types . . . 39
  Enabling SMF recording . . . 41
  Formatting the output data set . . . 43
  Disabling SMF recording for WebSphere Application Server . . . 43
  Disabling SMF recording for the entire MVS system . . . 43
  Overview of SMF record type 120 . . . 43
  Overview of SMF record type 80 . . . 60
  Collecting performance diagnosis information . . . 61

Chapter 3. Viewing diagnostic information . . . 63
Viewing CEEDUMPs in the job log . . . 63
Viewing SVC dumps . . . 63
Viewing CTRACE and JRas data through IPCS . . . 64
  Steps for using the IPCS dialog to format CTRACE data . . . 64
  Finding the subname for IPCS CTRACE . . . 65
  Steps for using IPCS in batch mode to format CTRACE data . . . 65
Viewing error log contents through the Log Browse Utility (BBORBLOG) . . . 68
  Using the log browse utility (BBORBLOG) . . . 68
  Error log stream record output. . . . 69
Using the z/OS display command . . . 71
Converting Java minor codes . . . 71
Viewing SYSPRINT output, STDOUT and STDERR streams in an HFS File . . . 71

Message routing 72
Using the Error Dump and Cleanup interface 73
Viewing the output data set 74
Example of SMF Browser output 75

Chapter 4. Automation and recovery scenarios and guidelines 77
APPC automation and recovery scenarios 77
WLM automation and recovery scenarios. 78
RACF automation and recovery scenarios 78
RRS automation and recovery scenarios 79
UNIX System Services automation and recovery scenarios 80
TCP/IP automation and recovery scenarios 80
DB2 automation and recovery scenarios 80
CICS automation and recovery scenarios. 81
IMS automation and recovery scenarios 82
LDAP automation and recovery scenarios 83
WebSphere Application Server for z/OS (Daemon) automation and recovery scenarios 84
Web server (servlet) automation and recovery scenarios 84

Chapter 5. Preparing for a call to IBM service 85
Using the IPCS VERBEXIT subcommand to display diagnostic data 85
Setting trace controls for IBM service 87
Setting dump controls for IBM service 90

Chapter 6. Diagnosing and fixing problems: Resources for learning 93

Chapter 7. Troubleshooting by task: what are you trying to do? 95
Troubleshooting installation problems 95
Installation completes but the administrative console does not start 95
Installation completes, but sample applications do not work 96
Troubleshooting testing and first time run problems 96
Web resource (JSP file, servlet, HTML file, image) does not display 97
Cannot access a data source 102
Cannot access an enterprise bean from a servlet, JSP file, stand-alone program, or other client 114
Cannot look up an object hosted by WebSphere Application Server from a servlet, JSP file, or other client. 117
Errors or access problems after enabling security 120
Errors after enabling Secure Sockets Layer, or Secure Sockets Layer-related error messages 124
Errors in messaging (JMS API) 126
Errors returned to client trying to send a SOAP request 126
Errors connecting to WebSphere MQ and creating WebSphere MQ queue connection factory 127
Troubleshooting application runtime and management problems 128
Problems starting or using the wsadmin command 128
Web module or application server dies or hangs 130
Errors when trying to configure or enable security 131
Workload not getting distributed. 133
Errors setting up multiserver environments. 134
Cannot uninstall an application or remove a node or application server 136
Problems creating or using HTTP sessions 137
JSP source code shown by the Web server 139
Problems using tracing, logging or other troubleshooting features 140
Errors connecting to the administrative console from a Netscape browser 140
Resolving timeout conditions 142
Understanding how timers work. 142
Guidelines for analyzing diagnostic data for timeout conditions 143
Identifying possible causes of and fixes for timeout conditions 145

Guidelines for altering timeout values. 146
Debugging client exceptions 148
Debugging applications that hang 148
Debugging problems related to Java Message Service (JMS) support. 150

Chapter 8. Troubleshooting by component: what is not working?. 151
Installation component troubleshooting tips 151
Administration and administrative console troubleshooting tips 151
Application Assembly Tool troubleshooting tips 152
Web Container troubleshooting tips 153
JDBC and data source troubleshooting tips 153
HTTP plug-in component troubleshooting tips. 153
HTTP session manager troubleshooting tips 155
Naming services component troubleshooting tips 156
Messaging (JMS) component troubleshooting tips 157
Universal Discovery, Description, and Integration, Web Service, and SOAP component troubleshooting tips 157
Enterprise bean and EJB container troubleshooting tips 157
Cannot restart the Deployment Manager monitoring policy 158
Security components troubleshooting tips 158
JSP engine troubleshooting tips. 175
Object request broker component troubleshooting tips 176
Web services security troubleshooting tips 177

Notices 181
Trademarks and service marks 181

How to send your comments

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

- To send comments on articles in the WebSphere Application Server Information Center
  1. Display the article in your Web browser and scroll to the end of the article.
  2. Click on the **Feedback** link at the bottom of the article, and a separate window containing an e-mail form appears.
  3. Fill out the e-mail form as instructed, and click on **Submit feedback** .
- 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. Diagnosing and fixing problems

The purpose of this section is to aid you in understanding why your enterprise application, application server, or WebSphere Application Server is not working and to help you resolve the problem. Unlike performance tuning which focuses on solving problems associated with slow processes and un-optimized performance, problem determination focuses on finding solutions to functional problems.

The kind of problem you are encountering, and how much you already know about it, determine what steps to take to resolve it:

1. For tips on investigating common problems organized according to tasks within WebSphere Application server, see Troubleshooting by task.
2. For tips on how to investigate common kinds of problems based on the component that is causing the problem, see Troubleshooting by component.
3. For help in using WebSphere Application Server utilities to help you diagnose the problem, see Working with diagnostic tools and controls.
4. For help in viewing diagnostic information like dumps, error logs and CTRACE information, see Viewing diagnostic information
5. For current information available from IBM Support on known problems and their resolution, see the IBM Support page.
6. IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

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.

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 roles defined in the Sun Microsystems Java 2 Enterprise Edition Specification V1.3.
- The programming models and specifications for application components (Enterprise beans, Web applications, and client programs) supported in the J2EE 1.3 environment.
- 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

The following list summarizes the z/OS tools you will use to access and work with diagnostic information.

Note: You must update the CFRM policy before using log streams that are CF-resident, such as the WebSphere error log and RRS logs. See Updating the CFRM policy for details.

- **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 stderr will end up in SYSOUT. In addition, SYSOUT might contain error messages that usually appear in the log stream, but were redirected to SYSOUT because 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 z/OS Web Library to access the following books:
- *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.

Chapter 2. Preparing for diagnosis

Best practices 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.
  - The Support Web page of the WebSphere Application Server for z/OS Web site, which contains a table of the latest authorized program analysis reports (APARs).

  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.

Best practices 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.
- 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.

Updating the CFRM policy

You must update the CFRM policy before using log streams that are CF-resident, such as the WebSphere error log and RRS logs. If you have the source for the current active CFRM policy, update the source and use the IXCMIAPU Administrative Data utility to generate the new policy. If you do not have the source for the current active CFRM policy, rebuild the source from the active CFRM policy:

1. Find the active policy by issuing the command: D XCF,POL You will get output similar to this (partial display):

```
D XCF,POL
  IXC364I  10.57.49  DISPLAY XCF 061
  . . .
  TYPE: CFRM
       POLNAME:       POLCF1N1
       STARTED:       03/14/2003 11:32:22
       LAST UPDATED:  03/14/2003 11:31:52
  . . .
```

2. List the active CFRM Policy's structure definitions by using the Administrative Data utility:

```
//STEP1   EXEC PGM=IXCMIAPU
//SYSPRINT DD   SYSOUT=*
//SYSABEND DD   SYSOUT=*
//SYSIN    DD   *
DATA TYPE(CFRM) REPORT(YES)
/*
```

3. Extract the definitions for the ACTIVE policy only. Using the SYSPRINT from the above utility job, edit the output with the following steps so it can be used to define a new policy in the next job:
   - Extract the definitions for the ACTIVE policy only.
   - Delete the heading lines.
   - Add the new structure definition using the BBOWCFRM member of the target CNTL dataset as a model.
   - Copy it into a FB-LRECL(80) dataset to be used as SYSIN for the following job.
4. Use the Administrative Data utility to update the CFRM policy. The policy name can be the same as the ACTIVE CFRM policy or a new name. If you use the active policy name, REPLACE(YES) must be specified on the DEFINE control statement.

```
//STEP20   EXEC PGM=IXCMIAPU
//SYSPRINT DD   SYSOUT=*
//SYSABEND DD   SYSOUT=*
//SYSIN    DD   *
 DATA TYPE(CFRM) REPORT(YES)

 DEFINE POLICY NAME(POLCF1N1) REPLACE(YES)

    CF NAME(CF1LPAR) DUMPSPACE(5000) PARTITION(0E) CPCID(00)
       TYPE(009672) MFG(IBM) PLANT(02) SEQUENCE(000000051205)

    CF NAME(CF2LPAR) DUMPSPACE(5000) PARTITION(0F) CPCID(00)
       TYPE(009672) MFG(IBM) PLANT(02) SEQUENCE(000000051205)

    STRUCTURE NAME(CTS130_DFHLOG) SIZE(24000) INITSIZE(12000)
```

```
          REBUILDPERCENT(1)  PREFLIST(CF1LPAR, CF2LPAR)
      . . .
                   <== Insert your new structure definitions here
```

5. Activate the new policy by issuing the following MVS Command:

   ```
   SETXCF START,POLICY,TYPE=CFRM,POLNAME=POLCF1N1
   ```

Configuring WebSphere Application Server for z/OS 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.

Steps for configuring WebSphere variables

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.

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, commonly known as console settings.

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. To have the configuration take effect, stop the server and then start the server again.

Setting up component trace (CTRACE)

This file describes how to set up 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 information about the advantages of using CTRACE facilities, see *z/OS MVS Diagnosis: Tools and Service Aids, GA22-7589*

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.

   Note: Tips:
   - See Setting Trace Controls 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.

- 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.

2. Through modify commands, you have the ability to dynamically and temporarily direct trace data to SYSPRINT or TRCFILE.
   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.
3. 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
4. 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 . Place the start procedure in your system proclib.
5. 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.

Table 1. Parameters for the CTIBBOxx parmlib member (continued)

`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.

6. Ensure that the DLL named `BBORTSS5` is in the link-pack area (LPA).

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"

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

Make sure you have properly prepared CTRACE controls and resources as described in "Steps for preparing CTRACE controls and resources" on page 6

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 and WebSphere variables, if you want values other than the defaults. The ISPF customization dialog generates instructions for starting the CTRACE writer and 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.

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

Make sure you have properly prepared CTRACE controls and resources as described in "Steps for preparing CTRACE controls and resources" on page 6

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.

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.
   - 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) and *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.

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

Using CTRACE to collect trace data for Java server applications

This file describes how to set up CTRACE

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.

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 log stream 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 about z/OS log stream requirements, access the *z/OS MVS Setting up a Sysplex, SA22-7625* available on the z/OS Library Web page

Using the z/OS modify command

This file lists parameters that allow you to dynamically change values related to tracing activity for a server or servant.

You may use either the WebSphere Application Server 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.

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 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	For more information, see..
TRACEALL	ras_trace_defaultTracingLevel	Internal tracing tips for WebSphere for z/OS
TRACEBASIC	ras_trace_basic Note: Do not change this variable unless directed by IBM service personnel.	Setting trace controls for IBM service
TRACEDETAIL	ras_trace_detail	Setting trace controls for IBM service
TRACESPECIFIC	ras_trace_specific Note: Do not change this variable unless directed by IBM service personnel.	Setting trace controls for IBM service
TRACE_EXCLUDE_SPECIFIC	ras_trace_exclude_specific Note: Do not change this variable unless directed by IBM service personnel.	Setting trace controls for IBM service
TRACEINIT	n/a	Example: Getting help for the modify command
TRACENONE	n/a	Example: Getting help for the modify command

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

TRACETOSYSPRINT	ras_trace_outputLocation=SYSPRINT	Internal tracing tips for WebSphere for z/OS
TRACETOTRCFILE	ras_trace_outputLocation=TRCFILE	Internal tracing tips for WebSphere for z/OS
TRACEJAVA	n/a	Example: Getting help for the modify command

Types of configuration variables

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

Controlling behavior through timeout values

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. 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

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 J2EE server will wait for the complete HTTP request to arrive. The J2EE 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 J2EE 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 J2EE 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 J2EE 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 J2EE 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 J2EE 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.

- If the environment variable is not set, the TCP/IP option is not set.
- 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 J2EE 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 J2EE 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`

Debugging WebSphere Application Server applications

In order to debug your application, you must use your application development tool (such as WebSphere Studio Application Developer) to create a Java project or a project with a Java nature. You must then import the program that you want to debug into the project. By following the steps below, you can import the WebSphere Application Server examples into a Java project.

There are two debugging styles available:

- **Step-by-step** debugging mode prompts you whenever the server calls a method on a Web object. A dialog lets you step into the method or skip it. In the dialog, you can turn off step-by-step mode when you are finished using it.

- **Breakpoints** debugging mode lets you debug specific parts of programs. Add breakpoints to the part of the code that you must debug and run the program until one of the breakpoints is encountered.

Breakpoints actually work with both styles of debugging. Step-by-step mode just lets you see which Web objects are being called without having to set up breakpoints ahead of time.

You need not import an entire program into your project. However, if you do not import all of your program into the project, some of the source might not compile. You can still debug the project. Most features of the debugger work, including breakpoints, stepping, and viewing and modifying variables. You must import any source that you want to set breakpoints in.

The inspect and display features in the source view do not work if the source has build errors. These features let you select an expression in the source view and evaluate it.

1. Create a Java Project by opening the New Project dialog.
2. Select **Java** from the left side of the dialog and **Java Project** in the right side of the dialog.
3. Click **Next** and then specify a name for the project (such as WASExamples).
4. Press **Finish** to create the project.
5. Select the new project, choose **File > Import > File System**, then **Next** to open the import file system dialog.
6. Select the directory Browse pushbutton and go to the following directory: *installation_root*\installedApps*node_name* \DefaultApplication.ear\DefaultWebApplication.war.
7. Select the checkbox next to DefaultWebApplication.war in the left side of the Import dialog and then click **Finish**. This will import the JavaServer Pages files and Java source for the examples into your project.
8. Add any JAR files needed to build to the Java Build Path. To do this, select **Properties** from the right-click menu. Choose the Java Build Path node and then select the Libraries tab. Use the Add External JARs pushbutton to add the following JAR files:
   - *installation_root*\installedApps*node_name*\DefaultApplication.ear\Increment.jar.

     Once you have added this JAR file, select it and use the **Attach Source** pushbutton to attach Increment.jar as the source - Increment.jar contains both the source and class files.
   - *installation_root*\lib\j2ee.jar
   - *installation_root*\lib\pagelist.jar
   - *installation_root*\\lib\webcontainer.jar

   Click **OK** when you have added all of the JARs.
9. You can set some breakpoints in the source at this time if you like, however, it is not necessary as step-by-step mode will prompt you whenever the server calls a method on a Web object. Step-by-step mode is explained in more detail below.
10. To start debugging, you need to start the WebSphere Application Server in debug mode and make note of the JVM debug port. The default value of the JVM debug port is 7777.
11. Once the server is started, switch to the debug perspective by selecting **Window > Open Perspective > Debug**. You can also enable the debug launch in the Java Perspective by choosing **Window > Customize Perspective** and selecting the **Debug** and **Launch** checkboxes in the **Other** category.
12. Select the workbench toolbar **Debug** pushbutton and then select **WebSphere Application Server Debug** from the list of launch configurations. Click the **New** pushbutton to create a new configuration.
13. Give your configuration a name and select the project to debug (your new WASExamples project). Change the port number if you did not start the server on the default port (7777).
14. Click **Debug** to start debugging.
15. Load one of the examples in your browser (for example, http://localhost:9080/hitcount).

Attaching WebSphere Studio Application Developer to a remote debug session

The steps below describe how to attach WebSphere Studio Application Developer to a remote debug session on WebSphere Application Server for z/OS. Remote debugging can prove useful when the program you are debugging behaves differently on a z/OS system than on your local system.

1. Enable the debug engine on WebSphere Application Server for z/OS using the adminstrative console. See Debugging WebSphere Application Server Applications.
2. Import the the java source code you wish to debug into WebSphere Studio Application Developer and set breakpoints. See the *WebSphere Studio Application Developer InfoCenter* at `http://publib.boulder.ibm.com/infocenter/wsphelp/` for instructions on setting breakpoints.
3. Open a WebSphere Studio Application Developer debug perspective and create a debug session configuration. See the *WebSphere Studio Application Developer InfoCenter* at `http://publib.boulder.ibm.com/infocenter/wsphelp/` for instructions.
4. Attach WebSphere Studio Application Developer to the WebSphere Application Server for z/OS debug runtime. See ″Connecting to a remote VM with the remote Java application launch configuration″ in the *WebSphere Studio Application Developer InfoCenter* at `http://publib.boulder.ibm.com/infocenter/wsphelp/`
5. Execute the java code in WebSphere Application Server for z/OS to hit the breakpoints set in WebSphere Studio Application Developer.
6. Use WebSphere Studio Application Developer debugger controls and features to debug the application. See the *WebSphere Studio Application Developer InfoCenter* at `http://publib.boulder.ibm.com/infocenter/wsphelp/` for more information.

Unit testing with DB2

The steps below describe how to setup a unit test environment that would allow you to develop and unit test code with DB2 z/OS to support Container Managed Persistence (CMP) development and access DB2 test data that resides on z/OS.

1. Configure DB2 Distributed Data Facility on z/OS to allow remote TCP/IP connections from your WebSphere Studio Application Developer workstation.
2. Install the DB2 Client Configuration Assistant on the workstation where WebSphere Studio Application Developer is installed. The DB2 Client Configuration Assistant is shipped with DB2.
3. Use the DB2 Client Configuration Assistant to define a DB2 alias.
4. Use the DB2 alias you defined to access the DB2 subsystem on z/OS using the DB2 Distributed Data Facility (DDF).

Adding logging and tracing to your application

Designers and developers of applications that run with or under WebSphere Application Server, such as servlets, JSP files, enterprise beans, client applications, and their supporting classes, may find it useful to use the same facility for generating messages that WebSphere Application Server itself uses, JRas.

This approach has advantages over simply adding `System.out.println()` statements to your code:

- Your messages appear in the WebSphere Application Server standard message format with additional data, such as a date and time stamp, added automatically.
- You can more easily correlate problems and events in your own application to problems and events associated with WebSphere Application Server components.
- You can take advantage of the WebSphere Application Server log file management features.
- You can view your messages with the WebSphere Application Server user-friendly Log Analyzer tool.

Unless you choose to extend the JRas framework, using the JRas API set is only slightly more complicated than writing basic `System println()` statements.

Programming with the JRas framework

Use the JRas extensions to incorporate message logging and diagnostic trace into WebSphere Application Server applications. The JRas extensions are based on the stand-alone JRas logging toolkit.

1. Retrieve a reference to the JRas manager.
2. Retrieve message and trace loggers by using methods on the returned manager.
3. Call the appropriate methods on the returned message and trace loggers to create message and trace entries, as appropriate.

Understanding the JRas facility

Developing, deploying and maintaining applications are complex tasks. For example, when a running application encounters an unexpected condition it might not be able to complete a requested operation. In such a case you might want the application to inform the administrator that the operation has failed and give information as to why. This enables the administrator to take the proper corrective action. Application developers or maintainers might need to gather detailed information relating to the execution path of a running application in order to determine the root cause of a failure that is due to a code bug. The facilities that are used for these purposes are typically referred to as message logging and diagnostic trace.

Message logging (messages) and diagnostic trace (trace) are conceptually quite similar, but do have important differences. It is important for application developers to understand these differences in order to use these tools properly. To start with, the following operational definitions of messages and trace are provided.

Message
A message entry is an informational record intended to be viewed by end users, systems administrators and support personnel. The text of the message must be clear, concise and interpretable by an end user. Messages are typically localized, meaning they are displayed in the national language of the end user. Although the destination and lifetime of messages might be configurable, some level of message logging is always enabled in normal system operation. Message logging must be used judiciously due to both performance considerations and the size of the message repository.

Trace A trace entry is an information record that is intended to be used by service engineers or developers. As such a trace record may be considerably more complex, verbose and detailed than a message entry. Localization support is typically not used for trace entries. Trace entries may be fairly inscrutable, understandable only by the appropriate developer or service personnel. It is assumed that trace entries are not written during normal runtime operation, but may be enabled as needed to gather diagnostic information.

WebSphere Application Server provides a message logging and diagnostic trace API that can be used by applications. This API is based on the stand-alone JRas logging toolkit which was developed by IBM. The stand-alone JRas logging toolkit is a collection of interfaces and classes that provide message logging and diagnostic trace primitives. These primitives are not tied to any particular product or platform. The stand-alone JRas logging toolkit provides a limited amount of support (typically referred to as systems management support), including log file configuration support based on property files.

As designed, the stand-alone JRas logging toolkit does not contain the support required for integration into the WebSphere Application Server runtime or for usage in a J2EE environment. To overcome these limitations, WebSphere Application Server provides a set of extension classes to address these shortcomings. This collection of extension classes is referred to as the JRas extensions. The JRas extensions do not modify the interfaces introduced by the stand-alone JRas logging toolkit, but simply provide the appropriate implementation classes. The conceptual structure introduced by the stand-alone JRas logging toolkit is described below. It is equally applicable to the JRas extensions.

JRas Concepts

The following is a basic overview of important concepts and constructs introduced by the stand-alone JRas logging toolkit. It is not meant to be an exhaustive overview of the capabilities of this logging toolkit, nor is it intended to be a detailed discussion of usage or programming paradigms. More detailed information, including code examples, is available in JRas extensions and its subtopics, including in the Javadoc for the various interfaces and classes that make up the logging toolkit.

Event Types
: The stand-alone JRas logging toolkit defines a set of event types for messages and a set of event types for trace. Examples of message types include informational, warning and error. Examples of trace types include entry, exit and trace.

Event Classes
: The stand-alone JRas logging toolkit defines both message and trace event classes.

Loggers
: A logger is the primary object with which the user code interacts. Two types of loggers are defined. These are message loggers and trace loggers. The set of methods on message loggers and trace loggers are different, since they provide different functionality. Message loggers create only message records and trace loggers create only trace records. Both types of loggers contain masks that indicates which categories of events the logger should process and which it should ignore. Although every JRas logger is defined to contain both a message and trace mask, the message logger only uses the message mask and the trace logger only uses the trace mask. For example, by setting a message logger's message mask to the appropriate state, it can be configured to process only Error messages and ignore Informational and Warning messages. Changing the state of a message logger's trace mask has no effect.

A logger contains one or more handlers to which it forwards events for further processing. When the user calls a method on the logger, the logger will compare the event type specified by the caller to its current mask value. If the specified type passes the mask check, the logger will create an event object to capture the information relating to the event that was passed to the logger method. This information may include information such as the names of the class and method which is logging the event, a message and parameters to log, among others. Once the logger has created the event object, it forwards the event to all handlers currently registered with the logger.

Methods that are used within the logging infrastructure itself should not make calls to the logger method. When an application uses an object that extends a thread class, implements the hashCode(), and makes a call to the logging infrastructure from that method, the result is a recursive loop.

Handlers
: A handler provides an abstraction over an output device or event consumer. An example is a file handler, which knows how to write an event to a file. The handler also contains a mask that is used to further restrict the categories of events the handler will process. For example, a message logger may be configured to pass both warning and error events, but a handler attached to the message logger may be configured to only pass error events. Handlers also include formatters, which the handler invokes to format the data in the passed event before it is written to the output device.

Formatters
: Handlers are configured with Formatters, which know how to format events of certain types. A handler may contain multiple formatters, each of which know how to format a specific class of event. The event object is passed to the appropriate formatter by the handler. The formatter returns formatted output to the handler, which then writes it to the output device.

JRas Extensions

JRas extensions

The stand-alone JRas logging toolkit defines interfaces and provides a variety of concrete classes that implement these interfaces. Since the stand-alone JRas logging toolkit was developed as a general purpose toolkit, the implementation classes do not contain the configuration interfaces and methods necessary for use in the WebSphere Application Server product. In addition, many of the implementation classes are not written appropriately for use in a J2EE environment. To overcome these shortcomings, WebSphere Application Server provides the appropriate implementation classes that allow integration into the WebSphere Application Server environment. The collection of these implementation classes is referred to as the JRas extensions.

Usage Model

You can use the JRas extensions in three distinct operational modes:

Integrated
In this mode, message and trace records are written only to logs defined and maintained by the WebSphere Application Server runtime. This is the default mode of operation and is equivalent to the WebSphere Application Server 4.0 mode of operation.

stand-alone
In this mode, message and trace records are written solely to stand-alone logs defined and maintained by the user. You control which categories of events are written to which logs, and the format in which entries are written. You are responsible for configuration and maintenance of the logs. Message and trace entries are not written to WebSphere Application Server runtime logs.

Combined
In this mode message and trace records are written to both WebSphere Application Server runtime logs and to stand-alone logs that you must define, control, and maintain. You can use filtering controls to determine which categories of messages and trace are written to which logs.

The JRas extensions are specifically targeted to an integrated mode of operation. The integrated mode of operation can be appropriate for some usage scenarios, but there many scenarios are not adequately addressed by these extensions. Many usage scenarios require a stand-alone or combined mode of operation instead. A set of user extension points has been defined that allow the JRas extensions to be used in either a stand-alone or combined mode of operations.

JRas extension classes:

WebSphere Application Server provides a base set of implementation classes that collectively are referred to as the JRas extensions. Many of these classes provide the appropriate implementations of loggers, handlers and formatters for use in a WebSphere Application Server environment. As previously noted, this collection of classes is targeted at an Integrated mode of operation. If you choose to use the JRas extensions in either stand-alone or combined mode, you can reuse the logger and manager class provided by the extensions, but you must provide your own implementations of handlers and formatters.

WebSphere Application Server Message and Trace loggers

The message and trace loggers provided by the stand-alone JRas logging toolkit cannot be directly used in the WebSphere Application Server environment. The JRas extensions provide the appropriate logger implementation classes. Instances of these message and trace logger classes are obtained directly and exclusively from the WebSphere Application Server Manager class, described below. You cannot directly instantiate message and trace loggers. Obtaining loggers in any manner other than directly from the Manager is not allowed. Doing so is a direct violation of the programming model.

The message and trace loggers instances obtained from the WebSphere Application Server Manager class are subclasses of the `RASMessageLogger()` and `RASTraceLogger()` classes provided by the stand-alone JRas logging toolkit. The `RASMessageLogger()` and `RASTraceLogger()` classes define the set of methods that are directly available. Public methods introduced by the JRas extensions logger subclasses cannot be called directly by user code. Doing so is a violation of the programming model.

Loggers are named objects and are identified by name. When the Manager class is called to obtain a logger, the caller is required to specify a name for the logger. The Manager class maintains a name-to-logger instance mapping. Only one instance of a named logger will ever be created within the lifetime of a process. The first call to the Manager with a particular name will result in the logger being created and configured by the Manager. The Manager will cache a reference to the instance, then return it to the caller. Subsequent calls to the Manager that specify the same name will result in a reference to the cached logger being returned. Separate namespaces are maintained for message and trace loggers. This means a single name can be used to obtain both a message logger and a trace logger from the Manager, without ambiguity, and without causing a namespace collision.

In general, loggers have no predefined granularity or scope. A single logger could be used to instrument an entire application. Or users may determine that having a logger per class is more desirable. Or the appropriate granularity may lie somewhere in between. Partitioning an application into logging domains is rightfully determined by the application writer.

The WebSphere Application Server logger classes obtained from the Manager are thread-safe. Although the loggers provided as part of the stand-alone JRas logging toolkit implement the serializable interface, in fact loggers are not serializable. Loggers are stateful objects, tied to a Java virtual machine instance and are not serializable. Attempting to serialize a logger is a violation of the programming model.

Please note that there is no provision for allowing users to provide their own logger subclasses for use in a WebSphere Application Server environment.

WebSphere Application Server handlers

WebSphere Application Server provides the appropriate handler class that is used to write message and trace events to the WebSphere Application Server run-time logs. You cannot configure the WebSphere Application Server handler to write to any other destination. The creation of a WebSphere Application Server handler is a restricted operation and not available to user code. Every logger obtained from the Manager comes preconfigured with an instance of this handler already installed. You can remove the WebSphere Application Server handler from a logger when you want to run in stand-alone mode. Once you have removed it, you cannot add the WebSphere Application Server handler again to the logger from which it was removed (or any other logger). Also, you cannot directly call any method on the WebSphere Application Server handler. Attempting to create an instance of the WebSphere Application Server handler, to call methods on the WebSphere Application Server handler or to add a WebSphere Application Server handler to a logger by user code is a violation of the programming model.

WebSphere Application Server formatters

The WebSphere Application Server handler comes preconfigured with the appropriate formatter for data that is written to WebSphere Application Server logs. The creation of a WebSphere Application Server formatter is a restricted operation and not available to user code. No mechanism exists that allows the user to obtain a reference to a formatter installed in a WebSphere Application Server handler, or to change the formatter a WebSphere Application Server handler is configured to use.

WebSphere Application Server manager

WebSphere Application Server provides a Manager class located in the `com.ibm.websphere.ras` package. All message and trace loggers must be obtained from this Manager. A reference to the Manager is obtained by calling the static `Manager.getManager()` method. Message loggers are obtained by calling the `createRASMessageLogger()` method on the Manager. Trace loggers are obtained by calling the `createRASTraceLogger()` method on the Manager class.

The manager also supports a *group* abstraction that is useful when dealing with trace loggers. The group abstraction allows multiple, unrelated trace loggers to be registered as part of a named entity called a group. WebSphere Application Server provides the appropriate systems management facilities to manipulate the trace setting of a group, similar to the way the trace settings of an individual trace logger.

For example, suppose component A consist of 10 classes. Suppose each class is configured to use a separate trace logger. Suppose all 10 trace loggers in the component are registered as members of the same group (for example Component_A_Group). You can then turn on trace for a single class. Or you can turn on trace for all 10 classes in a single operation using the group name if you want a component trace. Group names are maintained within the namespace for trace loggers.

Extending the JRas framework:

Since the Jras extensions classes do not provide the flexibility and behavior required for many scenarios, a variety of extension points have been defined. You are allowed to write your own implementation classes to obtain the required behavior.

In general, the JRas extensions require you to call the Manager class to obtain a message logger or trace logger. No provision is made to allow you to provide your own message or trace logger subclasses. In general, user-provided extensions cannot be used to affect the integrated mode of operation.The behavior of the integrated mode of operation is solely determined by the WebSphere Application Server run-time and the JRas extensions classes.

Handlers

The stand-alone JRas logging toolkit defines the `RASIHandler` interface. All handlers must implement this interface. You can write your own handler classes that implement the `RASIHandler` interface. You should directly create instances of user-defined handlers and add them to the loggers obtained from the Manager.

The stand-alone JRas logging toolkit provides several handler implementation classes. These handler classes are inappropriate for usage in the J2EE environment. You cannot directly use or subclass any of the Handler classes provided by the stand-alone JRas logging toolkit. Doing so is a violation of the programming model.

Formatters

The stand-alone JRas logging toolkit defines the `RASIFormatter` interface. All formatters must implement this interface. You can write your own formatter classes that implement the `RASIFormatter` interface. You can only add these classes to a user-defined handler. WebSphere Application Server handlers cannot be configured to use user-defined formatters. Instead, directly create instances of your formatters and add them to the your handlers appropriately.

As with handlers, the stand-alone JRas logging toolkit provides several formatter implementation classes. Direct usage of these formatter classes is not supported.

Message event types

The stand-alone JRas toolkit defines message event types in the `RASIMessageEvent` interface. In addition, the WebSphere Application Server reserves a range of message event types for future use. The `RASIMessageEvent` interface defines three types, with values of 0x01, 0x02, and 0x04. The values 0x08 through 0x8000 are reserved for future use. You can provide your own message event types by extending this interface appropriately. User-defined message types must have a value of 0x1000 or greater.

Message loggers retrieved from the Manager have their message masks set to *pass* or process all message event types defined in the `RASIMessageEvent` interface. In order to process user-defined message types, you must manually set the message logger mask to the appropriate state by user code after the message logger has been obtained from the Manager. WebSphere Application Server does not provide any built-in systems management support for managing any message types.

Message event objects

The stand-alone JRas toolkit provides a `RASMessageEvent` implementation class. When a message logging method is called on the message logger, and the message type is currently enabled, the logger creates and distributes an event of this class to all handlers currently registered with that logger.

You can provide your own message event classes, but they must implement the `RASIEvent` interface. You must directly create instances of such user-defined message event classes. Once it is created, pass your message event to the message logger by calling the message logger's `fireRASEvent()` method directly. WebSphere Application Server message loggers cannot directly create instances of user-defined types in response to calling a logging method (`msg()`, `message()`...) on the logger. In addition, instances of user-defined message types are never processed by the WebSphere Application Server handler. You cannot create instances of the `RASMessageEvent` class directly.

Trace event types

The stand-alone JRas toolkit defines trace event types in the `RASITraceEvent` interface. You can provide your own trace event types by extending this interface appropriately. In such a case you must ensure that the values for the user-defined trace event types do not collide with the values of the types defined in the `RASITraceEvent` interface.

Trace loggers retrieved from the Manager typically have their trace masks set to reject all types. A different starting state can be specified by using WebSphere Application Server systems management facilities. In addition, the state of the trace mask for a logger can be changed at run-time using WebSphere Application Server systems management facilities.

In order to process user-defined trace types, the trace logger mask must be manually set to the appropriate state by user code. WebSphere Application Server systems management facilities cannot be used to manage user-defined trace types, either at start time or run-time.

Trace event objects

The stand-alone JRas toolkit provides a `RASTraceEvent` implementation class. When a trace logging method is called on the WebSphere Application Server trace logger and the type is currently enabled, the logger creates and distributes an event of this class to all handlers currently registered with that logger.

You can provide your own trace event classes. Such trace event classes must implement the `RASIEvent` interface. You must create instances of such user-defined event classes directly. Once it is created, pass the trace event to the trace logger by calling the trace logger's `fireRASEvent()` method directly. WebSphere Application Server trace loggers cannot directly create instances of user-defined types in response to calling a trace method (`entry()`, `exit()`, `trace()`) on the trace logger. In addition, instances of user-defined trace types are never processed by the WebSphere Application Server handler. You cannot create instances of the `RASTraceEvent` class directly.

User defined types, user defined events and WebSphere Application Server

By definition, the WebSphere Application Server handler will process user-defined message or trace types, or user-defined message or trace event classes. Message and trace entries of either a user-defined type or user-defined event class cannot be written to the WebSphere Application Server run-time logs.

Writing User Extensions:
General Considerations

You can configure the WebSphere Application Server to use Java 2 security to restrict access to protected resources such as the file system and sockets. Since user written extensions typically access such protected resources, user written extensions must contain the appropriate security checking calls, using

AccessController `doPrivileged()` calls. In addition, the user written extensions must contain the appropriate policy file. In general, it is recommended that you locate user written extensions in a separate package. It is your responsibility to restrict access to the user written extensions appropriately.

Writing a handler

User written handlers must implement the `RASIHandler` interface. The `RASIHandler` interface extends the `RASIMaskChangeGenerator` interface, which extends the `RASIObject` interface. A short discussion of the methods introduced by each of these interfaces follows, along with implementation pointers. For more in depth information on any of the particular interfaces or methods, see the corresponding product Javadoc.

RASIObject interface

The `RASIObject` interface is the base interface for stand-alone JRas logging toolkit classes that are stateful or configurable, such as loggers, handlers and formatters.

- The stand-alone JRas logging tookit supports rudimentary properties-file based configuration. To implement this configuration support, the configuration state is stored as a set of key-value pairs in a properties file. The methods*`public Hashtable getConfig()`* and *`public void setConfig(Hashtable ht)`* are used to get and set the configuration state. The JRas extensions do not support properties based configuration and it is recommended that these methods be implemented as no-operations. You can implement your own properties based configuration using these methods.
- Loggers, handlers and formatters can be named objects. For example, the JRas extensions require the user to provide a name for the loggers that are retrieved from the manager. You can name your handlers. The methods *`public String getName()`* and *`public void setName(String name)`* are provided to get or set the name field. The JRas extensions currently do not call these methods on user handlers. You can implement these methods as you want, including as no operations.
- Loggers, handlers and formatters can also contain a description field. The methods *`public String getDescription()`* and *`public void setDescription(String desc)`* can be used to get or set the description field. The JRas extensions currently do not use the description field. You can implement these methods as you want, including as no operations.
- The method *`public String getGroup()`* is provided for usage by the `RASManager`. Since the JRas extensions provide their own Manager class, this method is never called. It is recommended you implement this as a no-operation.

RASIMaskChangeGenerator interface

The `RASIMaskChangeGenerator` interface is the interface that defines the implementation methods for filtering of events based on a mask state. This means that it is currently implemented by both loggers and handlers. By definition, an object that implements this interface contains both a message mask and a trace mask, although both need not be used. For example, message loggers contain a trace mask, but the trace mask is never used since the message logger never generates trace events. Handlers however can actively use both mask values. For example a single handler could handle both message and trace events.

- The methods *`public long getMessageMask()`* and *`public void setMessageMask(long mask)`* are used to get or set the value of the message mask. The methods *`public long getTraceMask()`* and *`public void setTraceMask(long mask)`* are used to get or set the value of the trace mask.

In addition, this interface introduces the concept of *calling back* to interested parties when a mask changes state. The callback object must implement the `RASIMaskChangeListener` interface.

- The methods *`public void addMaskChangeListener(RASIMaskChangeListener listener)`* and *`public void removeMaskChangeListener(RASIMaskChangeListener listener)`* are used to add or remove listeners to the handler. The method *`public Enumeration getMaskChangeListeners()`* returns an Enumeration over the list of currently registered listeners. The method *`public void fireMaskChangedEvent(RASMaskChangeEvent mc)`* is used to call back all the registered listeners to inform them of a mask change event.

For efficiency reasons, the Jras extensions message and trace loggers implement the RASIMaskChangeListener interface. The logger implementations maintain a "composite mask" in addition to the logger's own mask. The logger's composite mask is formed by logically *or'ing* the appropriate masks of all handlers that are registered to that logger, then *and'ing* the result with the logger's own mask. For example, the message logger's composite mask is formed by or'ing the message masks of all handlers registered with that logger, then and'ing the result with the logger's own message mask.

This means that all handlers are required to properly implement these methods. In addition, when a user handler is instantiated, the logger it is to be added to should be registered with the handler using the *addMaskChangeListener()* method. When either the message mask or trace mask of the handler is changed, the logger must be called back to inform it of the mask change. This allows the logger to dynamically maintain the composite mask.

The RASMaskChangedEvent class is defined by the stand-alone JRas logging toolkit. Direct usage of that class by user code is allowed in this context.

In addition the RASIMaskChangeGenerator introduces the concept of caching the names of all message and trace event classes that the implementing object will process. The intent of these methods is to allow a management program such as a GUI to retrieve the list of names, introspect the classes to determine the event types that they might possibly process and display the results. The JRas extensions do not ever call these methods, so they can be implemented as no operations, if desired.

- The methods *public void addMessageEventClass(String name)* and *public void removeMessageEventClass(String name)* can be called to add or remove a message event class name from the list. The method *public Enumeration getMessageEventClasses()* will return an enumeration over the list of message event class names. Similarly, the *public void addTraceEventClass(String name)* and *public void removeTraceEventClass(String name)* can be called to add or remove a trace event class name from the list. The *method public Enumeration getTraceEventClasses()* will return an enumeration over the list of trace event class names.

RASIHandler interface

The RASIHandler interface introduces the methods that are specific to the behavior of a handler.

The RASIHandler interface as provided by the stand-alone JRas logging toolkit supports handlers that run in either a synchronous or asynchronous mode. In asynchronous mode, events are typically queued by the calling thread and then written by a worker thread. Since spawning of threads is not allowed in the WebSphere Application Server environment, it is expected that handlers will not queue or batch events, although this is not expressly prohibited.

- The methods *public int getMaximumQueueSize()* and *public void setMaximumQueueSize(int size)* throw IllegalStateException are provided to manage the maximum queue size. The method *public int getQueueSize()* is provided to query the actual queue size.
- The methods *public int getRetryInterval()* and *public void setRetryInterval(int interval)* support the notion of error retry, which again implies some type of queueing.
- The methods *public void addFormatter(RASIFormatter formatter)*, *public void removeFormatter(RASIFormatter formatter)* and *public Enumeration getFormatters()* are provided to manage the list of formatters that the handler can be configured with. Different formatters can be provided for different event classes, if appropriate.
- The methods *public void openDevice()*, *public void closeDevice()* and *public void stop()* are provided to manage the underlying device that the handler abstracts.
- The methods *public void logEvent(RASIEvent event)* and *public void writeEvent(RASIEvent event)* are provided to actually pass events to the handler for processing.

Writing a formatter

User written formatters must implement the RASIFormatter interface. The RASIFormatter interface extends the RASIObject interface. The implementation of the RASIObject interface is the same for both handlers and formatters. A short discussion of the methods introduced by the `RASIFormatter` interface follows. For more in depth information on the methods introduced by this interface, see the corresponding product javadoc.

RASIFormatter interface

- The methods *public void setDefault(boolean flag)* and *public boolean isDefault()* are used by the concrete RASHandler classes provided by the stand-alone JRas logging toolkit to determine if a particular formatter is the default formatter. Since these `RASHandler` classes must never be used in a WebSphere Application Server environment, the semantic significance of these methods can be determined by the user.
- The methods *public void addEventClass(String name)*, *public void removeEventClass(String name)* and *public Enumeration getEventClasses()* are provided to determine which event classes a formatter can be used to format. You can provide the appropriate implementations as you see fit.
- The method *public String format(RASIEvent event)* is called by handler objects and returns a formatted String representation of the event.

Programming model summary:

The programming model described in this section builds upon and summarizes some of the concepts already introduced. This section also formalizes usage requirements and restrictions. Use of the WebSphere Application Server JRas extensions in a manner that does not conform to the following programming guidelines is prohibited.

As described previously, you can use the WebSphere Application Server JRas extensions in three distinct operational modes. The programming models concepts and restrictions apply equally across all modes of operation.

- You must not use implementation classes provided by the stand-alone JRas logging toolkit directly, unless specifically noted otherwise. Direct usage of those classes is not supported. IBM Support will provide no diagnostic aid or bug fixes relating to direct usage of classes provided by the stand-alone JRas logging toolkit.
- You must obtain message and trace loggers directly from the Manager class. You cannot directly instantiate loggers.
- There is no provision that allows you to replace the WebSphere Application Server message and trace logger classes.
- You must guarantee that the logger names passed to the Manager are unique, and follow the naming constraints documented below. Once a logger is obtained from the Manager, you must not attempt to change the name of the logger by calling the `setName()` method.
- Named loggers can be used more than once. For any given name, the first call to the Manager results in the Manager creating a logger that is associated with that name. Subsequent calls to the Manager that specify the same name result in a reference to the existing logger being returned.
- The Manager maintains a hierarchical namespace for loggers. It is recommended but not required that a dot-separated, fully qualified class name be used to identify any given logger. Other than dots or periods, logger names cannot contain any punctuation characters, such as asterisk (*), comma(.), equals sign(=), colon(:), or quotes.
- Group names must comply with the same naming restrictions as logger names.
- The loggers returned from the Manager are subclasses of the RASMessageLogger and RASTraceLogger provided by the stand-alone JRas logging toolkit. You are allowed to call any public method defined by the RASMessageLogger and RASTraceLogger classes. You are not allowed to call any public method introduced by the provided subclasses.
- If you want to operate in either stand-alone or combined mode, you must provide your own Handler and Formatter subclasses. You are not allowed to use the Handler and Formatter classes provided by the stand-alone JRas logging toolkit. User written Handlers and Formatters must conform to the documented guidelines.
- Loggers obtained from the Manager come with a WebSphere Application Server handler installed. This handler will write message and trace records to logs defined by the WebSphere Application Server runtime. Manage these logs using the provided systems management interfaces.

- You can programmatically add and remove user-defined Handlers from a logger at any time. Multiple additions and removals of user defined handlers are allowed. You are responsible for creating an instance of the handler to add, configuring the handler by setting the handler's mask value and formatter appropriately, then adding the handler to the logger using the `addHandler()` method. You are responsible for programmatically updating the masks of user-defined handlers as appropriate.
- You may get a reference to the handler installed within a logger by calling the `getHandlers()` method on the logger and processing the results. You must not call any methods on the handler obtained in this fashion. You are allowed to remove the WebSphere Application Server handler from the logger by calling the logger's `removeHandler()` method, passing in the reference to the WebSphere Application Server handler. Once removed, the WebSphere Application Server handler cannot be re-added to the logger.
- You are allowed to define your own message type. The behavior of user-defined message types and restrictions on their definitions is discussed in Extending the JRas framework.
- You are allowed to define your own message event classes. The usage of user-defined message event classes is discussed in Extending the JRas framework.
- You are allowed to define your own trace types. The behavior of user-defined trace types and restrictions on your definitions is discussed in Extending the JRas framework.
- You are allowed to define your own trace event classes. The usage of user-defined trace event classes is discussed in Extending the JRas framework.
- You must programmatically maintain the bits in the message and trace logger masks that correspond to any user-defined types. If WebSphere Application Server facilities are being used to manage the predefined types, these updates must not modify the state of any of the bits corresponding to those types. If you are assuming ownership responsibility for the predefined types then you can change all bits of the masks.

JRas Messages and Trace event types

This section describes JRas message and trace event types.

Event types

The base message and trace event types defined by the stand-alone JRas logging toolkit are not the same as the ″native″ types recognized by the WebSphere Application Server run-time. Instead the basic JRas types are mapped onto the native types. This mapping may vary by platform or edition. The mapping is discussed below.

Platform Message Event Types

The message event types that are recognized and processed by the WebSphere Application Server runtime are defined in the RASIMessageEvent interface provided by the stand-alone JRas logging toolkit. These message types are mapped onto the native message types as follows.

WebSphere Application Server native type	JRas RASIMessageEvent type
`Audit`	TYPE_INFO, TYPE_INFORMATION
`Warning`	TYPE_WARN, TYPE_WARNING
`Error`	TYPE_ERR, TYPE_ERROR

Platform Trace Event Types

The trace event types recognized and processed by the WebSphere Application Server runtime are defined in the `RASITraceEvent` interface provided by the stand-alone JRas logging toolkit. The `RASITraceEvent` interface provides a rich and overly complex set of types. This interface defines both a simple set of levels, as well as a set of enumerated types.

- For a user who prefers a simple set of levels, `RASITraceEvent` provides `TYPE_LEVEL1`, `TYPE_LEVEL2`, and `TYPE_LEVEL3`. The implementations provide support for this set of levels. The levels are hierarchical (that is, enabling level 2 will also enable level 1, enabling level 3 also enables levels 1 and 2).

- For users who prefer a more complex set of values that can be *OR'd* together, `RASITraceEvent` provides `TYPE_API`, `TYPE_CALLBACK`, `TYPE_ENTRY_EXIT`, `TYPE_ERROR_EXC`, `TYPE_MISC_DATA`, `TYPE_OBJ_CREATE`, `TYPE_OBJ_DELETE`, `TYPE_PRIVATE`, `TYPE_PUBLIC`, `TYPE_STATIC`, and `TYPE_SVC`.

The trace event types are mapped onto the native trace types as follows:

Mapping WebSphere Application Server trace types to JRas `RASITraceEvent` "Level" types.

WebSphere Application Server native type	JRas RASITraceEvent level type
`Event`	TYPE_LEVEL1
`EntryExit`	TYPE_LEVEL2
`Debug`	TYPE_LEVEL3

Mapping WebSphere Application Server trace types to JRas `RASITraceEvent` enumerated types.

WebSphere Application Server native type	JRas RASITraceEvent enumerated types
`Event`	TYPE_ERROR_EXC, TYPE_SVC, TYPE_OBJ_CREATE, TYPE_OBJ_DELETE
`EntryExit`	TYPE_ENTRY_EXIT, TYPE_API, TYPE_CALLBACK, TYPE_PRIVATE, TYPE_PUBLIC, TYPE_STATIC
`Debug`	TYPE_MISC_DATA

For simplicity, it is recommended that one or the other of the tracing type methodologies is used consistently throughout the application. For users who decide to use the non-level types, it is further recommended that you choose one type from each category and use those consistently throughout the application to avoid confusion.

Message and Trace parameters

The various message logging and trace method signatures accept parameter types of `Object`, `Object[]` and `Throwable`. WebSphere Application Server will process and format the various parameter types as follows.

Primitives
: Primitives, such as `int` and `long` are not recognized as subclasses of `Object` and cannot be directly passed to one of these methods. A primitive value must be transformed to a proper `Object` type (`Integer`, `Long`) before being passed as a parameter.

`Object`
: `toString()` is called on the object and the resulting `String` is displayed. The `toString()` method should be implemented appropriately for any object passed to a message logging or trace method. It is the responsibility of the caller to guarantee that the `toString()` method does not display confidential data such as passwords in clear text, and does not cause infinite recursion.

`Object[]`
: The `Object[]` is provided for the case when more than one parameter is passed to a message logging or trace method. `toString()` is called on each `Object` in the array. Nested arrays are not handled. (i.e. none of the elements in the `Object` array should be an array).

`Throwable`
: The stack trace of the `Throwable` is retrieved and displayed.

Array of Primitives
: An array of primitive (e.g. `byte[]`, `int[]` is recognized as an `Object`, but is treated somewhat as a second cousin of `Object` by Java code. In general, arrays of primitives should be avoided, if possible. If arrays of primitives are passed, the results are indeterminate and may change depending on the type of array passed, the API used to pass the array and the release of the

product. For consistent results, user code should preprocess and format the primitive array into some type of String form before passing it to the method. If such preprocessing is not performed, the following may result.

- [B@924586a0b - This is deciphered as ″a byte array at location X″. This is typically returned when an array is passed as a member of an `Object[]`. It is the result of calling `toString()` on the `byte[]`.
- Illegal trace argument : array of long. This is typically returned when an array of primitives is passed to a method taking an `Object`.
- 01040703... : the hex representation of an array of bytes. Typically this may be seen when a byte array is passed to a method taking a single `Object`. This behavior is subject to change and should not be relied on.
- ″1″ ″2″ ... : The String representation of the members of an `int[]` formed by converting each element to an Integer and calling `toString` on the Integers. This behavior is subject to change and should not be relied on.
- [Ljava.lang.Object;@9136fa0b : An array of objects. Typically this is seen when an array containing nested arrays is passed.

Controlling message logging

Writing a message to a WebSphere Application Server log requires that the message type passes three levels of filtering or screening.

1. The message event type must be one of the message event types defined in the `RASIMessageEvent` interface.
2. Logging of that message event type must be enabled by the state of the message logger's mask.
3. The message event type must pass any filtering criteria established by the WebSphere Application Server run-time itself.

When a WebSphere Application Server logger is obtained from the Manager, the initial setting of the mask is to forward all native message event types to the WebSphere Application Server handler. It is possible to control what messages get logged by programmatically setting the state of the message logger's mask.

Some editions of the product allow the user to specify a message filter level for a server process. When such a filter level is set, only messages at the specified severity levels are written to WebSphere Application Server logs. This means that messages types that pass the message logger's mask check may be filtered out by the WebSphere Application Server itself.

Controlling Tracing

Each edition of the product provides a mechanism for enabling or disabling trace. The various editions may support static trace enablement (trace settings are specified before the server is started), dynamic trace enablement (trace settings for a running server process can be dynamically modified) or both.

Writing a trace record to a WebSphere Application Server requires that the trace type passes three levels of filtering or screening.

1. The trace event type must be one of the trace event types defined in the `RASITraceEvent` interface.
2. Logging of that trace event type must be enabled by the state of the trace logger's mask.
3. The trace event type must pass any filtering criteria established by the WebSphere Application Server run-time itself.

When a logger is obtained from the Manager, the initial setting of the mask is to suppress all trace types. The exception to this rule is the case where the WebSphere Application Server run-time supports static trace enablement and a non-default startup trace state for that trace logger has been specified. Unlike message loggers, the WebSphere Application Server may dynamically modify the state of a trace loggers trace mask. WebSphere Application Server will only modify the portion of the trace logger's mask corresponding to the values defined in the `RASITraceEvent` interface. WebSphere Application Server will not modify undefined bits of the mask that may be in use for user defined types.

When the dynamic trace enablement feature available on some platforms is used, the trace state change is reflected both in the Application Server run-time and the trace loggers trace mask. If user code programmatically changes the bits in the trace mask corresponding to the values defined by in the `RASITraceEvent` interface, the trace logger's mask state and the run-time state will become unsynchronized and unexpected results will occur. Therefore, programmatically changing the bits of the mask corresponding to the values defined in the `RASITraceEvent` interface is not allowed.

Instrumenting an application with JRas extensions

To instrument an application using the WebSphere Application Server JRas extensions, perform the following steps:

1. Determine the mode the extensions will be used in: integrated, stand-alone or combined.
2. If the extensions will be used in either stand-alone or combined mode, create the necessary handler and formatter classes.
3. If localized messages will be used by the application, create a resource bundle as described in Creating JRas resource bundles and message files.
4. In the application code, get a reference to the Manager class and create the manager and logger instances as described in Creating JRas manager and logger instances.
5. Insert the appropriate message and trace logging statements in the application as described in Creating JRas manager and logger instances.

Creating JRas resource bundles and message files:

The WebSphere Application Server message logger provides the `message()` and `msg()` methods to allow the user to log localized messages. In addition, it provides the `textMessage()` method for logging of messages that are not localized. Applications can use either or both, as appropriate.

The mechanism for providing localized messages is the Resource Bundle support provided by the IBM Developer Java Technology Edition. If you are not familiar with resource bundles as implemented by the Developer's Kit, you can get more information from various texts, or by reading the Javadoc for the `java.util.ResourceBundle`, `java.util.ListResourceBundle` and `java.util.PropertyResourceBundle` classes, as well as the `java.text.MessageFormat` class.

The `PropertyResourceBundle` is the preferred mechanism to use. In addition, note that the JRas extensions do not support the extended formatting options such as {1, date} or {0,number, integer} that are provided by the `MessageFormat` class.

You can forward messages that are written to the internal WebSphere Application Server logs to other processes for display. For example, messages displayed on the administrator console, which can be running in a different location than the server process, can be localized using the *late binding* process. Late binding means that WebSphere Application Server does not localize messages when they are logged, but defers localization to the process that displays the message.

To properly localize the message, the displaying process must have access to the resource bundle where the message text is stored. This means that you must package the resource bundle separately from the application, and install it in a location where the viewing process can access it. If you do not want to take these steps, you can use the early binding technique to localize messages as they are logged.

The two techniques are described as follows:

Early binding

The application must localize the message before logging it. The application looks up the localized text in the resource bundle and formats the message. When formatting is complete, the application logs the message using the `textMessage()` method. Use this technique to package the application's resource bundles with the application.

Late binding

The application can choose to have the WebSphere Application Server runtime localize the message in the process where it is displayed. Using this technique,the resource bundles are packaged in a stand-alone `.jar` file, separately from the application. You must then install the resource bundle `.jar` file on every machine in the installation from which an administrator's console or log viewing program might be run. You must install the `.jar` file in a directory that is part of the extensions classpath. In addition, if you forward logs to IBM service, you must also forward the `.jar` file containing the resource bundles.

To create a resource bundle, perform the following steps.

1. Create a text properties file that lists message keys and the corresponding messages. The properties file must have the following characteristics:
   - Each property in the file is terminated with a line-termination character.
   - If a line contains only white space, or if the first non-white space character of the line is the pound sign symbol (`#`) or exclamation mark (`!`), the line is ignored. The `#` and `!` characters can therefore be used to put comments into the file.
   - Each line in the file, unless it is a comment or consists only of white space, denotes a single property. A backslash (`\`) is treated as the line-continuation character.
   - The syntax for a property file consists of a key, a separator, and an element. Valid separators include the equal sign (`=`), colon (`:`), and white space ().
   - The key consists of all characters on the line from the first non-white space character to the first separator. Separator characters can be included in the key by escaping them with a backslash (`\`), but doing this is not recommended, because escaping characters is error prone and confusing. It is instead recommended that you use a valid separator character that does not appear in any keys in the properties file.
   - White space after the key and separator is ignored until the first non-white space character is encountered. All characters remaining before the line-termination character define the element.

   See the Java documentation for the `java.util.Properties` class for a full description of the syntax and construction of properties files.
2. The file can then be translated into localized versions of the file with language-specific file names (for example, a file named `DefaultMessages.properties` can be translated into DefaultMessages_de.properties for German and DefaultMessages_ja.properties for Japanese).
3. When the translated resource bundles are available, write them to a system-managed persistent storage medium. Resource bundles are then used to convert the messages into the requested national language and locale.
4. When a message logger is obtained from the JRas manager, it can be configured to use a particular resource bundle. Messages logged via the `message()` API will use this resource bundle when message localization is performed. At run time, the user's locale setting is used to determine the properties file from which to extract the message specified by a message key, thus ensuring that the message is delivered in the correct language.
5. If the message loggers `msg()` method is called, a resource bundle name must be explicitly provided.

The application locates the resource bundle based on the file's location relative to any directory in the classpath. For instance, if the property resource bundle named DefaultMessages.properties is located in the *baseDir*/*subDir1*/*subDir2*/`resources` directory and *baseDir* is in the class path, the name *subdir1*.*subdir2*.`resources.DefaultMessage` is passed to the message logger to identify the resource bundle.

Developing JRas resource bundles:
Resource bundle sample

You can create resource bundles in several ways. The best and easiest way is to create a properties file that supports a `PropertiesResourceBundle`. This sample shows how to create such a properties file.

For this sample, four localizable messages are provided. The properties file is created and the key-value pairs inserted into it. All the normal properties files conventions and rules apply to this file. In addition, the creator must be aware of other restrictions imposed on the values by the Java `MessageFormat` class. For example, apostrophes must be ″escaped″ or they will cause a problem. Also avoid use of non-portable characters. WebSphere Application Server does not support usage of extended formatting conventions that the `MessageFormat` class supports, such as {1, date} or {0,number, integer}.

Assume that the base directory for the application that uses this resource bundle is ″`baseDir`″ and that this directory will be in the classpath. Assume that the properties file is stored in a subdirectory of `baseDir` that is not in the classpath (e.g. `baseDir/subDir1/subDir2/resources`). In order to allow the messages file to be resolved, the name `subDir1.subDir2.resources.DefaultMessage` is used to identify the `PropertyResourceBundle` and is passed to the message logger.

For this sample, the properties file is named `DefaultMessages.properties`.

```
# Contents of DefaultMessages.properties file
MSG_KEY_00=A message with no substitution parameters.
MSG_KEY_01=A message with one substitution parameter: parm1={0}
MSG_KEY_02=A message with two substitution parameters: parm1={0}, parm2 = {1}
MSG_KEY_03=A message with three parameter: parm1={0}, parm2 = {1}, parm3={2}
```

Once the file `DefaultMessages.properties` is created, the file can be sent to a translation center where the localized versions will be generated.

Creating JRas manager and logger instances:

You can use the JRas extensions in integrated, stand-alone, or combined mode. Configuration of the application will vary depending on the mode of operation, but usage of the loggers to log message or trace entries is identical in all modes of operation.

Integrated mode is the default mode of operation. In this mode, message and trace events are sent to the WebSphere Application Server logs. See Setting up for integrated JRas operation for information on configuring for this mode of operation.

In the combined mode, message and trace events are logged to both WebSphere Application Server and user-defined logs. See Setting up for combined JRas operation for more information on configuring for this mode of operation.

In the stand-alone mode, message and trace events are logged only to user-defined logs. See Setting up for stand-alone JRas operation for more information on configuring for this mode of operation.

Using the message and trace loggers

Regardless of the mode of operation, the use of message and trace loggers is the same. See Creating JRas resource bundles and message files for more information on using message and trace loggers.

Using a message logger

The message logger is configured to use the `DefaultMessages` resource bundle. Message keys must be passed to the message loggers if the loggers are using the `message()` API.

```
msgLogger.message(RASIMessageEvent.TYPE_WARNING, this,
     methodName, "MSG_KEY_00");
... msgLogger.message(RASIMessageEvent.TYPE_WARN, this,
     methodName, "MSG_KEY_01", "some string");
```

If message loggers use the `msg()` API, you can specify a new resource bundle name.

```
msgLogger.msg(RASIMessageEvent.TYPE_ERR, this, methodName,
                  "ALT_MSG_KEY_00", "alternateMessageFile");
```

You can also log a text message. If you are using the `textMessage` API, no message formatting is done.

```
msgLogger.textMessage(RASIMessageEvent.TYPE_INFO, this, methodName,"String and Integer",
"A String", new Integer(5));
```

Using a trace logger

Since trace is normally disabled, trace methods should be guarded for performance reasons.

```
private void methodX(int x, String y, Foo z)
{
   // trace an entry point. Use the guard to make sure tracing is enabled.
Do this checking before we waste cycles gathering parameters to be traced.
   if (trcLogger.isLoggable(RASITraceEvent.TYPE_ENTRY_EXIT) {
        // since I want to trace 3 parameters, package them up in an Object[]
        Object[] parms = {new Integer(x), y, z};
        trcLogger.entry(RASITraceEvent.TYPE_ENTRY_EXIT, this, "methodX", parms);
  }
... logic
  // a debug or verbose trace point
  if (trcLogger.isLoggable(RASITraceEvent.TYPE_MISC_DATA) {
        trcLogger.trace(RASITraceEvent.TYPE_MISC_DATA, this, "methodX" "reached here");
  }
  ...
  // Another classification of trace event. Here an important state change
has been detected, so a different trace type is used.
  if (trcLogger.isLoggable(RASITraceEvent.TYPE_SVC) {
     trcLogger.trace(RASITraceEvent.TYPE_SVC, this, "methodX", "an important event");
  }
 ...
  // ready to exit method, trace. No return value to trace
    if (trcLogger.isLoggable(RASITraceEvent.TYPE_ENTRY_EXIT)) {
        trcLogger.exit(RASITraceEvent.TYPE_ENTRY_EXIT, this, "methodX");
   }
}
```

Setting up for integrated JRas operation:

In the integrated mode of operation, message and trace events are sent to WebSphere Application Server logs. This is the default mode of operation.

1. Import the requisite JRas extensions classes

   ```
   import com.ibm.ras.*;
   import com.ibm.websphere.ras.*;
   ```

2. Declare logger references.

   ```
   private RASMessageLogger msgLogger = null;
   private RASTraceLogger trcLogger = null;
   ```

3. Obtain a reference to the Manager and create the loggers. Since loggers are named singletons, you can do this in a variety of places. One logical candidate for enterprise beans is the `ejbCreate()` method. For example, for the enterprise bean named ″myTestBean″, place the following code in the `ejbCreate()` method.

   ```
   com.ibm.websphere.ras.Manager mgr = com.ibm.websphere.ras.Manager.getManager();
   msgLogger = mgr.createRASMessageLogger("Acme", "WidgetCounter", "RasTest",
            myTestBean.class.getName());
   // Configure the message logger to use the message file created
   // for this application.
   msgLogger.setMessageFile("acme.widgets.DefaultMessages");
   trcLogger = mgr.createRASTraceLogger("Acme", "Widgets", "RasTest",
            myTestBean.class.getName());
   mgr.addLoggerToGroup(trcLogger, groupName);
   ```

Setting up for combined JRas operation:

In combined mode, messages and trace are logged to both WebSphere Application Server logs and user-defined logs. The following sample assumes that you have written a user defined handler named `SimpleFileHandler` and a user defined formatter named `SimpleFormatter`. It also assumes that you are not using user defined types or events.

1. Import the requisite JRas extensions classes
   ```
   import com.ibm.ras.*;
   import com.ibm.websphere.ras.*;
   ```
2. Import the user handler and formatter.
   ```
   import com.ibm.ws.ras.test.user.*;
   ```
3. Declare the logger references.
   ```
   private RASMessageLogger msgLogger = null;
     private RASTraceLogger trcLogger = null;
   ```
4. Obtain a reference to the Manager, create the loggers and add the user handlers. Since loggers are named singletons, you can obtain a reference to the loggers in a number of places. One logical candidate for enterprise beans is the `ejbCreate()` method. Make sure that multiple instances of the same user handler are not accidentally inserted into the same logger. Your initialization code must handle this. The following sample is a message logger sample. The procedure for a trace logger is similar.
   ```
   com.ibm.websphere.ras.Manager mgr = com.ibm.websphere.ras.Manager.getManager();
       msgLogger = mgr.createRASMessageLogger("Acme", "WidgetCounter", "RasTest",
              myTestBean.class.getName());
       // Configure the message logger to use the message file defined
       // in the ResourceBundle sample.
       msgLogger.setMessageFile("acme.widgets.DefaultMessages");

      // Create the user handler and formatter. Configure the formatter,
      // then add it to the handler.
      RASIHandler handler = new SimpleFileHandler("myHandler", "FileName");
      RASIFormatter formatter = new SimpleFormatter("simple formatter");
       formatter.addEventClass("com.ibm.ras.RASMessageEvent");
       handler.addFormatter(formatter);

      // Add the Handler to the logger. Add the logger to the list of the
      //handlers listeners, then set the handlers
      // mask, which will update the loggers composite mask appropriately.
      // WARNING - there is an order dependency here that must be followed.
      msgLogger.addHandler(handler);
      handler.addMaskChangeListener(msgLogger);
      handler.setMessageMask(RASIMessageEvent.DEFAULT_MESSAGE_MASK);
   ```

Setting up for stand-alone JRas operation:

In stand-alone mode, messages and traces are logged only to user-defined logs. The following sample assumes that you have a user-defined handler named `SimpleFileHandler` and a user-defined formatter named `SimpleFormatter`. It is also assumes that no user-defined types or events are being used.

1. Import the requisite JRas extensions classes
   ```
   import com.ibm.ras.*;
   import com.ibm.websphere.ras.*;
   ```
2. Import the user handler and formatter.
   ```
   import com.ibm.ws.ras.test.user.*;
   ```
3. Declare the logger references.
   ```
   private RASMessageLogger msgLogger = null;
     private RASTraceLogger trcLogger = null;
   ```
4. Obtain a reference to the Manager, create the loggers and add the user handlers. Since loggers are named singletons, you can obtain a reference to the loggers in a number of places. One logical candidate for enterprise beans is the `ejbCreate()` method. Make sure that multiple instances of the

same user handler are not accidentally inserted into the same logger. Your initialization code must handle this. The following sample is a message logger sample. The procedure for a trace logger is similar.

```
com.ibm.websphere.ras.Manager mgr = com.ibm.websphere.ras.Manager.getManager();
    msgLogger = mgr.createRASMessageLogger("Acme", "WidgetCounter", "RasTest",
             myTestBean.class.getName());
    // Configure the message logger to use the message file defined in
    //the ResourceBundle sample.
    msgLogger.setMessageFile("acme.widgets.DefaultMessages");

    // Get a reference to the Handler and remove it from the logger.
    RASIHandler aHandler = null;
    Enumeration enum = msgLogger.getHandlers();
    while (enum.hasMoreElements()) {
         aHandler = (RASIHandler)enum.nextElement();
         if (aHandler instanceof WsHandler)
              msgLogger.removeHandler(wsHandler);
    }

    // Create the user handler and formatter. Configure the formatter,
    // then add it to the handler.
    RASIHandler handler = new SimpleFileHandler("myHandler", "FileName");
    RASIFormatter formatter = new SimpleFormatter("simple formatter");
    formatter.addEventClass("com.ibm.ras.RASMessageEvent");
    handler.addFormatter(formatter);

   // Add the Handler to the logger. Add the logger to the list of the
   // handlers listeners, then set the handlers
   // mask, which will update the loggers composite mask appropriately.
   // WARNING - there is an order dependency here that must be followed.
   msgLogger.addHandler(handler);
   handler.addMaskChangeListener(msgLogger);
   handler.setMessageMask(RASIMessageEvent.DEFAULT_MESSAGE_MASK);
```

Logging messages and trace data for Java server applications

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

Determining where to issue the messages

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

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

System performance when logging messages and trace data

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

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

Note: You can only use WebSphere Application Server for z/OS support for logging messages and trace data for Java applications, not for Java applets.

Issuing application messages to the MVS master console

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

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

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

Using a message logger:

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

To use a message logger, in your Java application:
1. Define the message logger.
2. Drive the method to instruct WebSphere Application Server for z/OS to create the message logger.
3. Code messages at appropriate points in your application.

Collecting job-related information with Systems Management Facility (SMF)

This article gives an overview of how to enable and use the System Management Facilities (SMF) to collect and record system and job-related information.

System Management Facilities (SMF) can be enabled to collect and record system and job-related information on the WebSphere for z/OS system. This information can be used to bill users, report system reliability, analyze your configuration, schedule work, identify system resource usage, and perform other performance-related tasks that your organization may require.

You can enable SMF recording for:
- Capacity planning, to determine:
  – How many transactions have run?
  – What is the average and maximum completion time for methods running on each server?
  – How many clients are attached to each server instance? Of these clients, how many are active?
- Application profiling:

- To show an application broken down into its component parts.
- To provide timing information on the application's component parts.

- Error reporting:
  - To detect and record soft failures (those that are generated through an exception or those that are performance-related).
  - To use this error information to trigger an event that will cause an action to occur once a threshold has been reached.

Setting up SMF

Here is a sample SMFPRMxx member that will create interval records every 2 minutes, and record the following SMF record types:
- 30 - Address space
- 70-79 - RMF
- 82 - Crypto
- 88-90 - System Logger, Usage & System Data
- 101 - DB2
- 110 - CICS
- 120 - WebSphere

```
ACTIVE                      /*ACTIVE SMF RECORDING*/
    DSNAME(&SYSNAME..MAN1, &SYSNAME..MAN2)       /*TWO MAN DATASETS */
    LISTDSN                     /* LIST DATA SET STATUS AT IPL*/
    NOPROMPT                    /* DON'T PROMPT THE OPERATOR  */
    INTVAL(02)                  /* SMF GLOBAL RECORDING INTERVAL */
    SYNCVAL(00)                 /* GLOBAL SYNC VALUE            */
    MAXDORM(3000)               /* WRITE AN IDLE BUFFER AFTER 30 MIN*/
    STATUS(010000)              /* WRITE SMF STATS AFTER 1 HOUR*/
    SID(&SYSNAME(1:4))          /* USE SYSNAME AS SID     */
    SUBSYS(STC,INTERVAL(SMF,SYNC),
                          TYPE(0,30,70:79,88:90,101,110,120))
```

Set the SMF recording interval to 2 minutes by using the 'SET SMF=xx' command to activate the SMFPRMxx member from SYSx.PARMLIB. Use the 'D SMF,O' command to display the parameters in effect.

Use a tool like WSWS to simulate an application stress load.

While the transactions are running, switch to SDSF and RMF to observe the transactions.

SDSF

Use the SDSF DA panel to see how many application server address spaces are active, and observe at the CPU%, ECPU% and SIO rate. Use the ″ENC″ panel to see the enclaves running and what service classes they are running under.

RMF

See Using RMF for instructions on starting and using RMF to monitor your transactions.

Using RMF

RMF can usually be started with the simple 'S RMF' command from the MVS console.

Here is a typical RMF procedure:

```
//RMF      PROC
//IEFPROC  EXEC PGM=ERBMFMFC,REGION=0M,PARM='MEMBER(XS)'
//IEFPARM  DD   DDNAME=IEFRDER
//IEFRDER  DD   DSN=SYS1.PARMLIB,DISP=SHR
```

The following is a copy of the PARMLIB member ERBRMFXS: (parmeters beginning with a /* are not used in this example but may be useful to you.

```
CPU                        /* COLLECT CPU STATISTICS        */
     CHAN                       /* COLLECT CHANNEL STATISTICS    */
     CYCLE(1000)                /* SAMPLE AT 1 TIME / SECOND     */
     DEVICE(NOCHRDR)            /* NO CHARACTER RDR DEV STATS    */
     DEVICE(COMM)               /* ADDED COMM FOR 37X5           */
     DEVICE(DASD)               /* COLLECT DIRECT ACCESS DEVICE  */
     /* STATISTICS                 */
     DEVICE(NOGRAPH)            /* NO GRAPHICS DEVICE STATISTICS */
     DEVICE(NOTAPE)             /* NO TAPE DEVICE STATISTICS     */
     DEVICE(NOUNITR)            /* NO UNIT RECORD DEVICE STATS   */
     ENQ(SUMMARY)               /* ENQ REPORTING                 */
     INTERVAL(15M)              /* REPORT AT  15 MIN INTERVALS   */
     IOQ(DASD)                  /* I/O Q'ING FOR DEV IN LOG CU   */
     IOQ(COMM)                  /* I/O Q'ING FOR DEV IN LOG CU   */
     NOVSTOR                    /* NO RMF 3.2 AND LATER REL */
     OPTIONS                    /* OPERATOR MAY CHG RMF OPTIONS  */
     PAGING                     /* COLLECT PAGING STATISTICS     */
     PAGESP                     /* COLLECT PAGE/SWAP DATASET STAT*/
     RECORD                     /* RECORD INTO SMF DATASET       */
     NOSTOP                     /* STOP AFTER  90 MINUTES        */
     SYNC(SMF)                  /* INTERVAL SYNCED WITH SMF      */

     SYSOUT(H)                  /* SYSOUT CLASS OF OUTPUT REPORT */
     WKLD(PERIOD,SYSTEM)        /* COLLECT WORKLOAD MANAGER
                                   STATISTICS AND REPORT AT THE
                                   PERIOD LEVEL + TOTAL LINE     */
    TRACE(CCVUTILP)
```

The Monitor III data gatherer can be started after RMF with the ’F RMF,S III’ modify command.

To use the RMF Monitor 3 display, go to the ″Sysplex Summary″ display as follows:
1. Type RMF on ISPF command line.
2. Type 3 to see Monitor III choices.
3. Type S to get Sysplex reports.
4. Type 1 to see the Sysplex Summary report.

You can scroll back and forth in time with PF10 and PF11, or over-type the time field.

Look at the transactions in the WebSphere Application Server service and reporting class and note the Average Response time, Transactions per second, and Performance Index. You may also explore further in RMF Monitor III to see the ″System Information″ report.

SMF record types

This article describes the SMF record types.

Information resulting from the SMF data gathering process for WebSphere Application Server for z/OS are held in SMF record type 120. Two types of SMF records can be produced: activity records and interval records.

- Activity records are gathered as each activity within a server is completed. An activity is a logical unit of business function. It can be a server or user-initiated transaction.
- Interval records consist of data gathered at installation-specified intervals and provide capacity planning and reliability information.

Six records can be produced:
- the Server Activity record: Subtype 1
- the Server Interval record: Subtype 3

- the J2EE Container Activity Record: Subtype 5
- the J2EE Container Interval Record: Subtype 6
- the WebContainer Activity record: Subtype 7
- the WebContainer Interval record: Subtype 8

Server activity record

This file describes the server activity record

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

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

You can activate this record through the administrative console by setting server_SMF_server_activity_enabled=1.

Server interval record

This file describes the Server interval record

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

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

You can activate this record through the administrative console by setting server_SMF_server_interval_enabled=1. You can specify an interval through the administrative console by setting server_SMF_interval_length=n, where n is the desired number of seconds.

J2EE container activity record

This file describes the J2EE container activity record

The purpose of the J2EE container activity SMF record is to record activity within a J2EE container that is located inside the WebSphere Application Server transaction server.

This record can be used to perform basic charge-back accounting, application profiling, problem determination, and capacity planning. A single record is created for each activity that is run within a J2EE container located inside a WebSphere Application Server transaction server.

You can activate this record through the administrative console by setting server_SMF_container_activity_enabled=1.

J2EE container interval record

This file describes the J2EE container interval record

The purpose of the J2EE container interval SMF record is to record activity within a J2EE container that is located inside the WebSphere Application Server transaction server.

This record is produced at regular intervals and is an aggregate of the activities running inside a J2EE container during the interval. This record can be used to perform application profiling, problem determination, and capacity planning.

A single record is created for each active J2EE container located in a WebSphere Application Server transaction server within the interval being recorded. If there is more than one server instance associated with a server, a record for the container will exist for each server instance. To get a common view of the work running in the J2EE container during the interval, you must merge the records after processing.

You can specify an interval through the WebSphere Application Server administrative console by setting server_SMF_interval_length=n, where n is the desired number of seconds.

You can activate this record by setting server_SMF_container_interval_enabled=1 on the administrative console.

WebContainer activity record

This file describes the WebContainer activity record

The purpose of the WebContainer activity SMF record is to record activity within a WebContainer running inside a WebSphere Application Server for z/OS transaction server.

The Web container is deployed within an EJB and runs within the EJB container. The WebContainer acts as a Web server handling HTTP sessions and servlets. The EJB container is not aware of the work the WebContainer does. Instead, the EJB container only records that the EJB has been dispatched. Meanwhile, the WebContainer gathers the detailed information, such as HTTP sessions, servlets, and their respective performance data. A single WebContainer Activity record is created for each activity that is run within a Web container.

WebContainer SMF recording is activated and deactivated along with the activation and deactivation of SMF recording for the J2EE container.

WebContainer interval record

This file describes the WebContainer interval record

The purpose of the WebContainer interval SMF record is to record activity within a WebContainer running inside a WebSphere Application Server for z/OS transaction server.

The Web container execution environment consists of an EJB that is deployed into the EJB container. The WebContainer acts as a Web Server handling HttpSessions and Servlets. The EJB container is not aware of the purpose of the WebContainer activity record and only records that the EJB has been dispatched, but does not gather any of the detailed information, such as HttpSessions, Servlets, and their respective performance data. A single WebContainer record is created for each Web container.

In addition to data that is associated with an individual activity, there are some cases of Web container work that are performed outside the scope of an individual request. For example, some instances of http session finalization and http session invalidation are performed asynchronously. In such a case a WebContainer interval record would record this data

WebContainer SMF recording is activated and deactivated along with the activation and deactivation of SMF recording for the J2EE container.

Enabling SMF recording

Perform the following steps to enable SMF recording for WebSphere Application Server:

1. Use the WebSphere Application Server administrative console to enable properties for specific record types.
2. Edit the SMFPRMxx parmlib member.
3. Use the SET command to indicate which SMF parmlib member the system should use.
4. Format the output data set.

SMF has been enabled successfully when the SMF data is recorded in the data set which is specified in SMFPRMxx.

Using the WebSphere Application Server administrative console to enable properties for specific SMF record types

Ensure that you have proper access to the administrative console.

To enable SMF you must first use the WebSphere Application Server administrative console to enable properties for specific record types. To view or set properties using the WebSphere Application Server administrative console:

1. Click **Server>Application Servers** in the navigation tree. The Application Servers page appears.
2. Click the **application server name** in the **Name** column of the Application Server collection table. The configuration panel of the application server selected appears.
3. On the configuration panel, under the Additional Properties section, click on **Process Definition**.
4. Click on **control** in the **process Type** column.
5. On the configuration panel, under the Additional Properties section, click on **Environment Entries**.
6. To enable specific record types, specify one or more of the following properties:
   - server_SMF_server_activity_enabled=1
   - server_SMF_server_interval_enabled=1
   - server_SMF_container_activity_enabled=1
   - server_SMF_container_interval_enabled=1
   - server_SMF_interval_length=n, where n is the interval, in seconds, that the system will use to write records for a server instance.
7. Click **OK** or **Apply**.
8. Save the changes and make sure a file sync is performed before restarting the servers.
9. For the changes to take effect, restart the application server.

The steps are completed when the record types are successfully activated.

Editing the SMFPRMxx parmlib member

Make a working copy of the sample PARMLIB member SMFPRMYL.

Follow these steps to edit the SMFPRMxx parmlib member and enable SMF recording for WebSphere Application Server:

1. Insert an ACTIVE statement to indicate SMF recording. See *z/OS MVS Initialization and Tuning Guide* for more information.
2. Insert a SYS statement to indicate the types of SMF records you want the system to create. For example, use SYS(TYPE(120:120)) to select WebSphere Application Server type 120 records only. Keep the number of selected record types small to minimize the performance impact.
3. You can specify the interval in which you want the Server and Container interval records created (if no interval was specified in administrative console for the server or container definition) using the INTVAL(mm) statement in the SMFPRMxx parmlib member . The default SMF recording interval is 30 minutes. See *z/OS MVS Initialization and Tuning Reference* for more information.

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

Writing records to DASD

Make sure you have your modified PARMLIB member SMFPRMxx.

Follow this step to start writing records to DASD:
Issue the following command: t smf=xx where xx is the suffix of the SMF parmlib member (SMFPRMxx). See *z/OS MVS System Management Facilities (SMF), SA22-7630* for more information.

Writing records to DASD has been completed successfully when the data is recorded in the data set which is specified in SMFPRMxx.

Formatting the output data set

Make sure SMF recording is running.

Perform the following steps to format the SMF recording output data set into a readable format for printing to the screen or other output device:
1. Switch the SMF data sets by entering i smf from the MVS console to switch the SMF data sets.
2. Run the SMF Dump program (IFASMFDP) to create a sequential data set from the raw dump. A sample JCL is shown in *z/OS MVS System Management Facilities (SMF), SA22-7630*.

You have successfully formatted the output data set when SMFDUMP ends with return code 0.

Disabling SMF recording for WebSphere Application Server

Ensure that you have proper access to the administrative console.

SMF recording can be enabled for WebSphere Application Server, and for z/OS and OS/390. The following steps describe how to disable SMF recording for WebSphere Application Server:
1. Disabling SMF recording can be achieved by the steps outlined in Using the WebSphere Application Server administrative console to enable properties for specific record types, and by deselecting the records that have been enabled.
2. When you have finished making the administrative console entries, recycle the server.

The step for disabling SMF recording for WebSphere Application Server has been successfully completed when SMF records of records type 120 are no longer being recorded.

Disabling SMF recording for the entire MVS system

Make sure that you have your own working copy of SMFPRMxx and SMF is running.

SMF recording can be enabled for WebSphere Application Server, and for z/OS and OS/390. The following steps describe how to disable SMF recording for your MVS System (z/OS and OS/390):
Edit the SMFPRMxx parmlib member and set SMFPRMxx to ″NOACTIVE″ which will disable the writing of SMF records to DASD. Use the SET command to activate that SMF parmlib member on the MVS system.

SMF recording has successfully been disabled for the whole MVS system when SMF records for for both z/OS and OS/390 and WebSphere Application Server are no longer being written to DASD.

Overview of SMF record type 120

This file gives an overview of SMF record type 120

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

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

Viewing records with the SMF Browser

The SMF Browser available on the WebSphere for z/OS download site is able to display record type 120. To download the SMF Browser go to: http://www6.software.ibm.com/dl/websphere20/zosos390-p. For further information on the SMF Browser, download the browser package and read the associated documentation.

Record Type 120 (78) - WebSphere for z/OS performance statistics

The following section defines the SMF Record Type 120 (78) - WebSphere for z/OS performance statistics.

Record Type 120 (78) - WebSphere for z/OS performance statistics

WebSphere for z/OS writes record type 120 to collect WebSphere for z/OS performance statistics. For more information about SMF record types, see *z/OS MVS System Management Facilities (SMF)*.

All subtypes of the record type 120 have the following format:
- Standard header section
- Individual header extension for subtype x
- Product section
- Subtype-specific sections listed below.

Record type 120 has the following subtypes:
- **Subtype 1: Server activity record**
  - **Server activity section** (one section per record):

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

    Contains information about each communication session.
  - **JVM heap section** (zero, one, or multiple sections per record):

    Contains information about the heap in a server region.
- **Subtype 3: Server interval record**
  - **Server interval section** (one section per record):

    Contains aggregated information about all activities that occurred within the specified server interval.
  - **Server region section** (zero, one, or multiple sections per record):

    Contains information about server regions in the specified interval.
- **Subtype 5: J2EE container activity record**
  - **J2EE container activity section** (one section per record):

    Contains information about each activity that occurred within one J2EE container.
  - **Bean section** (multiple (0..n) sections per record):

    Contains information about all beans involved in this activity.
  - **Bean method section** (multiple (0..n) sections per bean section):

    Contains information about all methods of this bean involved in this activity.
- **Subtype 6: J2EE container interval record**
  - **J2EE container interval section** (one section per record):

    Contains aggregated information about all activities that occurred within one J2EE container in the specified interval.
  - **Bean section** (multiple (0..n) sections per record, see subtype 5):

    Contains information about all beans involved in this activity in the specified interval.
  - **Bean method section** (multiple (0..n) sections per bean section, see subtype 5):

    Contains information about all methods of this class involved in this activity in the specified interval.
- **Subtype 7: WebContainer activity record (Version 2)**
  - **WebContainer activity section** (one section per record):

    Contains information about each activity that occurred within one WebContainer.
  - **HttpSessionManager activity section** (one section per record):

    Contains information about all sessions involved in this activity.
  - **WebApplication section** (multiple (0..n) sections per record):

    Contains information about all WebApplications involved in this activity.
  - **Servlet section** (multiple (0..n) sections per WebApplication section):

    Contains information about all Servlets involved in this activity.
- **Subtype 8: WebContainer interval record (Version 2)**
  - **WebContainer interval section** (one section per record):

    Contains information about each activity that occurred within one WebContainer.
  - **HttpSessionManager section** (one section per record):

    Contains information about all sessions involved in this activity.
  - **WebApplication section** (multiple (0..n) sections per record):

    Contains information about all WebApplications involved in this activity.
  - **Servlet section** (multiple (0..n) sections per WebApplication section):

    Contains information about all Servlets involved in this activity.

Triplets:

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

You can use triplets to build self-describing SMF records that contain various types of data sections and a varying number of each of these sections. All data sections are described by triplets that consist of:
1. An offset that specifies the start position of the data
2. A length that describes the length of the section
3. A count that describes how many instances of the section are included in this record.

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

Splitting SMF records:

This section describes splitting SMF records.

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

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

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

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

Record environment and mapping:

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

Record environment

The following conditions exist for the generation of this record:

- **Macro** SMFWTM (record exit: IEFU83)
  Mode Task
  Storage Residency
  31-bit

Record mapping

For a description of the common SMF record header fields and the triplet fields (offset/length/number), if applicable, that locate the other sections on the record, see Header/Self-defining section.

For a description of triplets, see Using Triplets and MVS System Management Facilities (SMF)″ (SA22-7630).

Header/self-defining section:

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

Header/Self-defining section

The tables below describe the header/self-defining section of an SMF record.

Offset (decimal)	Offset (hexadecimal)	Name	Length	Format	Description
0	0	SM120LEN	2	binary	Record length. This field and the next field (total of four bytes) form the RDW (record descriptor word). See ″Standard SMF record header″ in MVS System Management Facilities (SMF)″ (SA22-7630), for a detailed description.
2	2	SM120SEG	2	binary	Segment descriptor (see record length field)

4	4	SM120FLG	1	binary	**Bit meaning when set** 0: New SMF record format 1: Subtypes used 2: Reserved 3-6: Version indicators* 7: Reserved *See ″Standard SMF record header″ in MVS System Management Facilities (SMF)″ (SA22-7630), for a detailed description.
5	5	SM120RTY	1	binary	Record type 120(X'78')
6	6	SM120TME	4	binary	Time since midnight, in hundredths of a second, that the record was moved into the SMF buffer.
10	A	SM120DTE	4	packed	Date when the record was moved into the SMF buffer, in the form 0*cyydddF*. See ″Standard SMF record header″ in MVS System Management Facilities (SMF)″ (SA22-7630), for a detailed description.
14	E	SM120SID	4	EBCDIC	System identification (from the SMFPRMxx SID parameter)
18	12	SM120SSI	4	EBCDIC	Subsystem identification from SUBSYS parameter
22	16	SM120RST	2	binary	Record subtype: 1: Server activity 2: Container activity 3: Server interval 4: Container interval. 5: J2EE container activity 6: J2EE container interval 7: WebContainer activity 8: WebContainer interval
24	18	SM120TRN	4	binary	Number of triplets in this record. A triplet is a set of three SMF fields (offset/length/number values) that defines a section of the record. The offset is the offset from the RDW. Subtypes: 1: Value is equal to the number of sessions +2 2 and 4: Value is equal to the number of classes +2.
28	1C	SM120PRS	4	binary	Offset to product section from RDW.
32	20	SM120PRL	4	binary	Length of product section.
36	24	SM120PRN	4	binary	Number of product sections.

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

44	2C	SM120SAL	4	binary	Length of server activity section
48	30	SM120SAN	4	binary	Number of server activity sections
52	34	SM120CSS	4	binary	Offset to communication session section from RDW
56	38	SM120CSL	4	binary	Length of communication session section
60	3C	SM120CSN	4	binary	Number of communication session sections
64	40	SM120JHS	4	binary	Offset to JVM heap section from RDW
68	44	SM120JHL	4	binary	Length of JVM heap section
72	48	SM120JHN	4	binary	Number of jvm heap sections

Individual header extension for subtype 3					
40	28	SM120SIS	4	binary	Offset to server interval section from RDW
44	2C	SM120SIL	4	binary	Length of server interval section
48	30	SM120SIN	4	binary	Number of server interval sections
The following triplet appears 0-n times; once for each server region section.					
52	34	SM120SRS	4	binary	Offset to server region section from RDW
56	38	SM120SRL	4	binary	Length of server region section
60	3C	SM120SRN	4	binary	Number of server region sections

Individual header extension for subtype 5					
40	28	SM120JA1	4	binary	Offset to J2EE container activity section from RDW
44	2C	SM120JA2	4	binary	Length of J2EE container activity section
48	30	SM120JA3	4	binary	Number of J2EE container activity sections

The following triplet appears 0-n times; once for each bean section.					
52	34	SM120JAS	4	binary	Offset to bean section from RDW
56	38	SM120JAL	4	binary	Length of bean section
60	3C	SM120JAN	4	binary	Number of bean sections

Individual header extension for subtype 6					
40	28	SM120JI1	4	binary	Offset to J2EE container interval section from RDW
44	2C	SM120JI2	4	binary	Length of J2EE container interval section
48	30	SM120JI3	4	binary	Number of J2EE container interval sections

The following triplet appears 0-n times; once for each bean section.					
52	34	SM120JIS	4	binary	Offset to bean section from RDW
56	38	SM120JIL	4	binary	Length of bean section
60	3C	SM120JIN	4	binary	Number of bean sections

Individual header extension for subtype 7					
40	28	SM120WA1	4	binary	Offset to WebContainer activity section from RDW.
44	2C	SM120WA2	4	binary	Length of WebContainer activity section.
48	30	SM120WA3	4	binary	Number of WebContainer activity sections.
52	34	SM120WA4	4	binary	Offset to HttpSessionManager activity section from RDW.
56	38	SM120WA5	4	binary	Length of HttpSessionManager activity section.
60	3C	SM120WA6	4	binary	Number of HttpSessionManager activity sections.
The following triplet appears 0-n times, once for each WebApplication section.					
64	40	SM120WA7	4	binary	Offset to WebApplication section from RDW.
68	44	SM120WA8	4	binary	Length of WebApplication section.
72	48	SM120WA9	4	binary	Number of WebApplication sections.
Individual header extension for subtype 8					
40	28	SM120WI1	4	binary	Offset to WebContainer interval section from RDW.
44	2C	SM120WI2	4	binary	Length of WebContainer interval section.
48	30	SM120WI3	4	binary	Number of WebContainer interval sections.
52	34	SM120WI4	4	binary	Offset to HttpSessionManager interval section from RDW.
56	38	SM120WI5	4	binary	Length of HttpSessionManager interval section.
60	3C	SM120WI6	4	binary	Number of HttpSessionManager interval sections.
The following triplet appears 0-n times, once for each WebApplication section.					
64	40	SM120WI7	4	binary	Offset to WebApplication section from RDW.
68	44	SM120WI8	4	binary	Length of WebApplication section.
72	48	SM120WI9	4	binary	Number of WebApplication sections.

Product section:

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

Product section

Offset	Offset	Name	Length	Format	Description
0	0	SM120MFV	4	binary	CB SMF version
4	4	SM120COD	8	EBCDIC	Character codeset in which strings in the SMF record are encoded
12	C	SM120END	4	binary	Encode of numbers in the SMF record

16	10	SM120TSF	4	binary	Encoding of timestamps: 1: S390STCK64: The time values are encoded in 64-bit S/390 Store Clock format.

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

Subtype 1: Server activity record:

This section includes Subtype 1: Server activity record.

Server activity section

The Server activity section contains information about each activity that occurred within one server.

Offset (decimal)	Offset (hexadecimal)	Name	Length	Format	Description
0	0	SM120HNM	64	EBCDIC	WebSphere for z/OS transaction server host name
64	40	SM120SNA	8	EBCDIC	WebSphere for z/OS transaction server name
72	48	SM120INA	8	EBCDIC	WebSphere for z/OS transaction server instance name
80	50	SM120SNM	4	binary	Total number of server regions that were involved to process this activity. If applicable, up to the first five server region address space IDs are listed within the next five fields.
84	54	SM120SR1	4	binary	The specific WebSphere for z/OS transaction server instance server region where the request ran
88	58	SM120SR2	4	binary	The specific WebSphere for z/OS transaction server instance server region where the request ran
92	5C	SM120SR3	4	binary	The specific WebSphere for z/OS transaction server instance server region where the request ran
96	60	SM120SR4	4	binary	The specific WebSphere for z/OS transaction server instance server region where the request ran
100	64	SM120SR5	4	binary	The specific WebSphere for z/OS transaction server instance server region where the request ran
104	68	SM120CRE	8	EBCDIC	The user credentials under which the activity began.
112	70	SM120ATY	4	binary	Type of activity that this record references: 1: Method request: This record refers to a method request that is not part of a global transaction. 2: Transaction: This record refers to a transaction.
116	74	SM120AID	20	HEX	Identity of the activity
136	88	SM120WLM	8	HEX	WLM enclave token
144	90	SM120AST	16	S390STCK	Activity start time
160	A0	SM120AET	16	S390STCK	Activity stop time
176	B0	SM120NIM	4	binary	Number of input methods

180	B4	SM120NGT	4	binary	Number of global transactions that were started in the server region
184	B8	SM120NLT	4	binary	Number of local transactions that were started in the server region
188	BC	SM120J2E	4	binary	J2EE server
192	C0	SM120CEL	8	EBCDIC	WebSphere for z/OS cell name
200	C8	SM120NOD	8	EBCDIC	WebSphere for z/OS node name
208	D0	SM120WCP	8	binary	Total CPU time accumulated by the WLM enclave. TOD clock format (bit 51 = microseconds).

Communications session section

There are zero, one, or multiple sections per record. The Communications session section contains information about each communication session.

Offset (decimal)	Offset (hexadecimal)	Name	Length	Format	Description
0	0	SM120CSH	8	HEX	Communications session handle
8	8	SM120CSA	64	EBCDIC	Communications session address
72	48	SM120CSO	4	binary	Communications session optimization 1: Local communications session: The session is a local OS/390 optimized communications session. 2: Remote communications session: The session is a remote communications session. 3: Remote encrypted (SSL) 4: Remote within sysplex. 5: HTTP session. 6: HTTP encrypted session.
76	4C	SM120SDR	4	binary	Data received; the number of bytes received by the server
80	50	SM120SDT	4	binary	Data transferred; the number of bytes transferred from the server back to the client.

JVM Heap section

There are zero, one, or multiple sections per record. The JVM heap section contains information about the heap in each server region.

Offset (decimal)	Offset (hexadecimal)	Name	Length	Format	Description
0	0	SM120JHA	4	binary	Servant address space ID
4	4	SM120JHH	4	binary	The heap for which the following data applies.
8	8	SM120JHC	4	binary	The total number of allocation failures on this heap or, if querying shared storage, the subpool identifier. A negative value indicates the information is for the shared memory page pool.
12	C	SM120JHF	8	binary	The total number of free bytes in the heap/subpool/page pool.
20	14	SM120JHT	8	binary	The total number of bytes in the heap/subpool/page pool.

Subtype 3: Server interval record:

This section includes Subtype 3: Server interval record

Server interval section

The server interval section contains information about each activity that occurred within one server.

Offset (decimal)	Offset (hexadecimal)	Name	Length	Format	Description
0	0	SM120HN2	64	EBCDIC	WebSphere for z/OS transaction server host name
64	40	SM120SNI	8	EBCDIC	WebSphere for z/OS transaction server name
72	48	SM120INI	8	EBCDIC	WebSphere for z/OS transaction server instance name
80	50	SM120SST	16	S390STCK	Time that the sample began in the server
96	60	SM120SET	16	S390STCK	Time that the sample ended
112	70	SM120NG2	4	binary	Number of global transactions that have run through the server instance during the interval that have been initiated by the server instance during the interval
116	74	SM120NL2	4	binary	Number of local transactions that have been initiated by the server instance during the interval
120	78	SM120NCS	4	binary	Number of communications sessions that exist at the end of the interval
124	7C	SM120NCA	4	binary	The number of communications sessions that have been active during the interval
128	80	SM120NLS	4	binary	Number of local communication sessions that exist at the end of the interval
132	84	SM120NLA	4	binary	Number of active local communication sessions that have been attached and active within the server instance during the interval
136	88	SM120NRS	4	binary	Number of remote communication sessions that exist at the end of the interval
140	8C	SM120NRA	4	binary	Number of active remote communication sessions that have been attached and active within the server instance during the interval
144	90	SM120BTS	4	binary	Number of bytes that have been transferred to the server from all attached clients
148	94	SM120BFS	4	binary	Number of bytes that have been sent from the server to all attached clients
152	98	SM120BTL	4	binary	Number of bytes that have been transferred to the server from all locally attached clients
156	9C	SM120BFL	4	binary	Number of bytes that have been transferred from the server to all locally attached clients
160	A0	SM120BTR	4	binary	Number of bytes that have been transferred to the server from all remotely attached clients
164	A4	SM120BFR	4	binary	Number of bytes that have been transferred from the server to all remotely attached clients
168	A8	SM120J2	4	binary	J2EE server.
172	AC	SM120CL1	8	EBCDIC	WebSphere for z/OS transaction server cell name
180	B4	SM120ND1	8	EBCDIC	WebSphere for z/OS transaction server node name
188	BC	SM120NHS	4	binary	Number of HTTP communication sessions that exist at the end of the interval

192	C0	SM120NHA	4	binary	Number of HTTP communication sessions that have been attached and active within the server instance during the interval
196	C4	SM120BTH	4	binary	Number of bytes that have been transferred to the server from all HTTP attached clients
200	C8	SM120BFH	4	binary	Number of bytes that have been transferred from the server to all HTTP attached clients
204	CC	SM120TEC	8	binary	Total CPU time accumulated by the WLM enclaves. TOD clock format (bit 51 = microseconds).

Server region section

There are zero, one, or multiple sections per record. The server regions section contains information about each server region in the specified server interval.

Offset	Offset	Name	Length	Format	Description
0	0	SM120SSA	4	binary	Servant address space ID
4	4	SM120SNT	4	binary	Number of triplets.
The following triplet appears 0-n times; once for each heap id section.					
8	8	SM120SSO	4	binary	Offset to heap id section from the beginning of this server region section.
12	C	SM120SSL	4	binary	Length of heap id section.
16	10	SM120SSN	4	binary	Number of heap id sections.

Subtype 3: Heap id section

There are multiple (0..n) sections per server region section. The Heap id section contains information about all heaps of this server region involved in this activity

Offset	Offset	Name	Length	Format	Description
0	0	SM120HIH	4	binary	The heap for which the following data applies.
4	4	SM120HIC	4	binary	Number of allocation failures on this heap during the interval.
8	8	SM120HI1	8	binary	Minimum number of bytes during the interval.
16	10	SM120HI2	8	binary	Maximum number of bytes during the interval.
24	18	SM120HI3	8	binary	Average number of bytes during the interval.
32	20	SM120HI4	8	binary	Minimum number of free bytes during the interval.
40	28	SM120HI5	8	binary	Maximum number of free bytes during the interval.
48	30	SM120HI6	8	binary	Average number of free bytes during the interval.

Subtype 5: J2EE container activity record (Version 2):

This section includes Subtype 5: J2EE container activity record (Version 2)

J2EE container activity section

There is one section per record. The J2EE container activity section contains information about each activity that occurred within one J2EE container.

Offset	Offset	Name	Length	Format	Description
0	0	SM120JA4	64	EBCDIC	WebSphere for z/OS transaction server host name
64	40	SM120JA5	8	EBCDIC	WebSphere for z/OS transaction server name
72	48	SM120JA6	8	EBCDIC	WebSphere for z/OS transaction server instance name
80	50	SM120JA7	4	binary	The specific WebSphere for z/OS transaction server instance server region where the request ran
84	54	SM120JA8	512	Unicode	WebSphere for z/OS container name.
596	254	SM120JA9	8	HEX	The WLM enclave token
604	25C	SM120JAA	4	binary	RESERVED
608	260	SM120JAB	20	HEX	The identity of the activity
628	274	SM120CL2	8	EBCDIC	Cell
636	27C	SM120ND2	8	EBCDIC	Node

Bean section

There are multiple sections per record. The bean section contains information about all beans involved in this activity.

Offset	Offset	Name	Length	Format	Description
0	0	SM120JB1	512	Unicode	AMCName of the bean activated by the container. **Note:** If the length of the AMCName exceeds 256 DBCS characters (512 bytes), the rightmost 256 characters are recorded.
512	200	SM120JB2	60	binary	UUID based AMC name
572	23C	SM120JB3	4	binary	The bean’s type. 0: CMP entity bean. 1: BMP entity bean. 2: Stateless session bean. 3: Stateful session bean.
576	240	SM120JB4	4	binary	RESERVED
580	244	SM120JB5	4	binary	RESERVED
584	248	SM120JB6	4	binary	RESERVED
588	24C	SM120JB7	4	binary	The bean’s reentrance policy. 0: Not reentrant within transaction. 1: Reentrant within transaction.
592	250	SM120JB8	4	binary	RESERVED
596	254	SM120JMC	4	binary	RESERVED
600	258	SM120JM6	4	binary	RESERVED
604	25C	SM120JB9	4	binary	Number of method triplets in this bean section
The following triplet appears 0-n times; once for each bean method section.					
608	260	SM120JBS	4	binary	Offset to bean method section from the beginning of this bean section
612	264	SM120JBL	4	binary	Length of bean method section
616	268	SM120JBN	4	binary	Number of bean method sections

Bean method section

There are multiple sections per bean section. The bean method section contains information about all methods of beans involved in this activity.

Offset	Offset	Name	Length	Format	Description
0	0	SM120JM1	1,024	Unicode	The name of the method including its signature in its externalized, human-readable form. If the length of the method exceeds 512 DBCS characters (1024 bytes), the leftmost 512 characters are recorded.
1024	400	SM120JM2	4	binary	The number of times the method was invoked during the activity.
1028	404	SM120JM3	4	binary	Average response time. The response time is measured in milliseconds (the granularity provided by the JVM - hopefully, it will be equal to 0 in most cases).
1032	408	SM120JM4	4	binary	Maximum response time. The response time is measured in milliseconds.
1036	40C	SM120JM5	4	binary	The bean method's transaction policy. Values from com.ibm.WebSphere for z/OS.csi. TransactionAttribute.java: `0: "TX_NOT_SUPPORTED"` `1: "TX_BEAN_MANAGED"` `2: "TX_REQUIRED"` `3: "TX_SUPPORTS"` `4: "TX_REQUIRES_NEW"` `5: "TX_MANDATORY"` `6: "TX_NEVER"`
1040	410	SM120JM8	4	binary	RESERVED.
1044	414	SM120JM9	4	binary	RESERVED.
1048	418	SM120JMA	512	Unicode	List of ejbRoles associated with the method. Separator character: ";" (semicolon). If the length of the concatenated string exceeds 256 characters (512 bytes), only its leftmost 256 characters are recorded.
1560	618	SM120JMB	4	binary	RESERVED.
1564	61C	SM120JMD	4	binary	RESERVED.
1568	620	SM120JME	4	binary	ejbLoad: # of invocations
1572	624	SM120JMF	4	binary	ejbLoad: avg execution time
1576	628	SM120JMG	4	binary	ejbLoad: max execution time
1580	62C	SM120JMH	4	binary	ejbStore: # of invocations
1584	630	SM120JMI	4	binary	ejbStore: avg execution time
1588	634	SM120JMJ	4	binary	ejbStore: max execution time
1592	638	SM120JMK	4	binary	ejbActivate: # of invocations
1596	63C	SM120JML	4	binary	ejbActivate: avg execution time
1600	640	SM120JMM	4	binary	ejbActivate: max execution time
1604	644	SM120JMN	4	binary	ejbPassivate: # of invocations
1608	648	SM120JMO	4	binary	ejbPassivate: avg execution time
1612	64C	SM120JMP	4	binary	ejbPassivate: max execution time
1616	650	SM120JMQ	8	binary	Average cpu time in microseconds.
1624	658	SM120JMR	8	binary	Minimum cpu time in microseconds.
1632	660	SM120JMS	8	binary	Maximum cpu time in microseconds.

Subtype 6: Container interval record (Version 2):

This section includes Subtype 6: Container interval record (Version 2)

J2EE container interval section

There is one section per record. The J2EE container interval section contains information about each activity that occurred within one J2EE container in the specified interval.

Offset (decimal)	Offset (hexadecimal)	Name	Length	Format	Description
0	0	SM120JI4	64	EBCDIC	The WebSphere for z/OS transaction server host name.
64	40	SM120JI5	8	EBCDIC	The WebSphere for z/OS transaction server name.
72	48	SM120JI6	8	EBCDIC	The WebSphere for z/OS transaction server instance name.
80	50	SM120JI7	512	Unicode	The WebSphere for z/OS container name. **Note:** This is hardcoded to ″Default″ for the 4.0.1 timeframe.
592	250	SM120JI8	16	S390STCK	The time that the sample began in the server.
608	260	SM120JI9	16	S390STCK	The time that the sample ended.
624	270	SM120CL3	8	EBCDIC	Cell
632	278	SM120ND3	8	EBCDIC	Node

Subtype 6: Bean section:

See Subtype 5: Bean section

Subtype 6: Bean method section:

See Subtype 5: Bean method section

Subtype 7: WebContainer activity record (Version 2):

This section includes Subtype 7: WebContainer activity record (Version 2)

WebContainer activity section

There is one section per record. The WebContainer activity section contains information about each activity that occurred within one web container.

Offset (decimal)	Offset (hexadecimal)	Name	Length	Format	Description
0	0	SM120WAA	64	EBCDIC	The WebSphere transaction server host name.
64	40	SM120WAB	8	EBCDIC	The WebSphere transaction server name.
72	48	SM120WAC	8	EBCDIC	The WebSphere transaction server instance name.
80	50	SM120WAD	8	HEX	The WLM enclave token.
88	58	SM120WAE	20	HEX	The identity of the activity.
108	6C	SM120WAF	16	S390STCK	The time the activity began in the server.
124	7C	SM120WAG	16	S390STCK	The time the activity ended.

140	8C	SM120CL4	8	EBCDIC	Cell
148	94	SM120ND4	8	EBCDIC	Node

HttpSessionManager section

There is one section per record. The HttpSessionManager section contains information about all (there may be zero or one) http sessions associated to one single activity.

Offset	Offset	Name	Length	Format	Description
0	0	SM120WAH	4	binary	″created Sessions″: Number of http sessions that were created.
4	4	SM120WAI	4	binary	″invalidatedSessions″: Number of http session that were invalidated.
8	8	SM120WAJ	4	binary	″activeSessions″: Number of http sessions that were referenced during this activity.
12	C	SM120WAK	4	binary	″sessionLifeTime″: lifetime of the session in milliseconds. If ″invalidatedSessions″ > 0, this is the average lifetime (in milliseconds) of the invalidated http session.

WebApplication section

There are multiple (0-n) sections per record. The WebApplication section contains information about all WebApplications involved in this activity.

Offset	Offset	Name	Length	Format	Description
0	0	SM120WAL	256	Unicode	The name of the WebApplication.
256	100	SM120WAM	4	binary	Number of servlet triplets in this web application section.
The following triplet appears 0-n times, once for each servlet section.					
260	104	SM120WAN	4	binary	Offset to servlet section from the beginning of this WebApplication section.
264	108	SM120WAO	4	binary	Length of servlet section.
268	10C	SM120WAP	4	binary	Number of servlet sections.

Servlet activity section

There are multiple (0-n) sections per WebApplication section. The Servlet activity section contains information about each servlet associated with WebApplications involved in this activity.

Offset	Offset	Name	Length	Format	Description
0	0	SM120WAQ	256	Unicode	The name of the servlet.
256	100	SM120WAR	4	binary	″responseTime″: Response time in milliseconds.
260	104	SM120WAS	4	binary	″numErrors″: The number of errors that were encountered during the servlet execution.

264	108	SM120WAT	4	binary	″loaded″: 0: The servlet did not have to be loaded as a result of this request. 1: The servlet had to be loaded as the result of this request.

268	10C	SM120WAU	16	EBCDIC	″loadedSince″: Timestamp from System.currentTimeMillis() when the servlet was loaded, in HEX format.
					
					Sample: The data as it appears in the record has the format
					`e7ef7c577c`
					
					, which needs to be converted to a Java long:
					`996155348860`
					
					. The Java long digits can be converted to java.util.Date:
					`Thu Jul 26 15:49:08 GMT+02:00 2001`

284	11C	SM120CPU	8	binary	Cpu time in microseconds.

Subtype 8: WebContainer interval record (Version 2):

This section includes Subtype 8: WebContainer interval record (Version 2)

WebContainer interval section

There is one section per record. The WebContainer interval section contains information about each activity that occurred within one WebContainer record.

Offset (decimal)	Offset (hexadecimal)	Name	Length	Format	Description
0	0	SM120WIA	64	EBCDIC	The WebSphere transaction server host name.
64	40	SM120WIB	8	EBCDIC	The WebSphere transaction server name.
72	48	SM120WIC	8	EBCDIC	The WebSphere transaction server instance name.
80	50	SM120WID	16	S390STCK	The time the sample began.
96	60	SM120WIE	16	S390STCK	The time the sample ended.
112	70	SM120CL5	8	EBCDIC	Cell
120	78	SM120ND5	8	EBCDIC	Node

HttpSessionManager section

There is one section per record. The HttpSessionManager section contains information about all (there may be zero or one) http sessions associated to one single activity.

Offset	Offset	Name	Length	Format	Description
0	0	SM120WIF	4	binary	″createdSessions″: Number of http sessions that were created.
4	4	SM120WIG	4	binary	″invalidatedSessions″: Number of http sessions that were invalidated.
8	8	SM120WIH	4	binary	″activeSessions″: Current number of http sessions that are actively referenced in the server at the end of the interval.

12	C	SM120WII	4	binary	″minActiveSessions″: Minimum number of active http sessions during the interval..
16	10	SM120WIJ	4	binary	″maxActiveSessions″: Maximum number of active http sessions during the interval.
20	14	SM120WIK	4	binary	″sessionLifeTime″: Average lifetime (in milliseconds) of invalidated http sessions.
24	18	SM120WIL	4	binary	″sessionInvalidateTime″: Average time (in milliseconds) that was required to process the invalidation of http sessions.
28	1C	SM120WIM	4	binary	″finalizedSessions″: Number of sessions that were finalized.
32	20	SM120WIN	4	binary	″liveSessions″: Total number of http sessions being tracked by the server at the end of the interval. This includes both active and inactive sessions.
36	24	SM120WIO	4	binary	″minLiveSessions″: Minimum number of live http sessions during the interval.
40	28	SM120WIP	4	binary	″maxLiveSessions″: Maximum number of live http sessions during the interval.

WebApplication interval section

There are multiple (0-n) sections per record. The WebApplication interval section contains information about all WebApplications involved in this activity.

Offset	Offset	Name	Length	Format	Description
0	0	SM120WIQ	256	Unicode	The WebApplication name.
256	100	SM120WIR	4	binary	″numLoadedServlets″: Number of servlets that were loaded. **Note:** This value might differ from the number of servlet sections in this record since servlets might exist that have been inactive during the interval.
260	104	SM120WIS	4	binary	Number of servlet triplets in this web application section.
The following triplet appears 0-n times, once for each servlet section.					
264	108	SM120WIT	4	binary	Offset to servlet section from the beginning of this WebApplication section.
268	10C	SM120WIU	4	binary	Length of the servlet section.
272	110	SM120WIV	4	binary	Number of servlet section.

Servlet section

There are multiple (0-n) sections per WebApplication section. The Servlet activity section contains information about all servlets involved per WebApplication in this activity.

Offset	Offset	Name	Length	Format	Description
0	0	SM120WIW	256	Unicode	The servlet name.
256	100	SM120WIX	4	binary	″totalRequests″: Number of times the servlet service was requested during the interval.
260	104	SM120WIY	4	binary	″responseTime″: Average response time in milliseconds.
264	108	SM120WIZ	4	binary	″minResponseTime″: Minimum response time in milliseconds.

268	10C	SM120WJ1	4	binary	″maxResponseTime″: Maximum response time in milliseconds.
272	110	SM120WJ2	4	binary	″numErrors″: The number of errors that were encountered during servlet execution.
276	114	SM120WJ3	16	EBCDIC	″loadedSince″: Timestamp when the servlet was loaded. Sample: Fri May 25 08:42:25 EDT 2001
292	124	SM120WJ4	8	binary	Average cpu time in microseconds.
300	12C	SM120WJ5	8	binary	Minimum cpu time in microseconds.
308	134	SM120WJ6	8	binary	Maximum cpu time in microseconds.

Overview of SMF record type 80

This file gives an overview of SMF record type 80

As WebSphere Application Server becomes more capable of authentication and setting or changing the identity on a thread, so arises the need for the ability to audit these changes. Along with this also comes the need to audit the accompanying authorization requests made through EJBRoles checking, intending to produce audit records that include the original authenticated identity. This auditing in WebSphere Application Server is managed not through WebSphere Application Server itself, but through its External Security Manager (RACF or equivalent), where the SMF records are cut.

Preparing for audit support

In order to take advantage of auditing in WebSphere, you need to set up SMF and RACF and have both running.

1. Set up SMF for audit support. For information on setting up and starting SMF, see *z/OS MVS System Management Facilities (SMF), SA22-7630*
2. Enable auditing for the EJB Roles by setting the RACF AUDIT attribute. This will set up RACF for auditing in WebSphere Application Server. You can turn on auditing for the ADMIN and PAYROLL classes with the following command:
   ```
   RALTER EJBROLE (ADMIN,PAYROLL) AUDIT(ALL)
   ```
   - Alternately, you could modify the RACFROLE job to put the AUDIT information there.
   - For more information and additional parameters for the AUDIT attribute, see the *z/OS Security Server RACF Auditor’s Guide*.

Using audit support

This file gives an overview of how to use audit support.

Auditing is performed using SMF records issued by RACF or an equivalent External Security Manager. This means that SMF audit records are cut as part of the WebSphere use of SAF interfaces such as IRRSIA00 (to manage ACEEs) and the RACROUTE macro.

The table below lists the various security authentication mechanisms and the corresponding data that is written to each part of the ACEE X500NAME field (this data is also in the RACO and SMF records). The information under ″Service Name″ is the constant string that is included in the ″Issuer’s Distinguished Name″ field of X500NAME. The information under ″Authenticated Identity″ is the principal that is recorded in the ″Subject’s Distinguished Name″ field.

Table 3. Security authentication mechanisms and the corresponding data that is written to each part of the ACEE X500NAME field

Authentication mechanism	Service name	Authenticated identity

Table 3. Security authentication mechanisms and the corresponding data that is written to each part of the ACEE X500NAME field (continued)

Custom Registry	WebSphere Custom Registry	Custom registry principal name
Kerberos	WebSphere Kerberos	Kerberos principal, in the ″DCE″ format used for extracting the corresponding MVS userid using IRRSIM00 (/.../realm/principal)
RunAs Rolename	WebSphere Role Name	Role name
RunAs Server	WebSphere Server Credential	MVS userid
Trust Interceptor	WebSphere Authorized Login	MVS userid
RunAs Userid/Password	WebSphere Userid/Password	MVS Userid

In addition to tracking by MVS userid, events need to be traced to an originating userid. This is especially true for originating userids that are not MVS-based, such as EJB Roles, Kerberos principals, and Custom Registry principals.

Collecting performance diagnosis information

The following report options are listed here for information. IBM Service may request that you run one or more of these reports while assisting you with diagnosis. You do not need to collect this data unless it is requested by IBM Service.

- If you suspect that you are having throughput problems in a particular address space, for example by looking at some other real-time performance data, IBM Service may need to see a dump of one or more address spaces. This is done using the following parameters:
  ```
   JOBNAME=(<jobname list>)
   SDATA=(LSQA,PSA,SQA,SUM,SWA,TRT,WLM,CSA,RGN)
  ```
- If you suspect that the problem could be resulting from GRS latch or ENQ contention, check the RMF Enqueue Activity Report and enter the console command:
  ```
  D GRS,CONTENTION
  ```
  during a time period in which the performance problem is observed. SYS.BPX.A000.FSLIT.FILESYS.LSN represents HFS latches. Latch sets with a numeric suffix are file latches, specifically SYS.BPX.A000.FSLIT.FILESYS.LSN.01. If you detect file latch contention, the best way to determine the exact HFS file causing the problem is with an SVC dump, also collected during a time period in which contention occurred. You will need to dump one of the OMVS data spaces to get the file information.
  ```
  DUMP COMM=(description of problem)
  Reply to dump WTO, where serverproc is the name of your WebSphere Server
                     address space(s)
  JOBNAME=(OMVS,Serverproc),DSPNAME=('OMVS'.SYSZBPX1,'OMVS'.SYSZBPX2),
  SDATA=(CSA,GRSQ,LPA,NUC,PSA,RGN,SQA,TRT,SUM)
  ```
- Sometimes USS errors can cause performance problems. The USS Ctrace (SYSOMVS) MIN tracing option always records OMVS errors. You can take an SVC dump of the OMVS address space (as described in the previous bullet) and the data spaces and format the SYSOMVS CTRACE. Use IPCS options 7.2.1, suboption D, component SYSOMVS and the TALLY option (default is FULL). Look for trace events of errors in the TALLY report.
- To find delays in applications, collect application performance information
  – SMF 120 records.
  – Jinsight profile

Chapter 3. 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
Error log data	Viewing error log contents through the Log Browse Utility (BBORBLOG)
z/OS display command	Using the z/OS display command
Java minor codes	Converting Java minor codes
SYSPRINT	Viewing SYSPRINT output, STDOUT and STDERR streams in an HFS File
Message routing	Message routing

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.

Viewing SVC dumps

A 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, GA22-756 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.

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.

Steps for using the IPCS dialog to format CTRACE data

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. 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

Tips: 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.

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.

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.

Steps for using IPCS in batch mode to format CTRACE data

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.

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.

When your installation has modified the BLSCDDIR CLIST the steps outlined herein will 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 the 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.
   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 the 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.

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.

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=OM
//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=OM
//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:

1. Use ISPF option 6 to enter the proper command.

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

   the log-stream name is assumed to be `BBO.BOSSXXXX`
2. 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.
3. The WebSphere Application Server for z/OS provides an ISPF REXX EXEC named BBORBLOG, that allows you to browse the error log stream.
4. Save the 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.

Use the following information to determine viewing of the error log:
- By default, the macro formats the error records to fit a 3270 display.
- Timestamps are in Greenwich Mean Time (GMT) unless changed by setting WebSphere variable `ras_time_local` to 1.
- 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)

The browser takes two parameters:

Parameter	Description
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.

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.

Important: 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`

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

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

Component	Description
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)
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

- Each field is delimited by a blank.
- 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

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.

Viewing SYSPRINT output, STDOUT and STDERR streams in an HFS File

If you are familiar with UNIX or NT environments, you may be reluctant to use the facilities of SDSF (or IOF) to view the SYSPRINT output from your Application Server regions. In the past, you may have used a familiar editor (such as vi) in a TELNET session to view the STDOUT and STDERR streams. These STDOUT and STDERR streams have been redirected to SYSPRINT, and it is possible to redirect them to files in the HFS for viewing using an HFS editor.

The JCL below for an Application Server region uses this facility. A new SET statement has been added to point to the LOGPATH directory, and the SYSPRINT DD statement has been changed to point to a file in the LOGPATH/servername directory, in this case named: `was.log.d &LYYMMDD..t&LHHMMSS` The extra period (.) between the date and time variables is not a typographical error, but rather an idiosyncrasy of JCL syntax that is necessary to terminate the first variable. &LYYMMDD will be replaced with the local date in YYMMDD format and &LHHMMSS will be replaced by the local time in HHMMSS format. Using the local date and time ensures a unique file for each instance of the Application Server region that is started.

The PATHMODE subparameter sets the file mode to 775 and the PATHOPTS subparameter OWRONLY opens the file for WRITE access and the sub-parameter OCREAT indicates that if the file does not already exist, create it.

```
//TSTST1S PROC IWMSSNM='TSTST1S',PARMS='-ORBsrvname '
//  SET CBCONFIG='/WebSphere390/TSDCONF'
//  SET LOGPATH='/WebSphere390/logs'        <<----- Added to set path
//  SET RELPATH='controlinfo/envfile'
//WSDAS1S EXEC PGM=BBOSR,REGION=0M,TIME=NOLIMIT,
// PARM='/ &PARMS &IWMSSNM'
//BBOENV DD PATH='&CBCONFIG/&RELPATH/&SYSPLEX/&IWMSSNM/current.env'
//CEEDUMP  DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//*YSPRINT DD SYSOUT=*  Added the following 3 lines for SYSPRINT

//SYSPRINT DD PATHMODE=(SIRWXU,SIRWXG,SIROTH),          <<< 775
//            PATHOPTS=(OWRONLY,OCREAT),                <<<
//            PATH='&LOGPATH/TSTST1S/was.log.d&LYYMMDD..t&LHHMMSS'
```

In the previous example, if the file specified by the SYSPRINT DD statement is created on May 2, 2003 at 1:30:35 PM, the PATH parameter will resolve to: `/WebSphere390/logs/TSTST1S/was.log.d030503.t133035` which will be a unique file for this execution of this server instance.

Message routing

The following is a summary of how messages get routed in WebSphere Application Server Version 5:

- WTO messages issued by WebSphere Application Server during initialization are sent to hardcopy, but most can be routed to the data set specified by ras_default_msg_dd (see Setting Output destinations and characteristics).
- The Java ″Audit″ messages are also sent to hardcopy, but can be routed to the data set ras_hardcopy_msg_dd. (see Setting Output destinations and characteristics).
- Trace error, service, and fatal messages are sent to the error log specified by the ras_log_logstreamName. Otherwise, they go to CERR (SYSOUT). Some may also go to hardcopy. You must set the ras_log_logstreamName environment variable to the error log stream name. To set ras_log_logstreamName, on the administrative console select Environment -> Manage WebSphere Variables -> New.
- Early error messages will go to STDERR (SYSOUT) until WebSphere Application Server connects to the log stream. A WTO (BBOO0153I) is issued telling you how many messages went to STDERR before you connected to the log stream.
- Trace messages (such as BUFFER, SYSPRINT, and TRCFILE DD) are also routed to ras_trace_outputLocation.
- System.out.println, Tr.Sysout, stdout (cout), and client output go to SYSPRINT.(see Viewing SYSPRINT output, STDOUT and STDERR streams in an HFS File
- System.out.printerr, Tr.Syserr, and stderr (cerr) go to SYSOUT.

To use these new message routing variables, you must do two things:

1. Add these parameters to the server definitions using the Administrative Console under Environment -> Manage WebSphere Variables:
   - ras_default_msg_dd =DEFALTDD
   - ras_hardcopy_msg_dd =HRDCPYDD

   You can set these variables for individual control and servant processes, but it is easier to set them in the Environment variables for the entire cell. For the Daemon, you must prefix them with ″DAEMON_″ and set them at the cell level:
   - DAEMON_ras_default_msg_dd =DEFALTDD
   - DAEMON_ras_hardcopy_msg_dd =HRDCPYDD

Update the included &Z members in PROCLIB to add these new DD statements

```
//* Output DDs
//CEEDUMP DD SYSOUT=*,SPIN=UNALLOC,FREE=CLOSE
//SYSOUT DD SYSOUT=*,SPIN=UNALLOC,FREE=CLOSE
//SYSPRINT DD SYSOUT=*,SPIN=UNALLOC,FREE=CLOSE
//DEFALTDD DD SYSOUT=*,SPIN=UNALLOC,FREE=CLOSE
//HRDCPYDD DD SYSOUT=*,SPIN=UNALLOC,FREE=CLOSE
```

Note:

- If you specify the new environment variables, but do not specify the DD cards in the procedure, you will not get an error message indicating that the DD cards are missing, and the tracing output will not be written anywhere.
- If you try to direct the output for multiple streams to the same DD, such as setting both ras_default_msg_dd and ras_hardcopy_msg_dd to DEFALTDD (or to SYSPRINT,) then the allocation will fail and output will be sent to the default location (JOBLOG/SYSLOG).

For example, these DD files are used to segregate the messages and keep almost all of them off the hardcopy console

1. (SYSLOG.) JESMSGLG - a few start-up and shut-down messages
2. JESYSMSG - MVS allocation and deallocation messages
3. SYSOUT - a few start-up and shut-down messages
4. SYSPRINT - a few start-up and shut-down messages
5. HRDCPYDD - audit messages that would normally go to SYSLOG
6. DEFALTDD - informational messages that would normally go to SYSLOG

Using the Error Dump and Cleanup interface

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
```

Viewing the output data set

The data set should be viewed using a program that can display record type 120.

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

1. Verify that the *JAVA_HOME* environment variable refers to the current java installation. JAVA_HOME=../usr/bin/java/J1.3 This should be at least Java 1.3 since this release is the first to implicitly contain the necessary record support needed by the interpreter.
2. Download the SMF Record Interpreter from the WebSphere for z/OS web site at: http://www.ibm.com/software/webservers/appserv/zos_os390/index.html
3. Copy the file bbomsmfv.jar to your tools directory. Be sure that any edits made to the file in the future are made to both copies of the file, or just execute from the installation directory in the first place.
4. To interpret SMF data from a cataloged z/OS or OS/390 sequential file named ″USER.SMFDATA″ (which was previously created using the IFASMFDP utility as described above), execute: java -cp bbomsmfv.jar com.ibm.ws390.sm.smfview.Interpreter ″USER.SMFDATA″ It is implicit in the java command parameterization that your current working directory is the tools directory. If this is not the case, you will receive a NoClassDefFoundError on com.ibm.ws390.sm.smfview.Interpreter. Java doesn't generate a diagnostic when it doesn't find bbomsmfv.jar in the current directory.

The SMF ViewTool has been successfully installed and invoked when you do not receive any Java error messages after the invocation and the Browser output is shown on the screen.

Example of SMF Browser output

Example of SMF Browser output

The SMF Browser available on the WebSphere for z/OS download site is able to display record type 120. To download the SMF Browser go to: http://www6.software.ibm.com/dl/websphere20/zosos390-p. For further information on the SMF Browser, download the browser package and read the associated documentation.

The following example shows sample output from the SMF Browser. The example features subtype 7 and subtype 8, in that order.

```
Record#: 14;
 Type: 120; Size: 820; Date: Fri Nov 23 04:54:17  EST 2001;
 SystemID: SY1; SubsystemID: WAS; Flag: 94;
 Subtype: 7 (WEB CONTAINER ACTIVITY);
# Triplets: 4;
Triplet #: 1; offset: 76; length: 32; count: 1;
Triplet #: 2; offset: 108; length: 140; count: 1;
Triplet #: 3; offset: 264; length: 556; count: 1;
Triplet #: 1; Type: ProductSection;
  Version: 1; Codeset: Unicode; Endian: 1; TimeStampFormat: 1 (S390STCK64);
  IndexOfThisRecord: 1; Total # records: 1; Total # triplets: 4;
Triplet #: 2; Type: WebContainerActivitySection;
  HostName : PLEX1;
  ServerName : BBOASR4;
  ServerInstanceName: BBOASR4A;
  WlmEnclaveToken * 00000020 00000242 -------- --------
                  * ^... * p1047
  ActivityID * b6c7a7b7 14e9bc85 000000b0 00000007
             * 0926306b -------- -------- --------
             *,............ *Cp1047
             ActivityStartTime * b6c7a7b7 14e9bc85 40404040 40404040 *
             ActivityStopTime * b6c7a7b7 53a8a645 40404040 40404040 *
Triplet #: 3; Type: HttpSessionManagerActivitySection;
  # http sessions created: 0; # http sessions invalidated: 0;
  # http sessions active: 0;
Average session life time: 0 [sec*10**-3];
 Triplet #: 4; Type: WebApplicationActivitySection;
  Name: PolicyIVP-localhost_1;
  # Servlets: 1;
  Triplet #: 4.1; offset: 272; length: 284; count: 1;
  Triplet #: 4.1; Type: ServletActivitySection;
   Name: SimpleFileServlet;
   ResponseTime: 48 [sec*10**-3];
   # errors: 0;
   Loaded by this request: 0;
   Loaded since (raw): ea54948e0d;
   Loaded since: Thu Nov 22 10:02:49 EST 2001;
Record#: 72;
  Type: 120; Size: 1744; Date: Fri Nov 23 05:01:02 EST 2001;
  SystemID: SY1; SubsystemID: WAS; Flag: 94;
  Subtype: 8 (WEB CONTAINER INTERVAL);
  # Triplets: 4;
  Triplet #: 1; offset: 76; length: 32; count: 1;
  Triplet #: 2; offset: 108; length: 112; count: 1;
  Triplet #: 3; offset: 264; length: 1480; count: 1;
  Triplet #: 1; Type: ProductSection;
    Version: 1; Codeset: Unicode; Endian: 1; TimeStampFormat: 1 (S390STCK64);
    IndexOfThisRecord: 1; Total # records: 1; Total # triplets: 4;
  Triplet #: 2; Type: WebContainerIntervalSection;
    HostName : PLEX1;
    ServerName : BBOASR4;
    ServerInstanceName: BBOASR4A;
    SampleStartTime * b6c7a6fd 655c0604 40404040 40404040 *
    SampleStopTime * b6c7a939 9a0e614c 40404040 40404040 *
Triplet #: 3; Type: HttpSessionManagerIntervalSection;
    http sessions #created: 1; #invalidated: 0;
    http sessions #active: 0; Min #active: 0; Max #active: 0;
    Average session life time: 0;
    Average session invalidate time: 0;
    http sessions #finalized: 0; #tracked: 0;
    http sessions #min live: 0; #max live: 0;
  Triplet #: 4; Type: WebApplicationIntervalSection;
    Name: PolicyIVP-localhost_1;
    # Servlets loaded: 0;
    # Servlets: 4;
    Triplet #: 4.1; offset: 312; length: 292; count: 1;
    Triplet #: 4.2; offset: 604; length: 292; count: 1;
    Triplet #: 4.3; offset: 896; length: 292; count: 1;
    Triplet #: 4.4; offset: 1188; length: 292; count: 1;
    Triplet #: 4.1; Type: ServletIntervalSection;
      Name: SimpleFileServlet;
      # requests: 6;
      AverageResponseTime: 764 [sec*10**-3];
      MinimumResponseTime: 18 [sec*10**-3];
      MaximumResponseTime: 4133 [sec*10**-3];
      # errors: 0;
      Loaded since (raw): ea54948e0d;
      Loaded since: Thu Nov 22 10:02:49 EST 2001;
    Triplet #: 4.2; Type: ServletIntervalSection;
      Name: Was40Ivp;
      # requests: 4;
      AverageResponseTime: 4664 [sec*10**-3];
      MinimumResponseTime: 1584 [sec*10**-3];
      MaximumResponseTime: 12572 [sec*10**-3];
      # errors: 0;
      Loaded since (raw): ea58a1509e;
      Loaded since: Fri Nov 23 04:55:14 EST 2001;
    Triplet #: 4.3; Type: ServletIntervalSection;
      Name: /cebit.jsp;
      # requests: 1;
      AverageResponseTime: 204 [sec*10**-3];
      MinimumResponseTime: 204 [sec*10**-3];
      MaximumResponseTime: 204 [sec*10**-3];
      # errors: 0;
      Loaded since (raw): ea58a24a69;
      Loaded since: Fri Nov 23 04:56:18 EST 2001;
    Triplet #: 4.4; Type: ServletIntervalSection;
      Name: JSP 1.1 Processor;
      # requests: 1;
      AverageResponseTime: 482 [sec*10**-3];
      MinimumResponseTime: 482 [sec*10**-3];
      MaximumResponseTime: 482 [sec*10**-3];
      # errors: 0;
      Loaded since (raw): ea54948b66;
      Loaded since: Thu Nov 22 10:02:48 EST 2001;
```

Chapter 4. Automation and recovery scenarios and guidelines

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

APPC automation and recovery scenarios

Table 6. APPC automation and recovery scenarios

Task	APPC automation and recovery scenarios
Startup	APPC should be started before WebSphere Application Server for z/OS. In theory, WebSphere Application Server for z/OS *could* be started before APPC, but only as long as no objects get dispatched in containers that have an IMS APPC LRMI associated with them. If APPC is not up before WebSphere Application Server for z/OS, and you want to use an APPC connector to talk to IMS, then you will have no connectivity. APPC/MVS does not have to be up for CICS. APPC does not have to be started after VTAM.
Shutdown	Reverse the startup procedure. Shutdown WebSphere Application Server for z/OS, APPC, then VTAM.
Handling in-flight or indoubt transactions if there is a failure	If you are using APPC for communications and it fails, do the following: 1. Shutdown all servers with APPC connectivity. 2. Restart APPC (if it totally failed). 3. Restart the WebSphere Application Server for z/OS server. **Note:** APPC will resynchronize itself. If your transaction is indoubt, IMS waits until you restart APPC. IMS relies on RRS for recovery. RRS will resolve transactions that are in doubt by handshaking with every subsystem it was communicating with before it went down. If you are using CICS, note that CICS has its own coordinator.
How to determine if APPC is running	Issue the `DISPLAY APPC,LU,ALL` command. If APPC is not active, it will say so. In addition, the status of the logical units used by WebSphere Application Server for z/OS and/or IMS should be active or no APPC work will be successful.
What happens to WebSphere for z/OS if APPC goes down?	Any objects attempting to use the IMS APPC PAA will not work. The server region running on behalf of the container attempting to use APPC will likely get a `C9C24C05` error, indicating that an APPC ALLOCATE request was attempted and failed. Additional APPC error diagnostic information that helps to pinpoint the APPC problem is contained in the logs associated with this region.
What happens to other subsystems if APPC goes down?	Not applicable
Where to find more information	• *z/OS MVS Planning: Operations* • *z/OS MVS Planning: APPC/MVS Management* • *z/OS MVS Programming: Resource Recovery*

WLM automation and recovery scenarios

Table 7. Workload Manager (WLM) automation and recovery scenarios

Task	**WLM automation and recovery scenarios**
Startup	WLM is automatically started by z/OS or OS/390 when you IPL your system. You do not have to start it.
Shutdown	You cannot shutdown WLM.
Handling in-flight and indoubt transactions if there is a failure	Not applicable
How to determine if WLM is running	Not applicable
What happens to WebSphere for z/OS if WLM goes down?	Not applicable
What happens to other subsystems if WLM goes down?	Not applicable

How to handle a catastrophic failure of the WebSphere Application Server for z/OS server regions	Following a catastrophic failure of the WebSphere Application Server for z/OS server regions, you can use one of the following resume commands, depending on your application environment type:
	• Static application environment:
	`v wlm,applenv=`*`applenvname`*`,`
	`resume`
	• Dynamic application environment:
	`v wlm,applenv=`*`applenvname`*`,`
	`resume,`*`options`*

Where to find more information	• *z/OS MVS Planning: Workload Management* • *z/OS MVS Programming: Workload Management Services*

RACF automation and recovery scenarios

Table 8. RACF automation and recovery scenarios

Task	**RACF automation and recovery scenarios**
Startup	If it is installed, RACF is started as a part of IPL.
Shutdown	RACF is not shutdown.
Handling in-flight and indoubt transactions if there is a failure	Not applicable
How to determine if RACF is running	Use the RACF SETROPTS command to display the status of RACF.
What happens to WebSphere for z/OS if RACF goes down?	RACF goes into fail safe mode. This means that for every resource that is accessed, the operator is asked to verify if it is okay. In general, the system is IPLed if this occurs.
What happens to other subsystems if RACF goes down?	It depends on the subsystem and how RACF fails.
Where to find more information	• *z/OS Security Server RACF System Programmer's Guide* • *z/OS Security Server RACF Security Administrator's Guide*

RRS automation and recovery scenarios

Table 9. RRS automation and recovery scenarios

Task	**RRS automation and recovery scenarios**
Startup	Ensure System Logger has been started before RRS. **Note:** RRS will display error messages indicating that System Logger must be started first if you try to start RRS without starting System Logger.Ensure RRS is started before WebSphere for z/OS. RRS does not start by itself. RRS will start automatically only if it was registered with the Automatic Restart Manager (ARM) and if ARM is running. To start RRS, issue the `start` command: `start atrrrs,sub=master` **Note:** RRS doesn't restart itself if you issue the cancel command, so you need to restart it manually if it was canceled or if ARM isn't running.
Shutdown	Shutdown RRS in the reverse order that you started RRS. Shutdown WebSphere Application Server for z/OS, then RRS, followed by System Logger. There is no controlled way to bring down RRS. The best approach is: 1. Quiesce WebSphere Application Server for z/OS. 2. Shutdown WebSphere Application Server for z/OS. 3. Cancel RRS. **Note:** You may want to bring down the DB2 you are using for WebSphere Application Server for z/OS before canceling RRS. To cancel RRS, issue the command: `setrrs cancel`
Handling in-flight and indoubt transactions if there is a failure	Refer to the RRS system management panels to display in-flight and resolve indoubt transactions. You can display the resource managers on the RM panels in RRS, display all units of recovery (UR), filter the URs, and then resolve the indoubts. You cannot resolve in-flights. You can display all RRS-managed transactions. If you are using the IMS Connector for Java, this process applies only if IMS Connector for Java, IMS Connect, and the IMS subsystem have been configured locally on the same z/OS or OS/390 system image on which the WebSphere for z/OS J2EE server runs. The local configuration is the only configuration in which IMS Connector for Java runs as an RRS-transactional connector.
How to determine if RRS is running	Use the display command: `d a,atrrs` `atrrs` is the name of the default RRS procedure shipped with WebSphere Application Server for z/OS. Use the procedure name that you use to start RRS. The address space comes from the procedure.
What happens to WebSphere for z/OS if RRS goes down?	RRS is a required subsystem, so WebSphere Application Server for z/OS will not run without it. If RRS goes down, WebSphere Application Server for z/OS will get fatal errors. You need to get RRS started, then restart WebSphere Application Server for z/OS.
What happens to other subsystems if RRS goes down?	RRS is the z/OS or OS/390 transaction monitor. If you cancel RRS, you will have problems with any subsystems using it (for example, WebSphere Application Server for z/OS, DB2, IMS). You should understand the implications before you cancel RRS.
Where to find more information	*z/OS MVS Programming: Resource Recovery*

UNIX System Services automation and recovery scenarios

Table 10. UNIX System Services automation and recovery scenarios

Task	UNIX System Services automation and recovery scenarios
Startup	UNIX System Services is a permanent component of MVS and is started automatically at IPL time.
Shutdown	UNIX System Services does not support a shutdown capability, so it is always available.
Handling in-flight or indoubt transactions if there is a failure	The only data that could be considered transactional in nature is data stored in the HFS.
How to determine if UNIX System Services is running	UNIX System Services is always available as long as the system is up and running.
What happens to WebSphere for z/OS if UNIX System Services goes down?	If UNIX System Services fails, the system must be re-IPLed. WebSphere Application Server for z/OS will get errors and terminate.
What happens to other subsystems if UNIX System Services goes down?	If UNIX System Services fails, the system must be re-IPLed.
Where to find more information	*z/OS UNIX System Services Planning*

TCP/IP automation and recovery scenarios

Table 11. TCP/IP automation and recovery scenarios

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

DB2 automation and recovery scenarios

Table 12. DB2 automation and recovery scenarios

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

Table 12. DB2 automation and recovery scenarios (continued)

Task	DB2 automation and recovery scenarios
Handling in-flight or indoubt transactions if there is a failure	Use the RRS panels to resolve. See *z/OS MVS Programming: Resource Recovery*. The RRS panels are the preferred way to resolve DB2 indoubts because they allow you to view all resource managers that have an interest in the transaction. However, you can also use DB2 to resolve indoubts. You can issue the command: `DISPLAY THREAD(*) TYPE(INDOUBT)` to display DB2 information about the indoubt threads it knows about (if there are too many, you can go into S.LOG to view the information). This display will give you a DB2 identifier called a ″nid″. Copy the nid and paste it into this command: `-RECOVER INDOUBT (RRSAF) ACTION(COMMIT) NID(B1D379D17ED6CF900000009401010000)` where the `nid` is the one that you cut from the display command. You can issue this command to roll back the transaction: `-RECOVER INDOUBT (RRSAF) ACTION(ABORT) NID(B1D379D17ED6CF900000009401010000)`
How to determine if DB2 is running	Use the display command to display the DB2 address space.
What happens to WebSphere Application Server for z/OS if DB2 goes down?	WebSphere Application Server for z/OS needs DB2 to run, so it abends if DB2 goes down. You need to restart the LDAP server, then restart DB2, and then restart WebSphere Application Server for z/OS.
What happens to other subsystems if DB2 goes down?	Not applicable
Where to find more information	See the DB2 books at the following Internet location:http://www.ibm.com/servers/eserver/zseries/zos/

CICS automation and recovery scenarios

Table 13. CICS automation and recovery scenarios

Task	CICS automation and recovery scenarios
Startup	CICS and any required CICS products, such as CICS Transaction Gateway, need to be properly installed, initialized, and started before any work flows to a CICS-enabled WebSphere Application Server for z/OS application control server region are run.
Shutdown	Shutdown the WebSphere Application Server for z/OS application control region that uses CICS as a backing store, then shut down the CICS service.
Handling in-flight or indoubt transactions if there is a failure	If there is an error during processing, both CICS and WebSphere Application Server for z/OS rely on the underlying RRS subsystem to handle all rollback notifications to the registered interests. In the case of inflight transactions, RRS will notify all participants that a rollback is required, and normal rollback processing will occur in each registered party. In the case of indoubt transactions, it may be necessary to recycle the WebSphere Application Server for z/OS Application Control/Server region to release any pending transaction in CICS.
How to determine if CICS is running	This is installation dependent.

Table 13. CICS automation and recovery scenarios (continued)

Task	CICS automation and recovery scenarios
What happens to CICS if WebSphere Application Server for z/OS goes down?	Should WebSphere Application Server for z/OS happen to go down, one of two situations could occur: 1. If WebSphere Application Server for z/OS and CICS are currently engaged in a unit of work, then RRS processing as described above would occur and it may be necessary to recycle the application control server regions to release pending transactional work in CICS. 2. If WebSphere Application Server for z/OS and CICS are not currently engaged in a unit of work, CICS is not affected.
What happens to other subsystems if CICS goes down?	Not applicable
Where to find more information	*CICS Operations and Utilities Guide*

IMS automation and recovery scenarios

Table 14. IMS automation and recovery scenarios

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

LDAP automation and recovery scenarios

Table 15. LDAP automation and recovery scenarios

Task	**LDAP automation and recovery scenarios**
Startup	LDAP, as used by WebSphere Application Server for z/OS, is completely run within the WebSphere Application Server for z/OS address spaces using something called "the local backend." This support takes the front side of the LDAP client APIs and the backend database implementation and runs them completely inside the WebSphere Application Server for z/OS Naming Server and Interface Repository. For Naming and IR, OMVS and DB2 must be up *before* Naming and IR. To run the LDAP server, TCPIP, OMVS, and DB2 must all be up before the LDAP server. **Note:** There are two LDAP modes supported: 1. Local LDAP backend. 2. Remote LDAP Server. The WebSphere Application Server for z/OS environment has to be set up correspondingly, and DB2, TCP/IP, and the remote server have to be up and running before WebSphere Application Server for z/OS is started.
Shutdown	Shutdown Naming and IR, then OMVS and DB2. For the LDAP server, shutdown the LDAP server, then TCPIP and DB2, and then OMVS.
Handling in-flight or indoubt transactions if there is a failure	If there is a failure during processing, Naming and IR rely on RRS to issue a rollback directly to DB2 and, as a result, any work done by the LDAP code is rolled back along with it. For the LDAP server, AUTOCOMMIT is set to NO, causing any error to ROLLBACK for that transaction. This ensures the atomicity characteristic of LDAP operations.
How to determine if LDAP is running	In the case of WebSphere Application Server for z/OS, if Naming and IR are operating, then LDAP is operating. In the case of the LDAP server, and a started task is used for the LDAP server, use the SDSF to see if the started task is running. Examine the output log for the started task to see if any error messages were displayed. Alternatively, the `LDAPSRCH` command (from TSO), or `LDAPSEARCH` command (from Unix System Services shell) can be used to perform a simple search to verify that the LDAP server is running.
What happens to WebSphere Application Server for z/OS if LDAP goes down?	• In J2EE server regions, the LDAP server **must** be active since it is a separate server that no longer runs inside the WebSphere Application Server for z/OS region. Recycle the JNDI, then restart the J2EE application servers.
What happens to other subsystems if LDAP goes down?	Most z/OS or OS/390 subsystems do not depend on LDAP, but this may change in the future. In the case of accessing LDAP through the LDAP server, there is a way to configure the LDAP server to operate in a sysplex environment such that (using sysplex-enabled DNS) LDAP requests will be sent to the LDAP server in the sysplex that is operating (assuming that there is one). As an alternative, subsystems that want to use LDAP could configure a backup LDAP server to be contacted in case the primary server is not accessible. In this case, the application would assume that it could retrieve all of the same data that it could get from the backup on the primary which would be handled by some replication mechanism. The LDAP server currently supports a master/slave replication mechanism, but you could also try duplicating the sysplex server using DB2 data sharing.
Where to find more information	*z/OS Security Server LDAP Server Administration and Use*

WebSphere Application Server for z/OS (Daemon) automation and recovery scenarios

Table 16. WebSphere Application Server for z/OS automation and recovery scenarios

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

Web server (servlet) automation and recovery scenarios

Table 17. Web server (servlet) automation and recovery scenarios

Task	WebServer automation and recovery scenarios
Startup	Web servers have a relationship with WebSphere Application Server for z/OS only in the sense that a client application program that is written to use WebSphere Application Server for z/OS facilities may be written as a servlet. Any implications for ordering of startup will be introduced by the applications. You probably want to have the WebSphere Application Server for z/OS object servers up and ready before starting the client application that the web server is hosting.
Shutdown	There are no dependencies from the product code. Similar to most applications, you may want to quiesce the clients prior to taking down the target WebSphere Application Server for z/OS servers. Shut down the web server to stop the port of entry.
Handling in-flight or indoubt transactions if there is a failure	Since a web server is stateless, there are no in-flight or indoubt transactions.
How to determine if a web server is running	Use the z/OS display commands and viewer tools (SDDF) to monitor the Application Server.
What happens to WebSphere Application Server for z/OS if the web server goes down?	WebSphere Application Server for z/OS requires an IBM HTTP J2EE server web server in order to provide full function servlets. So, if the web server goes down, applications that require a port of entry (like servlets and SOAP) cannot run.
What happens to other subsystems if Web Server goes down?	HTTP only provides information to other subsystems, so they are unaffected if the web server goes down.
Where to find more information	*z/OS HTTP Server Planning, Installing, and Using*

Chapter 5. Preparing for a call to 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.
- Reset WebSphere variables that are for use only when directed by IBM service.
- Set WebSphere variable values for the location service daemon address space (same as those for servers, with the prefix "DAEMON_").

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.

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

To view or set your trace settings, use the WebSphere administrative console:

- Click **Environment > Manage WebSphere Variables**.
- On the Configuration Tab check for any of these variables in the name field and observe the variable setting in the value field.
- To change or set a variable, specify the variable in the name field and specify the setting in the value field. You can also describe the setting in the description field on this tab.

Note:

- If you use any level of tracing, including ras_trace_defaultTracingLevel=1, ensure that you set ras_trace_outputLocation to BUFFER.

  ras_trace_defaultTracingLevel=1 will write exceptions to the trace log as well as to the ERROR log.
- Set the ras_trace_BufferCount=4 and ras_trace_BufferSize=128.

  This will get 512KB of storage for the trace buffers (the minimum allowed) and reduce memory requirements.
- It is best to trace to CTRACE.

  If you are tracing to sysprint with ras_trace_defaultTracingLevel=3, you may experience an almost 100% throughput degradation. If you are tracing to CTRACE, however, you may only experience a 15% degradation in throughput.
- Make sure you disable JRAS tracing.

  To do this, look for the following lines in the trace.dat file pointed to by the JVM properties file:

  ```
  com.ibm.ejs.*=all=disable
  ```

  ```
  com.ibm.ws390.orb.*=all=disable
  ```

  Ensure that both lines are set to **=disable** or delete the two lines altogether.

  Note: If ras_trace_outputLocation is set, you may be tracing and not know it.

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.Specifies the default tracing level for WebSphere Application Server for z/OS.

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.

Note: Valid values for ras_trace_basic are:

- 0: RAS
- 1: Common Utilities
- 2: COS/Naming
- 3: COMM
- 4: ORB
- 5: IM
- 6: OTS
- 7: Shasta
- 8: Systems Management
- 9: OS/390 Wrappers
- A: Daemon
- B: IR
- C: Test
- D: COS/Query
- E: Security
- F: Externalization
- G: Adapter
- H: Lifecycle
- I: Identity
- J: JRAS (internal tracing--via direction from IBM support)
- K: Reference collections
- L: J2EE

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.

Note: Valid values for ras_trace_detail are:

- 0: RAS
- 1: Common Utilities
- 2: COS/Naming
- 3: COMM
- 4: ORB
- 5: IM
- 6: OTS
- 7: Shasta
- 8: Systems Management
- 9: OS/390 Wrappers
- A: Daemon
- B: IR
- C: Test
- D: COS/Query
- E: Security
- F: Externalization
- G: Adapter
- H: Lifecycle
- I: Identity
- J: JRAS (internal tracing--via direction from IBM support)
- K: Reference collections
- L: J2EE

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.

Enables traceback of system exception minor codes. Values are:

- *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
  ```

Chapter 6. Diagnosing and fixing problems: Resources for learning

In addition to this InfoCenter, there are several Web-based resources for researching and resolving problems related to the WebSphere Application Server.

The WebSphere Application Server support page

The official site for providing tools and sharing knowledge about WebSphere Application Server problems is the WebSphere Application Server support page: http://www.ibm.com/software/webservers/appserv/support.html. Among the features it provides are:

- A search field for searching the entire support site for documentation and fixes related to a specific exception, error message, or other problem. Use this search function before contacting IBM Support directly.
- *Hints and Tips*, *Technotes*, and *Solutions* links take you to specific problems and resolutions documented by WebSphere Application Server technical support personnel.
- A link *All fixes, fix packs, and tools* provides free WebSphere Application Server maintenance upgrades and problem determination tools.
  - *fixes* are software patches which address specific WebSphere Application Server defects. Selecting a specific defect from the list in the *All fixes, fix packs, and tools* page takes you to a description of what problem the fix addresses.
  - Fix packs are bundles of multiple fixes, tested together and released as a maintenance upgrade to WebSphere Application Server. If you select a fix pack from this page, you are taken to a page describing the target platform, WebSphere Application Server prerequisite level, and other related information. Selecting the *list defects* link on that page displays a list of the fixes which the fix pack includes. If you intend to install an fix which is part of a fix pack, it is usually better to upgrade to the complete fix pack rather than to just install the individual fix.
  - Tools are free programs that help you analyze the configuration, behavior and performance of your WebSphere Application Server installation.

Accessing WebSphere Application Server support page resources

Some resources on the WebSphere Application Server support page are marked with a key icon. To access these resources, you must supply a user ID and password, or to register if do not already have an ID. When registering, you are asked for your contract number, which is supplied as part of a WebSphere Application Server purchase.

WebSphere Developer Domain

The Developer Domains are IBM-supported sites for enabling developers to learn about IBM software products and how to use them. They contain resources such as articles, tutorials, and links to newsgroups and user groups. You can reach the WebSphere Developer Domain at http://www7b.software.ibm.com/wsdd/.

The IBM Support page

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Chapter 7. Troubleshooting by task: what are you trying to do?

This section provides troubleshooting information based on the task you were trying to accomplish when the problem occurred. To find more information about your problem, select a task category from the list below.

Troubleshooting installation problems

Select the problem you are having with WebSphere Application Server installation:
- The installation process completes, but sample applications, such as the snoop servlet or other applications from the Sample Gallery do not work.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Installation completes but the administrative console does not start

What kind of problem are you having?
- ″Internal Server Error″, ″Page cannot be found″, 404, or similar error trying to view administrative console.
- ″Unable to process login. Please check User ID and password and try again. ″ error when trying to access console page.
- Directory paths in the console are garbled.

If you are able to bring up the browser page, but the console's behavior is inconsistent, error-prone, or unresponsive, try upgrading the browser you are using. Older browsers may not support the administrative console's features. For a listing of supported Web browsers, see http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html.

"Internal Server Error", "Page cannot be found", 404, or similar error trying to view administrative console

If you are unable to view the administrative console, here are some steps to try:
- Verify that the application server which supports the administrative console is up and running.
  - For a ″base″ configuration, the administrative console is deployed by default on ″server1″. Before viewing the administrative console, you must verify ″server1″ is running. Do so by issuing the following command on the MVS console to list active processes:

    ```
    D A,L
    ```

    Note: See z/OS MVS System Commands for information on how to use MVS operator commands.
    Check for the application server procedure name of BB05ACR with the server short name of BB0S001. If it isn't running, enter the following command on the MVS console:

    ```
    START appserver_proc_name,JOBNAME=server_short_name,
          ENV=cell_short_name.node_short_name.server_short_name
    ```

    Example:

    ```
    START BBO5ACR,JOBNAME=BBOS001,ENV=PLEX1.SY1.BBOS001
    ```
  - For a Network Deployment configuration, the Deployment Manager runs the administrative console. To start the Deployment Manager, you can, for example, issue the following command from the MVS console:

```
START BBO5DCR,JOBNAME=BBODMGR,ENV=PLEX1.PLEX1.BBODMGR
```
  – View the joblog or sysprint for the application server or deployment manager to verify that the server supporting the administrative console has actually started.
- Check the URL you are using to view the console. By default, it is http://*server_name*:9080/admin., where *server_name* is the host name.
- Try to eliminate connection, address and firewall issues by:
  – Pinging the server machine from a command prompt, using the same server name as in the URL.

"Unable to process login. Please check User ID and password and try again. " error when trying to access console page

This error indicates that security has been enabled for WebSphere Application Server, and the user ID or password supplied is either invalid or not authorized to access the console.

To access the console,
- If you are the administrator, use the ID defined as the security administrative ID. This ID is stored in the WebSphere Application Server directory structure in the file security.xml.

Directory paths in the console are garbled

If directory paths used for classpaths or resources specified in the Application Assembly Tool, configuration files, or elsewhere, appear garbled in the administrative console, it may be because the Java runtime interprets a backslash (\) as denoting a control character.

To resolve, make sure you have no backslashes in your classpaths. You can only have forward slashes in them.

Installation completes, but sample applications do not work

If the WebSphere Application Server installation program completes successfully, but the sample applications do not run:
- Browse the application server logs for clues.
- Look up any error or warning messages in the message table by selecting the **Reference** view of this InfoCenter and expanding the **Messages** heading.
- You can also encounter some security-related problems, such as after turning on security, ″MSGS0508E: The JMS Server security service was unable to authenticate userid:″ error is displayed in the error log when starting an application server.

Troubleshooting testing and first time run problems

Select the problem you are having with testing or the first run of deployed code for WebSphere Application Server:
- A Web resource, such as a JSP, servlet, HTML file, or image, does not display.
- Cannot access a datasource.
- Cannot access an enterprise bean from a servlet, JSP file, stand-alone program, or other client.
- Cannot access an object hosted by WebSphere Application Server, such as an enterprise bean or connection pool, from a servlet, JSP file, stand-alone program, or other client.
- I have errors and access problems after enabling security.
- I get errors after enabling Secure Sockets Layer (SSL), or SSL-related error messages.
- I have problems with messaging.
- I get errors when trying to send a SOAP request.
- A .
- I get errors connecting to WebSphere MQ and creating WebSphere MQ queue connection factory.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Web resource (JSP file, servlet, HTML file, image) does not display

What kind of error do you see when you start an application?
- HTTP server and Application Server are working separately, but requests are not passing from HTTP server to Application Server.
- File serving problems (html, images, etc).
- Graphics do not appear on jsp or servlet output.
- SRVE0026E: [Servlet Error]-[Unable to compile class for JSP error on JSP.
- After modifying and saving a JSP, the change does not show up in the browser (the old JSP displays).
- Message similar to ″Message: /jspname.jsp(9,0) Include: Mandatory attribute page missing″ displays when trying to access JSP.
- The Java source generated from a JSP is not retained in the temp directory (only the classfile is found).
- The JSP Batch Compiler fails with the message ″Enterprise Application [application name you typed in] not found.″
- Non-English browser input is garbled.
- Scroll bars do not appear around items in the browser window.
- Error ″Page cannot be displayed... server not found or DNS error″ appears when attempting to browse a Java Server Page (JSP) using Internet Explorer.

Otherwise, if you are not able to display a resource in your browser follow these steps:
- Verify that your HTTP server is healthy by accessing the URL`http://`*`server_name`* from a browser and seeing whether the ″Welcome page″ appears. This indicates whether the HTTP server is up and running, regardless of the state of WebSphere Application Server.
- If the HTTP server ″Welcome page″ does not appear, that is, if you get a browser message such as ″page cannot be displayed″ or something similar, try to diagnose your Web server problem.
- If the HTTP server appears to be functioning, the problem is:
  - The Application Server may not be serving the target resource. To see if this is the case, try accessing the resource directly through the Application Server instead of through the HTTP server.
  - If you cannot access the resource directly through the Application Server:
    - Verify that the URL used to access the resource is correct.
    - If the URL is incorrect and it is created as a link from another JSP file, servlet, or HTML file:
      - After clicking the link, try correcting it by hand in the browser's URL field and reloading, to confirm that the problem is a malformed URL. If this is the problem, correct the URL in the ″from″ HTML file, servlet or jsp file.
    - If the URL appears to be correct, but the resource cannot be accessed directly through the Application Server, verify the health of the hosting Application Server and Web module:
      - View the hosting Application Server and Web module in the administrative console to verify they are up and running.
      - Copy a simple HTML or JSP file (such as `SimpleJsp.jsp` in the WebSphere Application Server directory structure) to your Web module document root, and try to access it. If this works, the problem is with your resource. View the logs of your Application Server to find out why your resource cannot be found or served
- If the resource can be accessed directly through the Application Server, but not through an otherwise healthy HTTP server, the problem lies with the HTTP plug-in -- the component that communicates between the HTTP server and the WebSphere Application Server.
- If JSP and servlet output is served, but not static resources such as `.html` and image files, see the steps for enabling file serving.
- If some kinds of resources display correctly, but you cannot display a servlet by its class name:
  - Ensure that the servlet is in a directory in the Web module classpath, such as in the `/`*`Web_module_name`*`.war/WEB-INF/classes directory`.
  - Ensure that you specify the full class name of the servlet, including its package name, in the URL.
  - Ensure that ″`/servlet`″ precedes the class name in the URL. For example: if the root context of a Web module is ″`myapp`″, and the servlet is `com.mycom.welcomeServlet`, then the URL should read:

http://<hostname>/myapp/servlet/com.mycom.welcomeServlet
  - Ensure that serving servlets by classname is enabled for the hosting Web module by opening the source Web module in the Application Assembly Tool and browse the ″serve servlets by classname″ setting in the IBM Extensions property page. If necessary, enable this flag and redeploy the Web module.
  - For servlets or other resources served by mapped URLs, the URL is `http://`*`hostname/web module context root/mappedURL`*.

Accessing a web resource through the application server (bypassing the HTTP server)

Starting with WebSphere Application Server version 4.0, you can bypass the HTTP server and access a web resource through the application server. It is not recommended to serve a production Web site in this way, but it provides a good diagnostic tool when it is not clear whether a problem resides in the HTTP server, WebSphere Application Server, or the HTTP plug-in.

To access a Web resource through the Application Server:
- Find out the port of the HTTP service in the target Application Server.
  - In the WebSphere Administrative Console, select **Servers->Manage Application Servers**.
  - Select the target server, then under Additional Properties select **Web Container**.
  - Under the Additional Properties of the Web Container, select **HTTP Transports**. You will see the ports listed for virtual hosts served by the Application Server.
- Using the HTTP transport port number of the Application Server, access the resource from a browser. For example, if the port is 9080, the URL would be `http://`*`hostname`*`:9080\myAppContext\myJSP.jsp`.
- If you are still unable to access the resource, ensure that the HTTP transport port is in the ″Host Alias″ list:
  1. Select **Application Servers>Your_ApplicationServer>Web Container>HTTP Transports** to check the Default virtual host and the HTTP transport ports used by this Application Server.
  2. Select **Environment>Manage Virtual Hosts>default host>Host Aliases** to check if the HTTP transport port exists. Add an entry if necessary. For example, if the HTTP port for your application is server is 9080, add a host alias of *:9082.

HTTP server and Application Server are working separately, but requests are not passing from HTTP server to Application Server

If your HTTP server appears to be functioning correctly, and the Application Server also works on its own, but browser requests sent to the HTTP server for pages are not being served, this indicates a problem in the WebSphere Application Server plug-in.

If this is the case:
- Determine whether the HTTP server is attempting to serve the requested resource itself, rather than forwarding it to the WebSphere Application Server.
  - Browse the HTTP server access log (*`IHS install root`*`\logs\access.log` for IBM HTTP Server). It may indicate that it could not find the file in its own document root directory.
  - browse the plug-in log file as described below.
- The file *`install_dir`*`/config/plugin-cfg.xml` determines which requests sent to the HTTP server are forwarded to the WebSphere Application Server, and to which Application Server. You may need to refresh this file:
  - In the WebSphere Application Server administrative console, expand the Environment tree control.
  - Select **Update WebSphere Plugin**.
  - Stop and restart the HTTP server and retry the Web request.
- Browse the file *`install_dir`*`/logs/http_plugin.log` for clues to the problem. Make sure the timestamps with the most recent Plugin Information stanza, which is printed out when the plug-in is loaded, correspond to the time the Webserver was started.
- Turn on plug-in tracing by setting the `LogLevel` attribute in the *`install_dir`*`/config/plugin-cfg.xml` file to `Trace` and reloading the request, then browsing the *`install_dir`*`/logs/http_plugin.log` file. You should be able to see the plug-in attempting to match the request URI with the various URI definitions

for the routes in the plugin-cfg.xml. You should be able to see what rules the plug-in is not matching against and then figure out if you need to add additional ones. If you just recently installed the application you may need to manually regenerate the plug-in configuration in order to pick up the new URIs related to the new application.

File serving problems (html, images, etc)

If text output appears on your JSP- or servlet-supported Web page, but image files do not:

- Ensure that your files are in the right place: the **document root** directory of your Web application WebSphere Application Server follows the J2EE standard, which means that the document root is the *Web_module_name*.war directory of your deployed Web application. Typically this directory will be found in the *installation_root*/installedApps/*nodename*/*appname*.ear or *installation_root*/installedApps/*nodename*/*appname*Network.ear directory.

  If the files are in a subdirectory of the document root, verify that the reference to the file reflects that. That is, if invoices.html is stored in Windows directory *Web_module_name*.war\invoices, then links from other pages in the Web application to display it should read "invoices\invoices.html", not "invoices.html".
- Ensure that your Web application is configured to enable file serving (i.e., display of static resources like image and .html files):
  – View the file serving property of the hosting Web module by browsing the source .war file in the Application Assembly Tool (AAT). If necessary, update the property and re-deploy the module.
  – Edit the **fileServingEnabled** property in the deployed Web application ibm-web-ext.xmi configuration file, typically found in the *install_root*/config/cells/*nodename* or *nodename*Network/applications/*application name*/deployments/*application name*/*Webmodule name*/web-inf directory.

Graphics do not appear on jsp or servlet output

If text output appears on your JSP- or -servlet-supported Web page, but image files do not:

- Ensure that your graphic files are in the right place: the **document root** directory of your Web application WebSphere Application Server 5 follows the J2EE standard, which means that the document root is the *Web_module_name*.war directory of your deployed Web application. Typically this directory will be found in the *installation_root*/installedApps/*nodename*/*appname*.ear or *installation_root*/installedApps/*nodename*/*appname*Network.ear directory.

  If the graphics files are in a subdirectory of the document root, verify that the reference to the graphic reflects that; e.g., if banner.gif is stored in Windows directory *Web_module_name*.war/images, the tag to display it should read: **<img SRC="images/banner.gif">**, not <img SRC="banner.gif">.
- Ensure that your Web application is configured to enable file serving (i.e., display of static resources like image and .html files).
  – View the file serving property of the hosting Web module by browsing the source .war file in the AAT. If necessary, update the property and re-deploy the module. Or
  – Edit the **fileServingEnabled** property in the deployed Web application ibm-web-ext.xmi configuration file, typically found in the *install_root*/config/cells/*nodename* or *nodename*Network/applications/*application name*/deployments/*application name*/*Webmodule name*/web-inf directory.
  – After following one of the above steps:
    - In the administrative console, expand the **Environment** tree control .
    - Click the link **Update WebSphere Plugin**.
    - Stop and restart the HTTP server and retry the Web request.

SRVE0026E: [Servlet Error]-[Unable to compile class for JSP

If this error appears in a browser when trying to access a new or modified .jsp file for the first time, the most likely cause is that the JSP Java source failed (was incorrect) during the javac compilation phase.

To confirm that this is the problem, check the logs for a compiler error message, such as:

```
Duplicate variable declaration: int myInt was int myInt
int myInt = 122;
String myString = "number is 122";
static int myStaticInt=22;
int myInt=121;
        ^
```

If this is the problem, fix the problem in the JSP source, save the source and re-request the JSP.

If this error occurs when trying to serve a JSP that was copied from another system where it ran successfully, then there is something different about the new server environment that prevents the JSP from running.

Browse the text of the error for a statement like:

```
 Undefined variable or class name: MyClass
```

This error indicates that a supporting class or jar file has not been copied to the target server, or is not on the classpath. To resolve, find the file `MyClass.class`, and place it on the Web module `WEB-INF/classes` directory, or place its containing `.jar` file in the Web module `WEB-INF/lib` directory.

Verify that the URL used to access the resource is correct

- For a JSP file, `html` file, or image file: **`http://`*`host_name/Web_module_context_root/subdir under doc root, if any/filename.ext`***. The document root for a web application is the *`application_name`*`.WAR` directory of the installed application.
  - JSP serving is enabled by default. File serving for `html` and image files must be enabled as a property of the Web module, in the Application Assembly Tool, or by setting the **`fileServingEnabled`** property to `"true"` in the `ibm-web-ext.xmi`file of the installed Web application and restarting the application.
- For servlets served by class name, the URL is `http://`*`hostname/Web_module_context_root`*`/servlet/`*`packageName.className`*.
  –
- Serving servlets by classname must be enabled as a property of the Web module, and is enabled by default. File serving for html and image files must be enabled as a property of the Web application, in the Application Assembly Tool, or by setting the **`fileServingEnabled`** property to `"true"` in the **`ibm-web-ext.xmi`** file of the installed Web application and restarting the application.

Correct the URL in the "from" html file, servlet or jsp

An HREF with no leading "/" inherits the calling resource context. For example:

- an HREF in `http://[hostname]/myapp/servlet/MyServlet` to `"ServletB"` resolves to `"http://hostname/myapp/servlet/ServletB"`
- an HREF in `http://[hostname]/myapp/servlet/MyServlet` to `"servlet/ServletB"` resolves to `"http://hostname/myapp/servlet/servlet/ServletB"` (an error)
- an HREF in `http://[hostname]/myapp/servlet/MyServlet` to `"/ServletB"` resolves to `"http://hostname/ServletB"` (an error, if `ServletB` requires the same context root as `MyServlet`)

After modifying and saving a JSP, the change does not show up in the browser (the old JSP displays)

The most likely cause of this error is that the Web application is not configured for servlet reloading, or the reload interval is too high.

To correct this problem, in the Application Assembly Tool, check the **Reloading Enabled** flag and the **Reload Interval** value in the IBM Extensions for the the Web module in question. Turn Reloading on, or if it is already on then set the Reload Interval lower.

Message like "Message: /jspname.jsp(9,0) Include: Mandatory attribute page missing" appears when attempting to browse JSP

The most likely cause of this error is that the JSP file failed during the translation to Java phase. Specifically, a JSPdirective, in this case an Include statement, was incorrect or referred to a file that could not be found.

To correct this problem, fix the problem in the JSP source, save the source and re-request the JSP.

The Java source generated from a JSP is not retained in the temp directory (only the classfile is found)

The most likely cause of this error is that the JSP Processor is not configured to keep generated Java source.

To correct this problem, in the Application Assembly Tool, check the **JSP Attributes** under **Assembly Property Extensions** for the Web module in question. Make sure the attribute **keepgenerated** is there and is set to true. If not, set this attribute and restart the Web application. To see the results of this operation, you will have to delete the classfile from the `temp` directory in order to force the JSP Processor to retranslate the JSP source into Java.

The JSP Batch Compiler fails with the message "Enterprise Application [application name you typed in] not found."

The most likely cause of this error is that the full Enterprise Application path and name, starting with the `.ear` subdirectory that resides in the *install_root*`\config\cells\`*node_name*`Network\applications` directory is expected as an argument to the JspBatchCompiler tool, not just the display name. For example:
- `″JspBatchCompiler -enterpriseapp.name sampleApp.ear/deployments/sampleApp″` is correct, as opposed to
- `″JspBatchCompiler -enterpriseapp.name sampleApp″`, which is incorrect.

Non-English browser input is garbled

If non-English-character-set browser input is apparently garbled after being read by a servlet or JSP, ensure that the request parameters are encoded according to the expected chararacter set before being read. For example, if the site is Chinese, the target `.jsp` should have a line:

```
req.setCharacterEncoding(″gb2312″);
```

before any req.getParameter() calls.

Note: This problem especially affects servlets and `jsps` ported from earlier versions of WebSphere Application Server, which converted characters automatically based upon the locale of the WebSphere Application Server.

Scroll bars do not appear around items in the browser window

In some browsers, tree or list type items that extend beyond their allotted windows do not have scroll bars to allow you to see the entire list.

To correct this problem, right click on the browser window and select **Reload** from the pop-up menu.

Error "Page cannot be displayed... server not found or DNS error" appears when attempting to browse a Java Server Page (JSP) using Internet Explorer

This error can occur when an HTTP timeout causes the servant to be brought down and restarted. To correct this problem, increase the ConnectionIOTimeout . value:

1. From the adminstrative console select System Administration > DeploymentManager > Administration Services > Custom Properties
2. Select ConnectionIOTimeout
3. Increase the ConnectionIOTimeout value
4. Click OK.

Cannot access a data source

What kind of database are you trying to access?

- Oracle
- DB2
- SQL Server
- Cloudscape
- Sybase
- My problem was not described under the topic for my database, or might not be DBM specific.

If none of these errors match the one you see:

1. Browse the log files of the application server for clues.
2. Browse the Helper Class property of the data source to verify that it is correct and that it is on the WebSphere Application Server classpath. Mysterious errors or behavior might be the result of a missing or misnamed Helper Class name. If WebSphere Application Server is not able to load the specified class, it uses a default helper class that might not function correctly with your database manager.

What kind of error do you see when you try to access your Oracle-based datasource

- ″DSRA8100E: Unable to get a {0} from the DataSource. Explanation: See the linkedException for more information.″
- Invalid Oracle URL specified
- ″DSRA0080E: An exception was received by the Data Store Adapter. See original exception message: ORA-00600″ when connecting to or using an Oracle data source.
- ″Error while trying to retrieve text for error″ error when connecting to an Oracle data source.
- java.lang.UnsatisfiedLinkError connecting to an Oracle data source.
- java.lang.NullPointerException or ″internal error: oracle.jdbc.oci8.OCIEnv″ connecting to an Oracle data source.
- WSVR0016W: Classpath entry, ${ORACLE_JDBC_DRIVER_PATH}/classes12.zip, in Resource, Oracle JDBC Thin Driver, located at cells/BaseApplicationServerCell/nodes/wasrtp/resources.xml has an invalid variable.

What kind of problem are you having accessing your DB2 database?

- SQL0805N Package ″*package name*″ was not found.
- SQLException, with ErrorCode -99,999 and SQLState 58004, with java ″StaleConnectionException: COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver] CLI0119E Unexpected system failure. SQLSTATE=58004″ using WAS40-type data source.
-
- SQL0805N Package ″NULLID.SQLLC300″ was not found. SQLSTATE=51002.
- SQL0567N ″DB2ADMIN″ is not a valid authorization ID. SQLSTATE=42602.
- CLI0119E System error. SQLSTATE=58004 - DSRA8100 : Unable to get a XAconnection, or DSRA0011E: Exception: COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver] CLI0119E Unexpected system failure. SQLSTATE=58004.

- COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver][DB2/NT] SQL0911N The current transaction has been rolled back because of a deadlock or timeout. Reason code ″2″. SQLSTATE=40001.
- ″COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource″ could not be found for data source (*data_source*).

What kind of problem are you having accessing your SQL Server database?
- ERROR CODE: 20001 and SQL STATE: HY000.
- Application fails with message stating ″Cannot find stored procedure...″

What kind of problem are you having accessing your Cloudscape database?
- Unexpected IOException wrapped in SQLException, accessing Cloudscape database.
- ″Select for update″ on one row causes table to become locked, triggering a deadlock condition.
- ″ERROR XSDB6: Another instance of Cloudscape might have already booted the database *databaseName*.″ error starting application server.
- Error ″The version of the IBM Universal JDBC driver in use is not licensed for connectivity to Cloudscape databases″
- Running an application causes a runtime exception which produces an unreadable message.

Note: Cloudscape errorCodes (2000, 3000, 4000) indicate levels of severity, not specific error conditions. In diagnosing Cloudscape problems, pay attention to the given sqlState value.

What kind of problem are you having accessing your Sybase database?
- SET CHAINED command not allowed within multi-statement transaction.
- ″Sybase Error 7713: Stored Procedure can only be executed in unchained transaction mode″ error.
- ″JZ0XS: The server does not support XA-style transactions. Please verify that the transaction feature is enabled and licensed on this server.″
- A Container Managed Persistence (CMP) enterprise bean is causing exceptions.

What kind of general data access problem do you have?
- ″IllegalConnectionUseException″
- WTRN0062E: An illegal attempt to enlist multiple one phase capable resources has occurred.
- ConnectionWaitTimeoutException.
- com.ibm.websphere.ce.cm.StaleConnectionException: [IBM][CLI Driver] SQL1013N The database alias name or database name ″NULL″ could not be found. SQLSTATE=42705
- java.sql.SQLException: java.lang.UnsatisfiedLinkError:
- ″J2CA0030E: Method enlist caught java.lang.IllegalStateException″ wrapped in error ″WTRN0063E: An illegal attempt to enlist a one phase capable resource with existing two phase capable resources has occurred″ when attempting to execute a transaction.
- java.lang.UnsatisfiedLinkError:xaConnect exception when attempting a database operation

DSRA8100E: Unable to get a {0} from the DataSource. Explanation: See the linkedException for more information.

When using oracle thin driver, Oracle throws ″java.sql.SQLException: invalid arguments in call″ if no username or password is specified when getting a connection. If you see this while running WebSphere Application Server, the alias is not set.

To remove the exception, define the alias on the data source.

Invalid Oracle URL specified

This error might be caused by an incorrectly specified URL on the URL property of the target data source.

Examine the URL property for the data source object in the administrative console. For the 8i OCI driver, ensure that **oci8** is used in the URL. For the 9i OCI driver, you can use either **oci8** or **oci**.

Examples of Oracle URLs:

- For the thin driver: jdbc:oracle:thin:@hostname.rchland.ibm.com:1521:IBM
- For the thick (OCI) driver: jdbc:oracle:oci8:@tnsname1

"DSRA0080E: An exception was received by the Data Store Adapter. See original exception message: ORA-00600" when connecting to or using an Oracle data source "DSRA0080E: An exception was received by the Data Store Adapter. See original exception message: ORA-00600" when connecting to or using an Oracle data source "DSRA0080E: An exception was received by the Data Store Adapter. See original exception message: ORA-00600" when connecting to or using an Oracle data source

A possible reason for this exception is that the version of the Oracle JDBC driver being used is older than the Oracle database. It is possible that more than one version of the Oracle JDBC driver has been configured on the WebSphere Application Server.

To confirm that this is the cause of the problem, examine the version of the JDBC driver. Sometimes you can determine the version by looking at the classpath to determine what directory the driver is in.

If you cannot determine the version this way, use the following program to determine the version. Before running the program, set the classpath to the location of your JDBC driver files.

```
import java.sql.*;
     import oracle.jdbc.driver.*;
     class JDBCVersion
     {
          public static void main (String args[])
          throws SQLException
          {
               // Load the Oracle JDBC driver
               DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
               // Get a connection to a database
               Connection conn = DriverManager.getConnection
               (″jdbc:oracle:thin:@appaloosa:1521:app1″,″sys″,″change_on_install″);
               // Create Oracle DatabaseMetaData object
               DatabaseMetaData meta = conn.getMetaData();
               // gets driver info:
               System.out.println(″JDBC driver version is ″ + meta.getDriverVersion());
          }
     }
```

If the driver and the database are at different versions, replace the JDBC driver with the correct version. If multiple drivers are configured, remove any that are at the incorrect level.

"Error while trying to retrieve text for error" error when connecting to an Oracle data source

The most likely cause is that the Oracle 8i OCI driver is being used with an ORACLE_HOME property that is either not set or is set incorrectly.

To correct the error, examine the user profile that WebSphere Application Server is running under to verify that the $ORACLE_HOME environment variable is set correctly.

"java.lang.UnsatisfiedLinkError:" connecting to an Oracle data source

The problem might be that the environment variable LIBPATH is not set or is set incorrectly, if your data source throws an **UnsatisfiedLinkError**, and the full exception indicates that the problem is related to an Oracle module, as in the following examples.
- Example of invalid LIBPATH for the 8i driver:

```
Exception in thread ″main″ java.lang.UnsatisfiedLinkError:
/usr/WebSphere/AppServer/java/jre/bin/libocijdbc8.so:
load ENOENT on shared library(s)
/usr/WebSphere/AppServer/java/jre/bin/libocijdbc8.so libclntsh.a
```

- Example of invalid LIBPATH for the 9i driver:

```
Exception in thread ″main″ java.lang.UnsatisfiedLinkError:
no ocijdbc9  (libocijdbc9.a or .so) in java.library.path
at java.lang.ClassLoader.loadLibrary(ClassLoader.java(Compiled Code))
at java.lang.Runtime.loadLibrary0(Runtime.java:780)
```

To correct the problem, examine the user profile that WebSphere Application Server is running under to verify that the LIBPATH environment variable includes Oracle libraries. Scan for the file `lobocijdbc8.so` to find the right directory.

java.lang.NullPointerException referencing 8i classes, or " internal error: oracle.jdbc.oci8. OCIEnv" connecting to an Oracle data source

The problem might be that the 9i OCI driver is being used on an AIX 32 bit machine, the LIBPATH is set correctly, but the ORACLE_HOME is not set or is set incorrectly, if you encounter an exception similar to either of the following, when your application attempts to connect to an Oracle data source:

- Exception example for java.lang.NullPointerException:

```
Exception in thread ″main″ java.lang.NullPointerException
at oracle.jdbc.oci8.OCIDBAccess.check_error(OCIDBAccess.java:1743)
at oracle.jdbc.oci8.OCIEnv.getEnvHandle(OCIEnv.java:69)
at oracle.jdbc.oci8.OCIDBAccess.logon(OCIDBAccess.java:452)
at oracle.jdbc.driver.OracleConnection. <init>(OracleConnection.java:287)
```

- Exception example for java.sql.SQLException:

```
Exception in thread ″main″ java.sql.SQLException:
internal error: oracle.jdbc.oci8. OCIEnv@568b1d21
at oracle.jdbc.dbaccess.DBError.throwSqlException(DBError.java:184)
at oracle.jdbc.dbaccess.DBError.throwSqlException(DBError.java:226)
at oracle.jdbc.oci8.OCIEnv.getEnvHandle(OCIEnv.java:79)
```

To correct the problem, examine the user profile that WebSphere Application Server is running under to verify that it has the `$ORACLE_HOME` environment variable set correctly, and that the `$LIBPATH` includes `$ORACLE_HOME/lib`.

WSVR0016W: Classpath entry, ${ORACLE_JDBC_DRIVER_PATH}/classes12.zip, in Resource, Oracle JDBC Thin Driver, located at cells/BaseApplicationServerCell/nodes/wasrtp/resources.xml has an invalid variable

This error occurs when no environment variable is defined for the property, ORACLE_JDBC_DRIVER_PATH.

Verify that this is the problem in the administrative console. Go to **Environment > Manage WebSphere Variables** to verify whether the variable `ORACLE_JDBC_DRIVER_PATH` is defined.

To correct the problem, click **New** and define the variable. For example, name : **ORACLE_JDBC_DRIVER_PATH** , value : **c:\oracle\jdbc\lib** Use a value that names the directory in your operating system and directory structure that contains the `classes12.zip` file.

SQL0805N Package "<package-name>" was not found

Possible reasons for these exceptions are:

- If the package name is `NULLID.SQLLC300`, see SQL0805N Package ″NULLID.SQLLC300″ was not found. SQLSTATE=51002. for the reason.
- You are attempting to use an XA-enabled JDBC driver on a DB2 database that is not XA-ready.

To correct the problem on a DB2/UDB database, run this one-time procedure, using the db2cmd interface while connected to the database in question:

1. **DB2 bind @db2ubind.lst blocking all grant public**
2. **DB2 bind @db2cli.lst blocking all grant public**

The `db2ubind.lst` and `db2cli.lst` files are in the `bnd` directory of your DB2 install_root. Run the commands from that directory.

SQLException, with ErrorCode -99,999 and SQLState 58004, with java "StaleConnectionException: COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver] CLI0119E Unexpected system failure. SQLSTATE=58004", when using WAS40-type data source

An unexpected system failure usually occurs when running in XA mode (two-phase commit). Among the many possible causes are:

- An invalid username or password was provided.
- The database name is incorrect.
- Some DB2 packages are corrupted.

To determine whether you have a username or password problem, look in the `db2diag.log` file to view the actual error message and SQL code. A message like the following, with an SQLCODE of -1403, indicates an invalid user ID or password:

```
2002-07-26-14.19.32.762905    Instance:db2inst1    Node:000
PID:9086(java)    Appid:*LOCAL.db2inst1.020726191932
XA DTP Support  sqlxa_open    Probe:101
DIA4701E Database ″POLICY2″ could not be opened
for distributed transaction processing.
String Title: XA Interface SQLCA  PID:9086 Node:000
SQLCODE = -1403
```

To resolve these problems:

1. Correct your username and password. If you specify your password on the GUI (for 40 Datasource), ensure that the username and password you specify on the bean are correct. The username and password you specify on the bean overwrite whatever you specify when creating the data source.
2. Use the correct database name.
3. Rebind the packages (in the `bnd` directory) as follows:

   ```
   db2connect to dbname
   c:\SQLLIB\bnd>DB2 bind @db2ubind.lst blocking all grant public
   c:\SQLLIB\bnd>DB2 bind @db2cli.lst blocking all grant public
   ```
4. Ensure that the file \WebSphere\AppServer\properties\wsj2cdpm.properties has the right userid and password.

SQL0805N Package "NULLID.SQLLC300" was not found. SQLSTATE=51002

Some possible causes of this error are:

- The underlying database was dropped and recreated.
- DB2 was ugpraded, and its packages are not rebound correctly.

To resolve this problem, rebind the DB2 packages by running the **db2cli.lst** script found in the `bnd` directory. For example:**db2>@db2cli.lst**.

SQL0567N "DB2ADMIN " is not a valid authorization ID. SQLSTATE=42602

If you encounter this error when attempting to access a DB2/UDB data source:
1. Verify that your username and password in the data source properties in the admin console, are correct.
2. Ensure that the userid and password do not contain blank characters (before, in between, or after).

CLI0119E System error. SQLSTATE=58004 - DSRA8100 : Unable to get a XAconnection or DSRA0011E: Exception: COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver] CLI0119E Unexpected system failure. SQLSTATE=5800

If you encounter this error when attempting to access a DB2/UDB data source:
1. Check your username and password ″custom properties″ in the data source properties page in the admin console. Ensure that they are correct.
2. Ensure the userid and password do not contain any blank characters (before, in between, or after).
3. Check that the `WAS.policy` file exists for the application.
4. View the entire exception listing for an underlying SQL error, and look it up using the DBM vendor message reference.

If you encounter this error while running DB2 on Red Hat Linux, the `max queues system wide` parameter is too low to allow DB2 to acquire the necessary resources to complete the transaction. When this is the problem, exception DSRA8100E can be preceded by exceptions J2CA0046E and DSRA0010E.

To correct this problem, edit the file `/proc/sys/kernal/msgmni` to increase the value of the `max queues system wide` parameter to a value greater than 128.

COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver][DB2/NT] SQL0911N The current transaction has been rolled back because of a deadlock or timeout. Reason code "2". SQLSTATE=40001

This is probably an application-caused DB2 deadlock, particularly if you see an error similar to the following when accessing a DB2 data source:

```
ERROR CODE: -911
COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver][DB2/NT] SQL0911N
The current transaction has been rolled back because of a deadlock or timeout.
Reason code ″2″.  SQLSTATE=40001
```

To diagnose the problem:
1. Execute these DB2 commands:
   a. **db2 update monitor switches using LOCK ON**
   b. **db2 get snapshot for LOCKS on dbName > *directory_name*\lock_snapshot.log**

   The `directory_name\lock_snapshot.log` now has the DB2 lock information.
2. Turn off the lock monitor by executing: **db2 update monitor switches using LOCK OFF**

To verify that you have a deadlock:
1. Look for an application handle that has a lock-wait status, then look for the **ID of agent holding lock** to verify the ID of the agent.
2. Go to that handle to verify it has a lock-wait status, and the ID of the agent holding the lock for it. If it is the same agent ID as the previous one, then you know that you have a circular lock (deadlock).

To resolve the problem:
1. Examine your application and use a less restrictive isolation level if no concurrency access is needed.

2. Use caution when moving to a lesser **accessIntent**, which can result in data integrity problems.
3. For DB2/UDB Version 7.2 and earlier releases, you can set the DB2_RR_TO_RS flag from the DB2 command line window to eliminate unnecessary deadlocks, such as when the accessIntent defined on the bean method is too restrictive, for example, PessmisticUpdate. The DB@_RR_TO_RS setting has two impacts:
   - If RR is your chosen isolation level, it is effectively downgraded to RS.
   - If you choose another isolation level, and the DB2_RR_TO_RS setting is on, a scan skips over rows that have been deleted but not committed, even though the row might qualify for the scan. The skipping behavior affects the RR, Read Stability (RS), and Cursor Stability (CS) isolation levels.

   For example, consider the scenario where transaction A deletes the row with column1=10 and transaction B does a scan where column1>8 and column1<12. With DB2_RR_TO_RS off, transaction B waits for transaction A to commit or rollback. If transaction A rolls back, the row with column1=10 is included in the result set of the transaction B query. With DB2_RR_TO_RS on, transaction B does not wait for transaction A to commit or rollback. Transaction B immediately receives query results that do not include the deleted row. Setting DB2_RR_TO_RS effectively changes locking behavior, thus avoiding deadlocks.

"COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource" could not be found for data source ([data-source-name])"

This error usually occurs when the classpath of the DB2 JDBC driver is set correctly to `${DB2_JDBC_DRIVER_PATH}/db2java.zip` but the environment variable DB2_JDBC_DRIVER_PATH is not set.

Confirm that this is the problem on the **Manage WebSphere Variables** panel. Select **Environment** to verify that there is no entry for the variable DB2_JDBC_DRIVER_PATH.

To correct this problem, add the variable DB2_JDBC_DRIVER_PATH with **value** equal to the directory path containing the `db2java.zip` file.

ERROR CODE: 20001 and SQL STATE: HY000 accessing SQLServer database

The problem might be that the Distributed Transaction Coordinator service is not started, if you see an error similar to the following when attempting to access an SQL Server database:

```
ERROR CODE: 20001
SQL STATE:  HY000
java.sql.SQLException: [Microsoft][SQLServer JDBC Driver]
[SQLServer]xa_open (0) returns -3
at com.microsoft.jdbc.base.BaseExceptions.createException(Unknown Source) ...
at com.microsoft.jdbcx.sqlserver.SQLServerDataSource.getXAConnection
(Unknown Source) ...
```

To confirm that this is the problem, in the Windows **Control Panel > Services** (or the **Control Panel > Administrative Tools > Services**) window, verify whether the service **Distributed Transaction Coordinator** or **DTC** is started. If not, it might be the cause of the problem.

To resolve this problem, start the Distributed Transaction Coordinator service.

Application fails with message stating "Cannot find stored procedure..." accessing an SQLServer database

One possible cause for this error is that the Stored Procedures for JTA feature was not installed on the Microsoft SQL Server.

To correct the problem, repeat the installation for the Stored Procedures for JTA feature, according to the ConnectJDBC installation guide.

Unexpected IOException wrapped in SQLException, accessing Cloudscape database

This problem can occur because Cloudscape databases use a large number of files. Some operating systems, such as Sun Solaris, limit the number of files an application can open at one time. If the default is a low number, such as 64, you can get this exception.

If your operating system lets you configure the number of file descriptors, you can correct the problem by setting the number to a high value, such as 1024.

"select for update" causes table lock and deadlock when accessing Cloudscape

If a **select for update** operation on one row locks the entire table, which creates a deadlock condition, the cause can be that you have not defined indexes on that table. Lack of an index on the columns you use in the **where** clause can cause Cloudscape to create a table lock rather than a row level lock.

To resolve this problem, create an index on the affected table.

ERROR XSDB6: Another instance of Cloudscape may have already booted the database "database"

This problem is caused by the fact that running Cloudscape embedded framework allows only one JVM to access the database instance at a time.

To resolve this problem:
1. Ensure that you do not have other JDBC client programs, such as **ij** or **cview** running on that database instance, when WebSphere Application Server is running.
2. Ensure that you do not use the same instance of the database for more than one data source.
3. Or, use the networkServer framework, which doesn’t have this limitation.

Error ″The version of the IBM Universal JDBC driver in use is not licensed for connectivity to Cloudscape databases″

ERROR XSDB6: Another instance of Cloudscape may have already booted the database "database"

This problem is caused by the fact that running Cloudscape embedded framework allows only one JVM to access the database instance at a time.

To resolve this problem:
1. Ensure that you do not have other JDBC client programs, such as **ij** or **cview** running on that database instance, when WebSphere Application Server is running.
2. Ensure that you do not use the same instance of the database for more than one data source.
3. Or, use the networkServer framework, which doesn’t have this limitation.

Error "The version of the IBM Universal JDBC driver in use is not licensed for connectivity to Cloudscape databases"

At the client runtime, an error similar to the following occurs:

```
The version of the IBM Universal JDBC driver in use is not
licensed for connectivity to Cloudscape databases.  To connect
to this DB2 server, please obtain a licensed copy of the IBM DB2
Universal Driver for JDBC and SQLJ.  An appropriate license file
db2jcc_license_*.jar for this target platform must be installed to
the application classpath.  Connectivity to Cloudscape databases is
enabled by any of the following license files:
{ db2jcc_license_c.jar, b2jcc_license_cu.jar, db2jcc_license_cisuz.jar }
```

The problem occurs because an incorrect JDBC driver jar file name is specified in the classpath for JDBC provider. For example, the jar file name may have an extra '_', as follows:

```
${UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license__cu.jar
```

To resolve the problem, correct the UNIVERSAL_JDBC_DRIVER_PATH jar file name in the JACL script. Restart the cluster and rerun the client.

Running an application causes a runtime exception which produces an unreadable message.

At client runtime, you may receive a message similar to the following: `Caused by: com.ibm.db2.jcc.a.SqlException: DB2 SQL error: SQLCODE: -1, SQLSTATE: 42X05, SQLERRMC: ANNUITYHOLDER20^T42X05`

The problem occurs because the property ″retrieveMessagesfromServerOnGetMessage″, which is required by WebSphere Application Server, has not been set.

To resolve the problem, on the admininstrative console
1. select **Resources -> JDBC Providers**
2. Click on a Cloudscape provider
3. Scroll down and click on **Data Sources**
4. Select your data source (or add a new one)
5. Scroll down and select **Custom Properties**
6. If the property *retrieveMessagesFromServerOnGetMessage* already exists, set its value to true. If the property does not exist, select **New**and add the property *retrieveMessagesFromServerOnGetMessage* with a value **true**
7. Rerun the client

The SystemOut.log will now generate readable messages so that you can resolve the underlying problem.

"Sybase Error 7713: Stored Procedure can only be executed in unchained transaction mode" error

This error occurs when either:
• The JDBC attempts to put the connection in **autocommit(true)** mode.
• A stored procedure is not created in a compatible mode.

To fix the **autocommit(true)** mode problem, let the application change the connection to chained mode using **Connection.setAutoCommit(false)**, or use a **set chained on** language command.

To resolve the stored procedure problem, use this command, **sp_procxmode *procedure_name*, ″anymode″**.

"JZ0XS: The server does not support XA-style transactions. Please verify that the transaction feature is enabled and licensed on this server."

This error occurs when XA-style transactions are attempted on a server that does not have Distributed Transaction Management (DTM) installed.

To correct this problem, use the instructions in the Sybase Manual titled: *Using Adaptive Server Distributed Transaction Management Features* to enable Distributed Transaction Management (DTM). The main steps in this procedure are:
1. Install the DTM option.
2. Check the `license.dat` file to verify that the DTM option was installed.
3. Restart the license manager.
4. Enable DTM in ISQL.

5. Restart the ASE service.

A Container Managed Persistence (CMP) enterprise bean is causing exceptions

This error is caused by improper use of reserved words. Reserved words cannot be used as column names.

To correct this problem, rename the variable to remove the reserved word. You can find a list of reserved words in the *Sybase Adaptive Server Enterprise Reference Manual; Volume 1: Building Blocks*, Chapter 4. This manual is available online at: http://manuals.sybase.com/onlinebooks/group-as/asg1250e/refman.

IllegalConnectionUseException

One possible reason for this error is that a connection obtained from a WAS40DataSource is being used on more than one thread. This is a violation of the J2EE 1.3 programming model, and an exception is generated when it is detected on the server. This problem occurs for users accessing a data source through servlets or Bean Managed Persistence (BMP) type enterprise beans.

To confirm that this is the problem, examine the code for sharing of connections. Code can inadvertently cause sharing by not following the programming model recommendations, for example by storing a connection in an instance variable in a servlet, which can cause the connection to be used on multiple threads at the same time.

WTRN0062E: An illegal attempt to enlist multiple one phase capable resources has occurred

Possible causes of this error include:

- An attempt to share a single phase connection, when each **getConnection** method has different connection properties; such as the AccessIntent. This causes the connection to be created as non-shareable.
- An attempt to have more than one unshareable connection participate in a global transaction, when the data source is not an XA resource.
- An attempt to have a one phase resource participate in a global transaction while an XA resource or another one phase resource has already participated in this global transaction.
  - Within the scope of a global transaction you try to get a connection more than once and at least one of the resource-refs you are using specifies that the connection is unshareable, and the data source is not configured to support two-phase commit transactions. It does not support an XAResource. If you do not use a resource-ref, you default to unshareable connections.
  - Within the scope of a global transaction you try to get a connection more than once and at least one of the resource-refs you are using specifies that the connection is shareable and the data source is not configured to support two-phase commit transactions. That is, it does not support an XAResource. In addition, even though you specify that connections should be shareable, each getConnection request is made with different connection properties (such as IsolationLevel or AccessIntent). In this case, the connections are not shareable, and multiple connections are handed back.
  - Multiple components (Servlets, Session Beans, BMP Entity Beans, or CMP Entity Beans) are accessed within a global transaction. All use the same DataSource, all specify shareable connections on their resource-refs, and you expect them to all share the same connection. If the properties are different, as stated above, you get multiple connections. AccessIntent settings on CMP beans change their properties. To share a connection, the AccessIntent setting must be the same. For more information about CMP beans sharing a connection with non-CMP components, see the *Data access application programming interface support* and *Example: Accessing data using IBM extended APIs to share connections between container-managed and bean-managed persistence beans* topics in the DataAccess section of the InfoCenter.

To correct this error:

- Check what your client code passes in with its **getConnection** requests, to ensure they are consistent with each other.
- Check the connection sharing scope from the resource binding, using the Application Assembly Tool (AAT).
  - If you are running an unshareable connection scope, ensure that your data source is an XA data source.
  - If you are running a shareable connection scope, ensure that all connection properties, including AccessIntent and other properties (such as userid), are sharable.
- Check the JDBC provider **implementation** class from the **Manage JDBC resource** panel of the administrative console to ensure that it is a class that supports XA-type transactions.

ConnectionWaitTimeoutException accessing a data source or resource adapter

If your application receives a com.ibm.websphere.ce.cm.ConnectionWaitTimeoutException or com.ibm.websphere.ce.j2c.ConnectionWaitTimeoutException when attempting to access a WebSphere Application Server data source or JCA-compliant resource adapter, respectively, some possible causes are:

- The maximum number of connections for a given pool is set too low. The demand for concurrent use of connections is greater then the configured maximum for the connection pool. One indication that this is the problem is that you receive these exceptions regularly, but your CPU utilization is not high. This indicates that there are too few connections available to keep the threads in the server busy.
- Connection Wait Time is set too low. Current demand for connections is high enough such that sometimes there is not an available connection for short periods of time. If your connection wait timeout value is too low, you may timeout shortly before a user returns a connection back to the pool. Adjusting the connection wait time may give you some relief. One indication that this is the problem is that you are using near the maximum number of connections for an extended period and receiving this error regularly.
- You are not closing some connections or are returning connections back to the pool at a very slow rate. This can easily happen when using unshareable connections, when you forget to close them, or you close them long after you are finished using them, thus keeping the connection from being returned to the pool for reuse. The pool soon becomes empty and all applications get ConnectionWaitTimeoutExceptions. One indication that this is the problem is that the connection pool has become starved and you receive this error on most requests.
- You are driving more load than the server or backend system have resources to handle. In this case you must determine which resources you need more of and upgrade configurations or hardware to address the need. One indication that this is the problem is that the application or database server CPU is nearly 100% busy.

To correct these problems, modify an application to use fewer connections or properly close the connections, change the pool settings of MaxConnections or ConnnectionWaitTimeout, or adjust resources and their configuration.

com.ibm.websphere.ce.cm.StaleConnectionException: [IBM][CLI Driver] SQL1013N The database alias name or database name "NULL" could not be found. SQLSTATE=42705

This error occurs when a data source has been defined but the databaseName attribute and corresponding value have not been added to the ″custom properties″.

To add the databaseName property:

1. Expand the **Resources->Manage JDBC Providers** link in the administrative console.
2. Select the JDBC Provider which supports the problem data source.
3. Select **Data Sources** and then select the problem data source.
4. Under **additional properties** select **Custom Properties**.
5. Select the **databaseName** property, or add one if it does not exist, and enter the actual database name as the value.
6. Click **Apply** or **OK**, and then **Save** from the action bar.

7. Try to access the data source again.

java.sql.SQLException: java.lang.UnsatisfiedLinkError:

This error indicates that the directory containing the binary libraries which support a database are not included in the LIBPATH environment variable for the environment in which the WebSphere Application Server is started.

The path containing the DBM vendor's libraries vary by dbm. One way to find them is by scanning the for missing library specified in the error message. Then the LIBPATH variable can be corrected to include the missing directory, either in the `.profile` of the account from which WebSphere Application Server is executed, or by adding a statement in a `.sh` file which then executes the ″`startServer`″ program.

"J2CA0030E: Method enlist caught java.lang.IllegalStateException" wrapped in error "WTRN0063E: An illegal attempt to enlist a one phase capable resource with existing two phase capable resources has occurred" when attempting to execute a transaction.

This error can occur when Last Participant Support (LPS) is missing or disabled. LPS allows a one-phase capable resource and a two-phase capable resource to be enlisted within the same transaction.

LPS is only available if the following are true:
- WebSphere Application Server Programming Model Extensions (PME), which is included in the Application Server Enterprise product) is installed.
- The option ″Additional Enterprise Extensions″ was enabled when PME was installed. If you perform a typical installation, this is be enabled by default. If you perform a custom installation, you have the option to disable this function, which would disable LPS.
- The application enlisting the one phase resource has been deployed with the **Accept heuristic hazard** option enabled. This is done through the Application Assembly Tool. To enable this option in the Application Assembly Tool:
  1. Load the `EAR` file into the Application Assembly Tool.
  2. If the `EAR` file is actually a JTEE1.2 `EAR` then it must be upgraded to a JTEE1.3 `EAR` by selecting **`File-> Convert EAR`** from the Application Assembly Tool.
  3. Select the `EAR` file in the left-hand panel of the Application Assembly Tool.
  4. Select the **WAS Enterprise** tab in bottom right-hand window panel of the Application Assembly Tool.
  5. Ensure that the **Accept heuristic hazard** option is selected.
  6. Save the `EAR` file.

java.lang.UnsatisfiedLinkError:xaConnect exception when attempting a database operation

This problem has two main causes:
- The most common cause is that the jdbc driver which supports connectivity to the database is missing, or is not the correct version, or that native libraries which support the driver are on on the system's path.
  – To resolve this problem on a Windows platform, ensure that the JDBC driver jar file is on the system PATH environment variable:
    - If you are using DB2, ensure that at least the DB2 client product has been installed on the WebSphere host
      - On DB2 version 7.2 or earlier, the file where the client product is installed on the WebSphere Application Server is db2java.zip. Ensure that the usejdbc2.bat program has been executed after the database install and after any upgrade to the database product.
      - On DB2 version 8.1 or later, it is recommended that you use the ″DB2 Universal JDBC Provider Driver″ when defining a JDBC provider in WebSphere Application Server. The driver file is db2jcc.jar. If you use the ″type 2″ (default) option, ensure that at least the DB2 client product is

installed on the WebSphere host. If you specify the ″type 4″ option, the DB2 client does not need to be installed, but the file db2jcc.jar still must be present.

When specifying the location of the driver file, it is recommended to that you specify the path and file name of the target DB2 installation, rather than simply copying the file to a local directory, if possible. Otherwise, you may be exposed to problems if the target DB2 installation is upgraded and the driver used by WebSphere Application Server is not. If you choose ″DB2 Legacy CLI-based type 2 JDBC Driver″ when defining a JDBC provider in WebSphere Application Server, then you must still follow the steps to ensure that you have the correct version of the db2java.zip file (see instructions for DB2 7.2 or earlier).

– On a Unix platform, ensure that any native libraries required to support the database client of your database product are specified in the LD_LIBRARY_PATH environment variable in the profile of the account under which WebSphere Application Server executes.

  If you are using DB2 The native library is libdb2jdbc.so. The best way to ensure that this library is accessed correctly by WebSphere is to call the db2profile script supplied with DB2 from the .profile script of the account (such as ″root″) under which WebSphere runs.

  - If you are using DB2 version 7.2 or earlier, ensure that the usejdbc2,script provided with DB2 is called from the profile of the account under which WebSphere Application server is launched.
  - If you are using DB2 version 8.1 or later, see the previous instructions for the Windows operating system.

- If the database manager is DB2, you may have chosen the option to create a 64-bit instance. A 64-bit configuration is not supported. If this has happened, remove the database instance and create a new one with the default 32-bit setting.

Note: For general help in configuring JDBC drivers and data sources in WebSphere Application Server, see the topic Accessing data from applications.

Cannot access an enterprise bean from a servlet, JSP file, stand-alone program, or other client

What kind of error are you seeing?

- **javax.naming.NameNotFoundException: Name *name* not found in context ″local″ message** when access is attempted
- **BeanNotReentrantException** is thrown
- **CSITransactionRolledbackException / TransactionRolledbackException** is thrown
- Call fails, Stack trace beginning **EJSContainer E Bean method threw exception [exception_name]** found in JVM log file.
- Call fails, **ObjectNotFoundException or ObjectNotFoundLocalException** when accessing stateful session EJB found in JVM log file.
- Attempt to start CMP EJB module fails with **javax.naming.NameNotFoundException: *dataSourceName***
-
- **Message BBOT0003W is issued**
- Symptom: **CNTR0001W: A Stateful SessionBean could not be passivated**

If the client is remote to the enterprise bean, which means, running in a different application server or as a stand-alone client, browse thelogs of the application server hosting the enterprise bean as well as log files of the client.

ObjectNotFoundException or ObjectNotFoundLocalException when accessing stateful session EJB

A possible cause of this problem is that the stateful session bean timed out and was removed by the container. This event must be coded for, according to the EJB 2.0 specification (available at `http://java.sun.com/products/ejb/docs.html`), section 7.6.2, Dealing with exceptions.

Stack trace beginning "EJSContainer E Bean method threw exception [exception_name]" found in JVM log file

If the ″exception name″ indicates an exception thrown by an IBM class, that is it begins ″com.ibm...″, then search for the exception name within this InfoCenter, and in the online help as described below. If ″exception name″ indicates an exception thrown by your application, contact the application developer to determine what might have caused it.

javax.naming.NameNotFoundException: Name name not found in context "local"

A possible reason for this is exception is that the enterprise bean is not local (not running in the same Java Virtual Machine [JVM] or Application Server) to the client JSP, servlet, Java application, or other enterprise bean, yet the call is to one of the enterprise bean's ″local″ interface methods. If access worked in a development environment but not when deployed to WebSphere Application Server, for example, it could be that the enterprise bean and its client were in the same JVM in development, but after deployment they are in separate processes.

To resolve this problem, contact the developer of the enterprise bean and determine whether the client call is to a method in the enterprise bean's local interface. If so, have the client code changed to call a remote interface method, or promote the local method into the remote interface.

References to enterprise beans with local interfaces are bound in a name space local to the server process with the URL scheme of `local:`.

BeanNotReentrantException is thrown

This problem can be caused by client code (typically a servlet or JSP) attempting to call the same stateful SessionBean from two different client threads. This situation often arises when the an application stores the reference to the stateful session bean in a static variable, uses a global (static) JSP variable to refer to the stateful SessionBean reference, or stores the stateful SessionBean reference in the HTTP session object and then has the client browser issue a new request to the servlet or JSP before the previous request has completed.

To resolve this problem, ask the developer of the client code to review their code for these conditions.

CSITransactionRolledbackException / TransactionRolledbackException is thrown

These are high-level exceptions thrown by an enterprise bean's container, and indicate that an enterprise bean call could not be successfully completed. When this exception is thrown, logs to determine the underlying cause.

Some possible causes are:
- The enterprise bean may throw an exception that was not declared as part of its method signature. The container is required to roll back the transaction in this case. Common causes of this situation are where the enterprise bean or code that it calls throws a NullPointerException, ArrayIndexOutOfBoundsException, or other Java ″runtime″ exception, or where a BMP bean encounters a JDBC error. The resolution is to investigate the enterprise bean code and resolve the underlying exception, or to add the exception to the problem method's signature.
- A transaction may attempt to do additional work after being placed in a ″`Marked Rollback`″, ″`RollingBack`″, or ″`RolledBack`″ state. Transactions cannot continue to do work after they have been set to one of these states. Often this occurs because the transaction has timed out which, in turn, often occurs because of a database deadlock. The resolution is to work with the application's database managements tools or administrator to determine whether database transactions called by the enterprise bean are timing out.
- A transaction may fail on commit due to ″dangling work″. This could be due to ″local″ transactions. The local transaction encounters some ″dangling work″ during commit. When a local transactions

encounters an ″unresolved action″ the default ″action″ is to ″rollback″. This can be adjusted to ″commit″ in the Application Assembly Tool. In the Application Assembly Tool (AAT), open the enterprise bean `.jar` file (or the `EAR` file containing the enterprise bean) and select the ″Session Beans″ or ″Entity Beans″ object in the component tree on the left. The ″Unresolved Action″ property is on the ″IBM Extensions″ tab of the container properties.

Attempt to start EJB module fails with "javax.naming.NameNotFoundException dataSourceName_CMP"exception

The possible causes of this problem are:
- When the DataSource resource was configured, Container Managed Persistence was not selected.
  - To confirm that this is the problem, in the administrative console, browse the properties of the data source given in the NameNotFoundException. On the Configuration panel, look for a checkbox labeled **Container Managed Persistence**.
  - To correct this problem, select checkbox for **Container Managed Persistence**.
- If Container Managed Persistence is selected, it is possible that the CMP DataSource could not be bound into the namespace.
  - Look for additional naming warnings or errors in the status bar, and in the hosting application server's logs. Check any further naming-exception problems that you find by looking at the topic Cannot access an object hosted by WebSphere Application Server (enterprise bean, connection pool, etc) from a servlet, JSP, stand-alone program , or other client.

Message BBOT0003W is issued

Message BBOT0003W indicates a transaction timeout. The timeout may result in the abnormal termination of the servant where the transaction was executing.
- The default timeout value for enterprise bean transactions is 120 seconds. After this time, the transaction times out and the connection is closed.
- If the transaction legitimately takes longer than the specified timeout period, on the Administrative console:
  1. Go to **Manage Application Servers -> server_name**
  2. Select the **Transaction Service properties** page
  3. Increase the **Total transaction lifetime timeout** value
  4. **Save** the configuration

Note: z/OS will use the value you set for **Total transaction lifetime timeout** as the default transaction timeout setting. If you set a value for this property that is greater than the Maximum transaction timeout value, z/OS will use the Maximum transaction timeout value as the default.

Symptom:CNTR0001W: A Stateful SessionBean could not be passivated

This error can occur when a Connection Object being used in the bean has not been closed or nulled out.

To confirm that this is the problem, look for an exception stack in the logs for the EJB Container which hosts the enterprise bean, which looks similar to:

```
StatefulPassi  W  CNTR0001W:
A Stateful SessionBean could not be passivated: StatefulBeanO
(BeanId(XXX#YYY.jar#ZZZZ),
state = PASSIVATING)
java.io.NotSerializableException:  com.ibm.ws.rsadapter.jdbc.WSJdbcConnection
  at  java.io.ObjectOutputStream.outputObject((Compiled  Code))
  at  java.io.ObjectOutputStream.writeObject(ObjectOutputStream.java(Compiled  Code))
  at  java.io.ObjectOutputStream.outputClassFields((Compiled  Code))
  at  java.io.ObjectOutputStream.defaultWriteObject((Compiled  Code))
  at  java.io.ObjectOutputStream.outputObject((Compiled  Code))
```

```
 at java.io.ObjectOutputStream.writeObject(ObjectOutputStream.java(Compiled Code))
 at com.ibm.ejs.container.passivator.StatefulPassivator.passivate((Compiled Code))
 at com.ibm.ejs.container.StatefulBeanO.passivate((Compiled Code)
 at com.ibm.ejs.container.activator.StatefulASActivationStrategy.atUnitOfWorkEnd
                                  ((Compiled Code))
 at com.ibm.ejs.container.activator.Activator.unitOfWorkEnd((Compiled Code))
 at com.ibm.ejs.container.ContainerAS.afterCompletion((Compiled Code)
```

where XXX,YYY,ZZZ is the Bean's name.

To correct this problem, the application must close all connections and set the reference to null for all connections. Typically this is done in the `ejbPassivate()` method of the bean. See the enterprise bean specification mandating this requirement, specifically section 7.4 in the EJB specification version 2.0. Also, note that the bean must be coded to reacquire these connections when the bean is reactivated. Otherwise, there will be NullPointerExceptions when the application tries to reuse the connections.

Cannot look up an object hosted by WebSphere Application Server from a servlet, JSP file, or other client

To resolve problems encountered when a servlet, JSP file, stand-alone application or other client attempts to access an enterprise bean, ConnectionPool, or other named object hosted by WebSphere Application Server, you must first verify that the target server can be accessed from the client:
- From a command prompt on the client's server, enter ″`ping` *`server_name`*″ and verify connectivity.
- Use the WebSphere Application Server administrative console to verify that the target resource's application server and, if applicable, EJB module or Web module, is started.

Continue only if there is no problem with connectivity and the target resource appears to be running.

What kind of error are you seeing?
- NameNotFoundException from JNDI lookup operation
- CannotInstantiateObjectException from JNDI lookup operation
- Message NMSV0610I appears in the server's log file, indicating that some Naming exception has occurred
- OperationNotSupportedException from JNDI Context operation.
- ″WSVR0046E: Failed to bind″ error, with Original exception: ″org.omg.CosNaming.NamingContextPackage.AlreadyBound″.
- ConfigurationException from ″new InitialContext″ operation or from a JNDI Context operation with a URL name.
- ServiceUnavailableException from ″new InitialContext″ operation.
- CommunicationException thrown from a ″new InitialContext″ operation.
-

NameNotFoundException from JNDI lookup operation

If you encounter this exception in trying to access an enterprise bean, data source, messaging resource, or other resource:
- Browse the properties of the target object in the administrative console, and verify that the jndi name it specifies matches the JNDI name the client is using.
- If you are looking up an object that resides on a server different from the one from which the initial context was obtained, you must use the fully qualified name.
  - If access is from another server object such as a servlet accessing an enterprise bean and you are using the default context, not specifying the fully qualified JNDI name, you may get this error if the object is being hosted on a different server.
  - If access is from a stand-alone client, it may be that the object you are attempting access is on a server different from the server from which you obtained the initial context.

To correct this problem, use the fully-qualified JNDIname:
- If the object is in a single server: `cell/nodes/`*`nodeName`*`/servers/`*`serverName`*`/`*`jndiName`*. Objects are not supported in this release.
- If the object is on a server cluster: `cell/clusters/`*`clusterName`*`/`*`jndiName`*.

CannotInstantiateObjectException from JNDI lookup operation

If you encounter this exception in trying to access an enterprise bean, data source, messaging resource, or other resource, possible causes include:
- A serialized Java object is being looked up, but the necessary classes required to deserialize it are not in the runtime environment.
- A Reference object is being looked up, and the associated factory used to process it as part of the lookup processing is failing.

To determine the precise cause of the problem:
- Look in the logs of the server hosting the target resource. Look for exceptions immediately preceding the CannotInstantiateObjectException. If it is a java.lang.NoClassDefFoundError or java.lang.ClassNotFoundException, make sure the class referenced in the error message can be located by the class loader.
- Print out the stack trace for the root cause and look for the factory class. It will be called by `javax.naming.NamingManager.getObjectInstance()`. The reason for the failure will depend on the factory implementation, and may require you to contact the developer of the factory class.

Message NMSV0610I appears in the server's log file, indicating that some Naming exception has occurred

This error is informational only and is provided in case the exception is related to an actual problem. Most of the time, it is not. If it is, the log file should contain adjacent entries to provide context.
- If no problems are being experienced, ignore this message. Also ignore the message if the problem you are experiencing does not appear to be related to the exception being reported and if there are no other adjacent error messages in the log.
- If a problem is being experienced, look in the log for underlying error messages.
- The information provided in message NMSV0610I can provide valuable debug data for other adjacent error messages posted in response to the Naming exception that occurred.

OperationNotSupportedException from JNDI Context operation

This error has two possible causes:
- An update operation, such as a bind, is being performed with a name that starts with `"java:comp/env"`. This context and its subcontexts are read-only contexts.
- A Context bind or rebind operation of a non-CORBA object is being performed on a remote name space that does not belong to WebSphere Application Server. Only CORBA objects can be bound to these CosNaming name spaces.

To determine which of these errors is causing the problem, check the full exception message.

WSVR0046E: Failed to bind, ejb/jndiName: ejb/jndiName. Original exception : org.omg.CosNaming.NamingContextPackage.AlreadyBound

This error occurs two enterprise bean server applications were installed on the same server such that a binding name conflict occurred. That is, a jndiName value is the same in the two applications' deployment descriptors. The error will surface during server startup when the second application using that jndiName value is started.

To verify that this is the problem, examine the deployment descriptors for all enterprise bean server applications running in the server in search for a jndiName that is specified in more than one enterprise bean application.

To correct the problem, change any duplicate jndiName values to ensure that each enterprise bean in the server process is bound with a different name.

ConfigurationException from "new InitialContext" operation or from a JNDI Context operation with a URL name

If you are attempting to obtain an initial JNDI context, a configuration exception can occur because an invalid JNDI property value was passed to the InitialContext constructor. This includes JNDI properties set in the System properties or in some jndi.properties file visible to the class loader in effect. A malformed provider URL is the most likely property to be incorrect. If the JNDI client is being run as a thin client such that the `CLASSPATH` is set to include all of the individual jar files required, make sure the `.jar` file containing the properties file `com/ibm/websphere/naming/jndiprovider.properties` is in the `CLASSPATH`.

If the exception is occurring from a JNDI Context call with a name in the form of a URL, the current JNDI configuration may not be set up properly so that the required factory class name cannot be determined, or the factory may not be visible to the class loader currently in effect. If the name is a Java: URL, the JNDI client must be running in a J2EE client or server environment. That is, the client must be running in a container.

Check the exception message to verify the cause.

If the exception is being thrown from the InitialContext constructor, correct the property setting or the `CLASSPATH`.

If the exception is being thrown from a JNDI Context method, make sure the property `java.naming.factory.url.pkgs` includes the package name for the factory required for the URL scheme in the name. URL names with the Java scheme can only be used while running in a container.

ServiceUnavailableException from "new InitialContext" operation

This exception indicates that some unexpected problem occurred while attempting to contact the name server to obtain an initial context. The ServiceUnavailableException, like all NamingException objects, can be queried for a root cause. Check the root cause for more information. It is possible that some of the problems described for CommunicationExceptions may also result in a ServiceUnavailableException.

Since this exception is triggered by an unexpected error, there is no probable cause to confirm. If the root cause exception does not indicate what the probable cause is, investigate the possible causes listed for CommunicationExceptions.

CommunicationException thrown from a "new InitialContext" operation

The name server identified by the provider URL cannot be contacted to obtain the initial JNDI context. There are many possible causes for this problem, including:
- The host name or port in the provider URL is incorrect.
- The host name cannot be resolved into an IP address by the domain name server, or the IP address does not match the IP address which the server is actually running under.
- A firewall on the client or server is preventing the port specified in the provider URL from being used.

To correct this problem:
- Make sure the provider URL and the network configurations on the client and server machines are correct.

- Make sure the host name can be resolved into an IP address which can be reached by the client machine. You can do this using the ping command.
- If you are running a firewall, make sure that use of the port specified in the provider URL will be allowed.

Errors or access problems after enabling security

What kind of error are you seeing?

- I cannot access part or all of administrative console or use wsadmin after enabling security
- I cannot access a Web page after enabling security
- The client cannot access an enterprise bean after enabling security
- The client never gets prompted when accessing a secured enterprise bean
- I cannot stop an application server, node manager, or node after enabling security
- Error Message: SECJ0314E: Current Java 2 Security policy reported a potential violation
- ″MSGS0508E: The JMS Server security service was unable to authenticate userid:″ error displayed in SystemOut.log when starting an application server
- ″SECJ0237E: One or more vital LTPAServerObject configuration attributes are null or not available″ after enabling security and starting application server.
- After enabling single sign-on, I cannot log on to the administrative console.

Can't access part or all of admin console or use wsadmin after enabling security

- If you cannot access the administrative console, or view or update certain objects, look in the logs of the application server which hosts the administrative console page for a related error message.
- You may not have authorized your ID for administrative tasks. This is indicated by errors such as:
  - **[8/2/02 10:36:49:722 CDT] 4365c0d9 RoleBasedAuth A SECJ0305A: Role based authorization check failed for security name MyServer/myUserId, accessId MyServer/S-1-5-21-882015564-4266526380-2569651501-1005 while invoking method getProcessType on resource Server and module Server.**
  - Exception message: ″**ADMN0022E: Access denied for the getProcessType operation on Server MBean**″
  - **When running the command: wsadmin -username j2ee -password j2ee: WASX7246E: Cannot establish ″SOAP″ connection to host ″BIRKT20″ because of an authentication failure. Please ensure that user and password are correct on the command line or in a properties file.**
- To grant an ID administrative authority:
  - From the Administrative Console, select **System Administration -> Console Users** and validate that the ID is a member. If it is not, add the ID with at least monitor access privileges, for read-only access.
- Check that the *enable_trusted_application* flag is set to true:
  - From the Administrative Console, select **Security -> Global Security -> Custom Properties -> Enable Trusted Application** and check that it is set to true.

Can't access a web page after enabling security

When secured resources cannot be accessed, causes include:

- Authentication errors - WebSphere Application Server security cannot identify the ID of the person or process. Symptoms of authentication errors include:
  - Netscape browser:
    - ″Authorization failed. Retry?″ message displayed after an attempt to login.
    - Allows any number of attempts to retry login and displays ″Error 401″ message when ″cancel″ is pressed to stop retry.
    - Typical browser message: ″Error 401: Basic realm='Default Realm'″.
  - Internet Explorer browser:
    - Login prompt displayed again after an attempt to login.
    - Allows 3 attempts to retry login.
    - Displays ″Error 401″ message after 3 unsuccessful retries.

- Authorization errors - security has identified the requesting person or process as not authorized to access the secured resource. Symptoms of authorization errors include:
  – Netscape browser: ″Error 403: AuthorizationFailed″ message is displayed.
  – Internet Explorer:
    - ″You are not authorized to view this page″ message is displayed.
    - ″HTTP 403 Forbidden″ error is also displayed.
- SSL errors - WebSphere Application Server security uses Secure Socket Layer (SSL) technology internally to secure and encrypt its own communication, and incorrect configuration of the internal SSL settings can cause problems. Also you might have enabled SSL encryption for your own Web application or enterprise bean client traffic which, if configured incorrectly, can cause problems regardless of whether WebSphere Application Server security is enabled.
  – SSL related problems are often indicated by error messages which contain a statement such as: `ERROR: Could not get the initial context or unable to look up the starting context.Exiting.` followed by `javax.net.ssl.SSLHandshakeException`
  – System SSL failures are indicated by error messages like the following:

```
Trace: 2003/08/11 19:53:40.682 01 t=9D49C0 c=UNK key=S2 (0E012048)
   Description: Failure Exit from -> SecurityManager::secureSocketInit
   Socket descriptor: 211
   gsk_secure_socket_init: 410
```

Authentication error accessing a Web page

Possible causes for authentication errors include:

Username or passwords invalid
Check the username and password and make sure they are correct.

Security configuration error : User registry type is not set correctly.
Check the user registry property in global security settings in the administrative console. Ensure that it is the intended user registry.

Internal program error
If the client application is a Java stand-alone program, it might not be gathering or sending credential information correctly.

Authorization error accessing a Web page

If a user who should have access to a resource does not, there is probably a missing configuration step. Review the steps for securing and granting access to resources.

Specifically:
- Check required roles for the accessed Web resource.
- Check the authorization table to make sure that the user, or the groups to which the user belongs, is assigned to one of the required roles.
- You can view required roles for the Web resource in the deployment descriptor of the Web resource.
- You can also view the authorization table for the application that contains the Web resource, using the administrative console.
- Test with a user who is granted the required roles, to see if the user can access the problem resources.
- If the problem user is required to be granted one or more of the required roles, use the administrative console to assign that user to required roles. Then stop and restart the application.

Cannot access an enterprise bean after enabling security

If client access to an enterprise bean fails after security is enabled:
- Review the steps for securing and granting access to resources.
- Browse the server's logs for errors relating to enterprise bean access and security. Look up any errors in the message table.

- Errors similar to **Authorization failed for /UNAUTHENTICATED while invoking *resource* securityName:/UNAUTHENTICATED;accessId:UNAUTHENTICATED not granted any of the required roles *roles*** indicate that:
  - an unprotected servlet or JSP accessed a protected enterprise bean. When unprotected servlet is accessed, the user is not prompted to login and hence the servlet runs as UNAUTHENTICATED. When it makes a call to an enterprise bean that is protected it will fail.

    To resolve this problem, secure the servlet that is accessing the secured enterprise bean. Make sure the servlet's `runAs` property is set to an ID that can access the enterprise bean.
  - An unauthenticated Java client program is accessing an enterprise bean resource that is protected. This can happen if the file read by the `sas.client.props` properties file used by the client program does not have the `securityEnabled` flag set to `true`.

    To resolve this problem, make sure that the `sas.client.props` file on the client side has its securityEnabled flag set to true.
- Errors similar to **Authorization failed for *valid_user* while invoking *resource* securityName:/username;accessId:xxxxxx not granted any of the required roles *roles*** indicate that a client attempted to access a secured enterprise bean resource, and the supplied user ID is not assigned the required roles for that enterprise bean.
  - Check the required roles for the enterprise bean resource being accessed. Required roles for the enterprise bean resource can be viewed in the deployment descriptor of the Web resource.
  - Check the authorization table and make sure that the user or the group that the user belongs to is assigned one of the required roles. The authorization table for the application that contains the enterprise bean resource can also be viewed using administrative console.
- If you are using LOCALOS and SAFAuthorization, check the SAF EJBROLEs setup. Specifically, make sure that
  - the EJBROLE class has been activated
  - The role has been defined to SAF
  - The userid has been permitted to the role and the class was refreshed after the permit operation.

Client program never gets prompted when accessing secured enterprise bean

Even though it appears security is enabled and an enterprise bean is secured, it may happen that the client executes the remote method without getting prompted.
- If the remote method is protected, you should get an authorization failure. Otherwise,
- Execute the method as an unauthenticated user.

Possible reasons for this include:
- The server you are communicating with may not have security enabled. Check with the WebSphere Application Server administrator to ensure that the server security is enabled. This is done in the global security settings from within the Security section of the administrative console.
- The client does not have security enabled in the `sas.client.props` file. Edit the `sas.client.props` to ensure the property `com.ibm.CORBA.securityEnabled=true`.
- The client does not have a ConfigURL specified. Ensure that the property `com.ibm.CORBA.ConfigURL` is specified on the command line of the Java client, using the -D parameter.
- The specified ConfigURL has an invalid URL syntax or the `sas.client.props` pointed to by it cannot be found. Ensure that the property **`com.ibm.CORBA.ConfigURL`** is valid, for example, similar to `file:/WebSphere/AppServer/properties/sas.client.props` on Windows systems. Check the Java documentation for a description of URL formatting rules. Also, validate that the file exists at the specified path.

Cannot stop an application server, node manager, or node after enabling security

If you are using command line utilities to stop WebSphere Application Server processes, you need to apply additional parameters after enabling security, in order to provide authentication and authorization information.

Error Message: SECJ0314E: Current Java 2 Security policy reported a potential violation on server

If you find errors on your server similar to:

```
Error Message: SECJ0314E: Current Java 2 Security policy reported a
potential violation of Java 2 Security Permission. Please refer to Problem
Determination Guide for further information.
{0}Permission\:{1}Code\:{2}{3}Stack Trace\:{4}Code Base Location\:{5}
```

then The Java Security Manager `checkPermission()` method has reported a `SecurityException`.

The reported exception may be critical to the secure system. Turn on security trace to determine the potential code that may have violated the security policy. Once the violating code is determined, you should verify if the attempted operation is permitted with respect to Java 2 Security, by examining all applicable Java 2 security policy files and the application code itself.

A more detailed report is enabled by either configuring RAS trace into debug mode, or specifying a Java property.

- Specify the following property in the **Application Servers >** ***server name*** **> ProcessDefinition > Java Virtual Machine** panel from the administrative console in the **Generic JVM arguments** panel:
  - add the runtime flag **`java.security.debug`**
  - Valid values:

    access
    to print all debug information including: required permission, code, stack, and code base location.

    stack to print debug information including: required permission, code, and stack.

    failure to print debug information including: required permission and code.

For a review of Java security policies and what they mean , see the Java 2 Security documentation at `http://java.sun.com/j2se/1.3/docs/guide/security/index.html` .

Note: If the application is running with Java Mail, this message may be benign. You can update the *installed Enterprise Application root*/`META-INF/was.policy` file to grant the following permissions to the application:

- `permission java.io.FilePermission ″${user.home}${/}.mailcap″, ″read″;`
- `permission java.io.FilePermission ″${user.home}${/}.mime.types″, ″read″;`
- `permission java.io.FilePermission ″${java.home}${/}lib${/}mailcap″, ″read″;`
- `permission java.io.FilePermission ″${java.home}${/}lib${/}mime.types″, ″read″;`

"MSGS0508E: The JMS Server security service was unable to authenticate userid:" error displayed in SystemOut.log when starting an application server

This error may be a result of installing the JMS (messaging API) sample and then enabling security. You can follow the instructions in the Configure and Run page of the corresponding JMS sample documentation to configure the sample to work with WebSphere Application Server security.

You can verify the installation of the message-driven bean sample by launching the installation program, selecting **Custom**, and browsing the components which are already installed in the **Select the features you like to install** panel. The JMS sample is shown as **Message-Driven Bean Sample**, under **Embedded Messaging**.

You can also verify this by using the administrative console to open the properties of the application server which contains the samples, selecting ″MDBSamples″ and clicking ″uninstall″.

SECJ0237E: One or more vital LTPAServerObject configuration attributes are null or not available

The most likely cause of this error is that LTPA is selected as authentication mechanism but the LTPA keys have not been generated. The LTPA keys are used for encrypting the LTPA token.

To resolve this problem:
1. Select **System Administration -> Console users -> LTPA**
2. Enter a password, which can be anything.
3. Enter the same password in ″Confirm Password″.
4. Click **Apply**.
5. Click **Generate Keys**.
6. Click on **Save**.

After enabling single sign-on, I cannot log on to the administrative console

This problem occurs when single sign-on (SSO) is enabled, and you attempt to access the administrative console using the short name of the server, for example `http://myserver:9090/admin`. The server will accept your userID and password, but returns you to the sign-on page instead of the administrative console.

To correct this problem, use the fully-qualified hostname of the server, for example `http://myserver.mynetwork.mycompany.com:9090/admin`.

Errors after enabling Secure Sockets Layer, or Secure Sockets Layer-related error messages

If you are unable to access resources using a Secure Sockets Layer (SSL) type URL (beginning with ″https:″), or encounter error messages which indicate SSL problems, ensure that your HTTP server has been configured correctly for SSL by browsing the welcome page of the HTTP server using SSL by entering the URL **https://***hostname*.

If the page works with HTTP, but not HTTPS, the problem is in the HTTP server.
- Refer to the documentation for your HTTP server for instructions on correctly enabling SSL. If you are using the IBM HTTP Server or Apache, go to:http://www.ibm.com/software/webservers/httpservers/library.html. Select the link *Frequently Asked Questions*, and the topic *SSL*.

If the HTTP server handles SSL-encrypted requests successfully, or is not involved (for example, traffic flows from a Java client application directly to an enterprise bean hosted by the WebSphere Application Server, or the problem appears only after enabling WebSphere Application Server security), what kind of error are you seeing?

System SSL

See *z/OS System Secure Sockets Layer Programming SC24-5901* for information on using the System Secure Sockets Layer (SSL) callable services programming interfaces.
- javax.net.ssl.SSLHandshakeException - The client and server could not negotiate the desired level of security. Reason: handshake failure
- javax.net.ssl.SSLHandshakeException - The client and server could not negotiate the desired level of security. Reason: unknown certificate
- javax.net.ssl.SSLHandshakeException - The client and server could not negotiate the desired level of security. Reason: bad certificate
- error when programmatically creating a credential.

javax.net.ssl.SSLHandshakeException - The client and server could not negotiate the desired level of security. Reason: handshake failure

If you see a Java exception stack similar to: **[Root exception is org.omg.CORBA.TRANSIENT: CAUGHT_EXCEPTION_WHILE_CONFIGURING_SSL_CLIENT_SOCKET: JSSL0080E: javax.net.ssl.SSLHandshakeException - The client and server could not negotiate the desired level of security. Reason: handshake failure:host=MYSERVER,port=1079 minor code: 4942F303 completed: No] at com.ibm.CORBA.transport.TransportConnectionBase.connect (TransportConnectionBase.java:NNN)**, some possible causes are:
- Not having common ciphers between the client and server.
- Not specifying the correct protocol.

To correct these problems:
- Review the SSL settings by browsing the **WebSphere Administrative Console Security Settings -> SSL Configuration Repertoires -> DefaultSSLSettings** (or other named SSL settings), then selecting the **Secure Sockets Layer (SSL)** option from the **Additional Properties** menu. You can also browse the file manually by viewing: *install_dir*/properties/sas.client.props.
- Check the property specified by **com.ibm.ssl.protocol** to determine which protocol is specified.
- Check the cipher types specified by **com.ibm.ssl.enabledCipherSuites**. You may want to add more cipher types to the list. To see which cipher suites are currently enabled, go to the properties page of the SSL settings as described above, and look for the **Cipher Suites** property. To see the list of all possible cipher suites, go to the properties page of the SSL settings as described above, then view the online help for that page. From the help page, click **Configure additional SSL settings**.
- Correct the protocol or cipher problem by using a different client or server protocol and/or cipher selection. Typical protocols are SSL or SSLv3.

javax.net.ssl.SSLHandshakeException: unknown certificate

If you see a Java exception stack similar to: **ERROR: Could not get the initial context or unable to look up the starting context. Exiting. Exception received: javax.naming.ServiceUnavailableException: A communication failure occurred while attempting to obtain an initial context using the provider url: ″corbaloc:iiop:localhost:2809″. Make sure that the host and port information is correct and that the server identified by the provider url is a running name server. If no port number is specified, the default port number 2809 is used. Other possible causes include the network environment or workstation network configuration. [Root exception is org.omg.CORBA.TRANSIENT: CAUGHT_EXCEPTION_WHILE_CONFIGURING_SSL_CLIENT_SOCKET: JSSL0080E: javax.net.ssl.SSLHandshakeException - The client and server could not negotiate the desired level of security. Reason: unknown certificate:host=MYSERVER,port=1940 minor code: 4942F303 completed: No]**, it may be caused by not having the server's personal certificate in the client truststore.

To correct this problem:
- Check the client truststore to determine if the signer certificate from the server personal certificate is there. For a self-signed server personal certificate, the signer certificate is the public key of the personal certificate. For a CA signed server personal certificate, the signer certificate is the root CA certificate of the CA which signed the personal certificate.
- Add the server signer certificate to the client truststore.

javax.net.ssl.SSLHandshakeException: bad certificate

If you see a Java exception stack similar to **ERROR: Could not get the initial context or unable to look up the starting context. Exiting. Exception received: javax.naming.ServiceUnavailableException: A communication failure occurred while attempting to obtain an initial context using the provider url: ″corbaloc:iiop:localhost:2809″. Make sure that the host and port information is correct and that the server identified by the provider url is a running name server. If no port number is specified, the default port number 2809 is used.Other possible causes include the network environment or**

workstation network configuration. [Root exception is org.omg.CORBA.TRANSIENT: CAUGHT_EXCEPTION_WHILE_CONFIGURING_SSL_CLIENT_SOCKET: JSSL0080E: javax.net.ssl.SSLHandshakeException - The client and server could not negotiate the desired level of security. Reason: bad certificate: host=MYSERVER,port=1940 minor code: 4942F303 completed: No], it can be caused by having a personal certificate in the client keystore used for SSL mutual authentication but not having extracted the signer certificate into the server truststore so that the server could trust it whenever the SSL handshake is made.

To verify this, check the server truststore to determine if the signer certificate from the client personal certificate is there. For a self-signed client personal certificate, the signer certificate is the public key of the personal certificate. For a CA signed client personal certificate, the signer certificate is the root CA certificate of the CA which signed the personal certificate.

To correct this problem, add the client signer certificate to the server truststore.

Errors in messaging (JMS API)

What kind of problem are you seeing?
- javax.jms.JMSException: MQJMS2008: failed to open MQ queue in JVM log.

javax.jms.JMSException: MQJMS2008: failed to open MQ queue in JVM log

This error can occur when the MQ queue name is not defined in the Internal JMS Server Queue Names List. This can occur if a WebSphere Application Server Queue Destination is created, without adding the Queue Name to the internal JMS Server Queue Names List.

To resolve this problem:
- Open the WebSphere Application Server Administrative Console.
- Click **Servers > Manage Application Servers >** *server_name*> **Server Components > JMS Servers**.
- Add the Queue Name to the list.
- Save the changes and restart the server.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Errors returned to client trying to send a SOAP request

What kind of problem are you seeing?
- SOAPException: faultCode=SOAP-ENV:Client; msg=Error opening socket; java.net.ConnectException: Connection refused: connect
- javax.security.cert.CertPathBuilderException: No end-entity certificate matching the selection criteria could be found.

If none of these errors match the one you see:
- Browse the application server logs.
- Look up any error or warning messages in the message table.

SOAPException: faultCode=SOAP-ENV:Client; msg=Error opening socket; java.net.ConnectException: Connection refused: connect

The most likely cause of this refused connection is that it was sent to the default port, 80, and an HTTP server is not installed or configured.

To verify this situation, send the message directly to the SOAP port. For example, to **`http://`*`hostname`*`:9080`**. If this works, there are two ways to resolve the problem:

- Continue specifying port 9080 on SOAP requests.
- If an HTTP server such as the IBM HTTP Server, iis, IPlanet, or others, is not installed, install one and then step through the WebSphere Application Server installation to install the associated plug-in component.
- If an HTTP server is installed:
  - Regenerate the HTTP plug-in configuration in the administrative console by clicking **Environment > Update WebServer Plugin**, and restart the HTTP server.
  - If the problem persists, view the HTTP server access and error logs, as well as the *install_dir*/logs/http_plugin.log file for more information.

javax.security.cert.CertPathBuilderException: No end-entity certificate matching the selection criteria could be found

This error usually indicates that new or updated security keys are needed. The security key files are:

- SOAPclient
- SOAPserver
- sslserver.p12

In an installed application, these files are located in: *install_dir*/installedApps/*application_name*.ear/soapsec.war/key/. After replacing these files, you must stop and restart the application.

To replace these files in a SOAP-enabled application that has not yet been installed:

- Expand *application_name*.ear.
- Expand soapsec.war.
- Replace the security key files in the key/ directory.
- After you have replaced these files, install the application and restart the server.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Errors connecting to WebSphere MQ and creating WebSphere MQ queue connection factory

The following exception may occur when trying to create the MDBListener instance:

```
6/23/03 22:45:58:232 CDT] 673106a8 MsgListenerPo W WMSG0049E:
Failed to start MDB PSSampleMDB against listener port SamplePubSubListenerPort
[6/23/03 22:47:58:289 CDT] 673106a8 FreePool E J2CA0046E:
Method createManagedConnctionWithMCWrapper caught an exception
during creation of the ManagedConnection for resource
JMS$SampleJMSQueueConnectionFactory, throwing ResourceAllocationException.
Original exception:  javax.resource.spi.ResourceAdapterInternalException:
 createQueueConnection failed
com.ibm.mq.MQException: MQJE001: An MQException occurred:
Completion Code 2, Reason 2009
MQJE003: IO error transmitting message buffer at
com.ibm.mq.MQManagedConnectionJ11.(MQManagedConnectionJ11.java:239)
```

This problem occurs because the MQ manager userid does not have write access to the /tmp directory. To correct this problem, before you use a Jacl procedure to configure WebSphere Application Server resources and install an application:

1. Ensure that all applications have write access to /tmp directory. Use the chmod 1777 command on the directory if necessary.

2. Create another subdirectory under `/tmp` (for example, `/tmp/mydir`). Use this directory as a ″working directory″ for the Jacl.
3. Restart the server.

Applications that use messaging on startup should start successfully.

Troubleshooting application runtime and management problems

Select the problem you are having with running or managing deployed code for WebSphere Application Server:
- I have problems bringing up or using the administrative console.
- I have problems starting or using the **wsadmin** command prompt.
- My Web module or application server dies or hangs.
- I get errors trying to configure and enable security.
- I cannot seem to distribute the workload across clustered servers.
- There are problems setting up the multiserver Deployment Manager environment.
- I cannot uninstall or remove a node or application server.
- I have problems creating or using HTTP sessions.
- I have problems using tracing, logging, logfiles, or other troubleshooting features.
- I get errors connecting to the administrative console from a Netscape browser.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Problems starting or using the wsadmin command

What kind of problem are you having?
- ″WASX7023E: Error creating ″SOAP″ connection to host″ or similar error trying to launch wsadmin command line utility.
- ″com.ibm.bsf.BSFException: error while eval'ing Jacl expression: no such method ″<command name>″ in class com.ibm.ws.scripting.AdminConfigClient″ returned from wsadmin command.
- WASX7022E returned from running ″wsadmin -c ...″ command, indicating invalid command.
- com.ibm.ws.scripting.ScriptingException: WASX7025E: String ″″ is malformed; cannot create ObjectName.

If you do not see your problem here:
- If you are not able to enter wsadmin command mode, try running **wsadmin -c** ″**\$Help wsadmin**″ for help in verifying that you are entering the command correctly.
- If you can get the wsadmin command prompt, enter **$Help help** to verify that you are using specific commands correctly.
- wsadmin commands are a superset of Jacl (Java Command Language), which is in turn a Java-based implementation of the Tcl command language. For details on Jacl syntax beyond wsadmin commands, refere to the Tcl developers' site, http://www.tcl.tk. For specific details relating to the Java implementation of Tcl, refer to http://www.tcl.tk/software/java.
- Browse the *install_dir*/logs/wsadmin.traceout file for clues.
  – Keep in mind that wsadmin.traceout is refreshed (existing log records are deleted) whenever a new wsadmin session is started.
  – If the error returned by wsadmin does not seem to apply to the command you entered, for example, you receive WASX7023E, stating that a connection could not be created to host ″myhost,″ but you did not specify ″-host myhost″ on the command line, examine the properties files used by wsadmin to determine what properties are specified. If you do not know what properties files were loaded, look for the WASX7326I messages in the wsadmin.traceout file; there will be one of these messages for each properties file loaded.

"WASX7023E: Error creating "SOAP" connection to host" or similar error trying to launch wsadmin command line utility

By default, the wsadmin utility attempts to connect to an application server at startup. This is because some commands act upon running application servers. This error indicates that no connection could be established.

To resolve this problem:
- If you are not sure whether an application server is running, start it by entering **startserver.sh *server short name*** from the command prompt. If the server is already running, you will see an error similar to ″ADMU3027E: An instance of the server is already running″.
- If you are running a z/OS configuration, you will first need to start the deployment manager by issuing the following command from a command prompt on the MVS console:

  ```
  START dmgr_proc_name,JOBNAME=server_short_name,
        ENV=cell_short_name.node_short_name.server_short_name
  ```

  Note: This command must be entered on a single line. It is split here for display purposes.

  Then you can launch wsadmin immediately to connect to the deployment manager, or start a node and application server to connect to.
- If an application server is running and you still get this error:
  - If you are running remotely (that is, on a different machine from the one running WebSphere Application Server), you must use the **-host *hostname*** option to the wsadmin command to direct wsadmin to the right physical server.
  - If you are using the -host option, try pinging the server machine from the command line from the machine on which you are trying to launch wsadmin to verify there are no issues of connectivity such as firewalls.
  - verify that you are using the right port number to connect to the WebSphere Application Server process:
    - If you are not specifying a port number (using the -port option) when you start wsadmin, wsadmin uses the default port specified in *install_dir*/properties/wsadmin.properties, property name=com.ibm.ws.scripting.port (default value =8879).
    - The port that wsadmin should send on depends on the server process wsadmin is trying to connect to.

      For a single-server installation, wsadmin attempts to connect to the application server process by default. To verify the port number:
      - Look in the file *install_dir*/config/cells/*node_name*/nodes/*node_name*/serverindex.xml for a tag containing the property **serverType=″APPLICATION_SERVER″**.
      - Look for an entry within that tag with the property **endPointName=″SOAP_CONNECTOR_ADDRESS″**.
      - Look for a **port** property within that tag. This is the port wsadmin should send on.

      In a Network Deployment installation, wsadmin launched from the bin directory on the Network Deployment installation attempts to send requests to the deployment manager by default. To verify the port number:
      - Get the hostname of the node on which the Deployment Manager is installed.
      - Using that hostname, look in *install_dir*/config/cells/*node_name*Network/nodes/*node_name*Manager/serverindex.xml for a tag containing the property **serverType=″DEPLOYMENT_MANAGER″**.
      - Within that tag, look for an entry with a property **endPointName=″SOAP_CONNECTOR_ADDRESS″**.
      - Within that tag, look for a ″port″ property. This is the port wsadmin should send on.

"com.ibm.bsf.BSFException: error while eval'ing Jacl expression: no such method "<command name>" in class com.ibm.ws.scripting.AdminConfigClient" returned from wsadmin command.

This error is usually caused by a misspelled command name. Use the **$AdminConfig help** command to get information about what commands are available. Note that command names are case-sensitive.

WASX7022E returned from running "wsadmin -c ..." command, indicating invalid command

If the command following -c appears to be valid, the problem may be caused by the shell attempting to do variable substitution. Variable substitution can occur on Unix System Services if wsadmin -c invokes a command that is enclosed in double quotes and includes dollar signs. To confirm that this is the problem, check the command to see if it contains an unescaped dollar sign, for example: **wsadmin -c ″$AdminApp install ....″**.

To correct this problem, escape the dollar sign with a backslash. For example: **wsadmin -c ″\$AdminApp install ...″**.

Note: When the command is enclosed in single quotes, the shell does not attempt to do variable substitution. Therefore, you do not need to escape the dollar sign. Example: **wsadmin.sh -c '$AdminApp install ...'**

com.ibm.ws.scripting.ScriptingException: WASX7025E: String "" is malformed; cannot create ObjectName

One possible cause of this error is that an empty string was specified for an object name. This can happen if you use one scripting statement to create an object name and the next statement to use that name, perhaps in an ″invoke″ or ″getAttribute″ command, but you don't check to see if the first statement really returned an object name. For example (the following samples use basic Jacl commands in addition to the wsadmin Jacl extensions to make a sample script):

```
#let's misspell ″Server″
set serverName [$AdminControl queryNames type=Server,*]
$AdminControl getAttributes $serverName
```

To correct this error, make sure that object name strings have values before using them. For example:

```
set serverName[$AdminControl queryNames type=Server,*]
if {$serverName == ″″} {puts ″queryNames returned empty - check query argument″}
else {$AdminControl getAttributes $serverName}
```

For details on Jacl syntax beyond wsadmin commands, refer to the Tcl developers' site, http://www.tcl.tk.

Web module or application server dies or hangs

If an application server dies (its process spontaneously closes), or freezes (its Web modules stop responding to new requests):

- Isolate the problem by installing Web modules on different servers, if possible.
- To detect memory leak problems, enable verbose garbage collection on the application server. This feature adds detailed statements to the JVM error log file of the application server about the amount of available and in-use memory. To set up verbose garbage collection:
  1. Select **Servers > Application Servers >** ***server_name*** **> Process Definition > Java Virtual Machine**, and enable **Verbose Garbage Collection**.
  2. Stop and restart the application server.
  3. Periodically, or after the application server stops, browse the log file for garbage collection statements. Look for statements beginning with ″allocation failure″. The string indicates that a need for memory allocation has triggered a JVM garbage collection (freeing of unused memory).

Allocation failures themselves are normal and not necessarily indicative of a problem. The allocation failure statement is followed by statements showing how many bytes are needed and how many are allocated.

If there is a steady increase in the total amount of free and used memory (the JVM keeps allocating more memory for itself), or if the JVM becomes unable to allocate as much memory as it needs (indicated by the bytes needed statement), there might be a memory leak.

- If the verbose garbage collection output indicates that the application server is running out of memory, one of the following problems might be present:
  - There is a memory leak in application code that you must address. To pinpoint the cause of a memory leak, enable the **RunHProf** function in the Servers > Application Servers > server_name > Process Definition > Java Virtual Machine pane of the problem application server:
    - In the same JVM pane, set the **HProf Arguments** field to a value similar to `depth=20,file=heapdmp.txt`. This value shows exception stacks to a maximum of 20 levels, and saves the heapdump output to the *install_root*`/bin/heapdmp.txt` file.
    - Save the settings.
    - Stop and restart the application server.
    - Re-enact the scenario or access the resource that causes the hang or crash, if possible. Stop the application server. If this is not possible, wait until the hang or crash happens again and stop the application server.
    - Examine the file into which the heapdump was saved. For example, examine the *install_root*`/bin/heapdmp.txt` file:
      - Search for the string, ″SITES BEGIN″. This finds the location of a list of Java objects in memory, which shows the amount of memory allocated to the objects.
      - The list of Java objects occurs each time there was a memory allocation in the JVM. There is a record of what type of object the memory instantiated and an identifier of a trace stack, listed elsewhere in the dump, that shows the Java method that made the allocation.
      - The list of Java object is in descending order by number of bytes allocated. Depending on the nature of the leak, the problem class should show up near the top of the list, but this is not always the case. Look throughout the list for large amounts of memory or frequent instances of the same class being instantiated. In the latter case, use the ID in the `trace stack` column to identify allocations occurring repeatedly in the same class and method.
      - Examine the source code indicated in the related trace stacks for the possibility of memory leaks.
  - The default maximum heap size of the application server needs to be increased.
- If an application server spontaneously dies, there will be an SDUMP. See Using the Error Dump and Cleanup interface for instructions on how to analyze the dump.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Errors when trying to configure or enable security

What kind of error are you seeing?

- ″LTPA password not set. validation failed″ message displayed as error in the Administrative Console after enabling global security.
- ″Validation failed for user [userid]. Please try again...″ displayed in the Administrative Console when enabling global security.
- The `setupClient.bat` or `setupClient.sh` file is not working correctly
- ″Java HotSpot(TM) Server VM warning: Unexpected Signal 11 occurred under user-defined signal handler 0x7895710a″ message occurs in the `native_stdout.log` file when enabling security on the HP-UX11i platform

- If you have successfully configured security (made changes, saved the configuration, and enabled security with no errors), but are now having problems accessing Web resources or the administrative console, refer to Errors or access problems after enabling security.

For general tips on diagnosing and resolving security-related problems, see the topic Troubleshooting the security component.

If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

"LTPA password not set. validation failed" message displayed as error in the Administrative Console after saving global security settings

This error can be caused if, when configuring WebSphere Application Server security, ″LTPA″ is selected as the authentication mechanism, and the LTPA password field is not set. To resolve this problem:
- Select Security **Authentication Mechanism > LTPA** in the console left-hand navigation pane.
- Complete the password and confirm password fields.
- Click **OK**.
- Try setting Global Security again.

"Validation failed for user userid. Please try again..." displayed in the Administrative Console after saving global security settings

This typically indicates that a setting in the User Registry configuration is not valid:
- If the user registry is LocalOS, it's likely that either the server userid and password is invalid or the server userid does not have ″Act As Part of the Operating System″ (for NT) or root authority (for Unix). It needs this authority in order to access the LocalOS user registry to authenticate.
- If the user registry is Lightweight Directory Access Protocol (LDAP):
  – Any of the settings that enable WebSphere Application Server to communicate with LDAP might be invalid, such as the LDAP server's userid, password, host, port, or LDAP filter. When you select **Apply** or **OK** on the Global Security panel, a validation routine connects to the registry just as it would during runtime when security is enabled. This is done in order to verify any configuration problems immediately, instead of waiting until the server restarts.
  – If the BIND DN is required, you must specify a DN instead of a short name.
  – Sometimes the LDAP server might be down during configuration. The best way to check this is to issue a command line search - LdapSearch, to search for the server ID. This way you can determine if the server is running and if the server ID is a valid entry in the LDAP.
- If the user registry is Custom, double check that your implementation is in the classpath. Also, check to see if your implementation is authenticating properly.
- Regardless of registry type, check the User Registries configuration panels to see if you can find a configuration error:
  – Go back to the User Registries configuration panels and retype the password for the server ID.
- See if there is an obvious configuration error. Double check the attributes specified.

The setupClient.bat or setupClient.sh file is not working correctly

The `setupClient.bat` file on Windows platforms and the `setupClient.sh` file on UNIX platforms incorrectly specify the location of the SOAP security properties file.

In the `setupClient.bat` file, the correct location should be:

```
set CLIENTSOAP=-Dcom.ibm.SOAP.ConfigURL=file:%WAS_HOME%/properties/soap.client.props
```

In the `setupClient.sh` file, the CLIENTSOAP variable should be:

```
CLIENTSOAP=-Dcom.ibm.SOAP.ConfigURL=file:$WAS_HOME/properties/soap.client.props
```

In the `setupClient.bat` and `setupClient.sh` files, complete the following steps:

1. Remove the leading / after file:.
2. Change `sas` to `soap`.

Java HotSpot(TM) Server VM warning: Unexpected Signal 11 occurred under user-defined signal handler 0x7895710a message occurs in the `native_stdout.log` file when enabling security on the HP-UX11i platform

After you enable security on HP-UX 11i platforms, the following error in the `native_stdout.log` file occurs, along with a core dump and WebSphere Application Server does not start:

```
Java HotSpot(TM) Server VM warning:
Unexpected Signal 11 occurred under user-defined signal handler 0x7895710a
```

To work around this error, apply the fixes recommended by HP for Java at the following URL: http://www.hp.com/products1/unix/java/infolibrary/patches.html.

Workload not getting distributed

What kind of problem are you seeing?

- Web (HTTP) requests are not distributed to all servers.
- Enterprise bean requests are not distributed to all servers.
-
- A failing server still receives enterprise bean requests (failover fails).
-

If none of these problem solution descriptions fixes your problem:

1. Browse the JVM logs of the problem deployment manager and application servers:
   a. Look up any error messages by selecting the **Reference** view of the InfoCenter and expanding **Messages** in the navigation tree.
   b. If Java exceptions appear in the log files, try to determine the actual subcomponent directly involved in the problem by examining the trace stack and looking for a WebSphere Application Server-related class near the top of the stack (names beginning with `com.ibm.websphere` or `com.ibm.ws`) that threw the exception. If appropriate, review the steps for troubleshooting the appropriate subcomponent under the Troubleshooting by component: what is not working? topic.

      For example, if the exception appears to have been thrown by a class in the `com.ibm.websphere.naming` package, review the Naming Services Component troubleshooting tips topic.
2. Ensure that all the machines in your configuration have TCP/IP connectivity to each other by running the **ping** command:
   a. From each physical server to the Deployment Manager
   b. From the Deployment Manager to each physical server
3. Although the problem is happening in a clustered environment, the actual cause might be only indirectly related, or unrelated, to clustering. Investigate all relevant possibilities:
   a. If an enterprise bean on one or more servers is not serving requests, review the Cannot access an enterprise bean from a servlet, JSP, stand-alone program, or other client and Cannot access an object hosted by WebSphere Application Server from a servlet, JSP file, or other client topics.
   b. If problems seem to appear after enabling security, review the Errors or access problems after enabling security topic.
   c. If an application server stops responding to requests, or spontaneously dies (its process closes), review the Web module or application server dies or hangs topic.
   d. If SOAP requests are not being served by some or all servers, review the Errors returned to client trying to send a SOAP request topic.
4. Check to see if the problem is identified and documented by looking at available online support (hints and tips, technotes, and fixes).

Web (HTTP) requests are not distributed to all servers

If HTTP requests are not being distributed to all servers:

- Check your PrimaryServers list. The plug-in load balances across all servers that are defined in the PrimaryServers list, if affinity has not been established. If you do not have a PrimaryServers list defined, the plug-in load balances across all servers defined in the cluster, if affinity has not been established. In the case where affinity has been established, the plug-in should go directly to that server, for all requests within the same HTTP session.
- If some servers are servicing requests and one or more others are not, try accessing a problem server directly to verify that it works, apart from workload management issues. If that does not work:
  – Use the administrative console to ensure that the affected server is running.
  – See the article Web resource (JSP, servlet, html file, image, etc) will not display for more information.
- See the article HTTP plug-in component troubleshooting tips for more information.

EJB requests are not distributed to all servers

If a client cannot reach a server in a cluster thought to be reachable, a server might be marked unusable, or is down. To verify this:

- Use the administrative console to verify that the server is started. Try starting it, or if started, stop and restart it.
- Browse the administrative console and verify that the node that runs the server having the problem appears. If it does not:
  – Review the steps for adding a node to a cluster.
  – Review the steps in the One or more nodes do not show up in the administrative console topic.
- If possible, try accessing the enterprise bean directly on the problem server to see if there is a problem with TCP/IP connectivity, application server health, or other problem not related to workload management. If this fails, review the topic Cannot access enterprise bean from a servlet, JSP, stand-alone program , or other client.

A failing server still receives enterprise bean requests (failover fails)

Some possible causes of this problem are:

- The client might have been in a transaction with an enterprise bean on the server that went down. Check the JVM logs of the application server hosting the problem enterprise bean instance. If a request is returned with **CORBA SystemException COMM_FAILURE org.omg.CORBA.completion_status.COMPLETED_MAYBE**, this might be working as designed. The design is to let this particular exception flow back to the client, since the transaction might have completed. Failing over this request to another server could result in this request being serviced twice.
- If the requests sent to the servers come back to the client with any other exceptions consistently, it might be that no servers are available.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Errors setting up multiserver environments

What kind of problem are you seeing?

- After creating and starting a cluster, the cluster does not start, and logs show that servers in the cluster are not found.
- One or more nodes do not show up in the administrative console.

- The addNode command fails.
- Application files are not present on all nodes.
- After downloading the Network Deployment plug-in to my system, my server does not start.

If none of these problem solution descriptions fixes your problem:
1. Browse the logs of the problem deployment manager and application servers:
   a. Look up any error messages by selecting the **Reference** view of the InfoCenter and expanding **Messages** in the navigation tree.
   b. If Java exceptions appear in the log files, try to determine the actual subcomponent directly involved in the problem by examining the trace stack and looking for a WebSphere Application Server-related class near the top of the stack (names beginning with `com.ibm.websphere` or `com.ibm.ws`) that threw the exception. If appropriate, review the steps for troubleshooting the appropriate subcomponent in the Troubleshooting by component: what is not working? topic.

      For example, if the exception appears to have been thrown by a class in the `com.ibm.websphere.naming` package, review the Naming Services Component troubleshooting tips topic.
2. Ensure that all the machines in your configuration have TCP/IP connectivity to each other by running the **ping** command:
   a. From each physical server to the Deployment Manager
   b. From the Deployment Manager to each physical server
3. Although the problem is happening in a clustered environment, the actual cause might be only indirectly related, or unrelated, to clustering. Investigate all relevant possibilities:
   a. If an enterprise bean on one or more servers is not serving requests, review the Cannot access an enterprise bean from a servlet, JSP, stand-alone program, or other client and Cannot access an object hosted by WebSphere Application Server from a servlet, JSP file, or other client topics.
   b. If problems seem to appear after enabling security, review the Errors or access problems after enabling security topic.
   c. If an application server stops responding to requests, or spontaneously dies (its process closes), review the Web module or application server dies or hangs topic.
   d. If SOAP requests are not being served by some or all servers, review the Errors returned to client trying to send a SOAP request topic.
4. Check to see if the problem is identified and documented by looking at available online support (hints and tips, technotes, and fixes).

After creating and starting a cluster, the cluster does not start, and logs show that servers in the cluster are not found

This error can occur when the configuration is not synchronized from the deployment manager to a node. If ***auto synchronization*** is enabled, wait until the synchronization has had a chance to run. If you are using manual synchronization, explicitly request a sync to each node on the cluster.

To determine whether synchronization has taken place, look at the configuration on the node machines using the administrative console and verify that the new cluster members are defined on each node.

One or more nodes do not show up in the administrative console

This can occur when there is a basic connectivity problem between the deployment manager server and other servers in the topology. To determine whether this is the problem, look for the file`serverindex.xml` in the deployment manager directory structure.
- If the problem node does not appear in the list, review the steps for adding a node to the cluster.
- If the problem node does appear in the list:
  – From the deployment manager server, ping the server name as it appears in the list. If the ping command shows no communication, verify that the hostname is correct in the list, and correct it if necessary, then restart the deployment manager.
  – If the name that appears in the list is the short name, ping the fully qualified network name. If the corrected name works, update the list and restart the deployment manager.

- If the problem server uses Dynamic Host Configuration Protocol (DHCP), try replacing the logical hostname with the IP address and restart the deployment manager. If this resolves the problem, be aware that you must change `serverindex.xml` each time the problem server address changes, potentially each time the problem machine is rebooted. To avoid this problem, consider assigning a static IP address to the server.
- If you still cannot establish communication between the servers, contact your network administrator to resolve the problem, and restart the deployment manager after the problem is corrected.

The addNode command fails

This error can occur when the deployment manager Domain Name Server (DNS) configuration is set up improperly. The default installation on Linux uses the loopback address (127.0.0.1) as the default host address. To verify that this is the problem, query the hostname of the suspect machine. If it returns localhost 127.0.0.1, or if file transfer traces at the node show the node trying to upload files to a URL that includes 127.0.0.1, the node has an incorrect DNS configuration.

To correct this problem, update the `/etc/hosts` file or the name service configuration file, `/etc/nsswitch.conf`, to query the Domain Name Server or Network Information Server (NIS) before searching hosts.

Application files are not present on all nodes

In the WebSphere Application Server Network Deployment environment, application binary files are transferred to the individual nodes where applications are supported as part of the **node sync** operation. During node sync, application files are only propagated if their deployment descriptors specify **enableDistribution=true**. This flag is specified as part of the application installation procedure in the administrative console, and is stored as a property in the *install_root*/config/cells/*cell_name*/applications/*application_name*/deployment.xml file.

To confirm that this is the cause, check to see whether the enableDistribution flag is set. If it is already set to true, ensure that the target node is configured to run auto file synchronization.

If both of these settings are correct and the problem persists, manually perform an explicit synchronization. If the application files still do not appear in the installation directory, use the EARExpander tool (located in *install_root*/bin) to expand the EAR file from the repository to the installation destination. On remote nodes, the repository should appear in theconfig/cells/*cell_name*/applications/*application_name*.ear/ directory.

After downloading the Network Deployment plug-in to my system, my server does not start

If you experience this situation, the most likely cause is that the transport paths in the plug-in must be modified to work in your environment. See the Example: Manually editing transport settings in the server.xml file topic for information on how to modify these settings.

Cannot uninstall an application or remove a node or application server

What kind of problem are you having?
- After uninstalling an application through wsadmin tool, the application continues to run and throws ″DocumentIOException″
- The `removeNode` command does not remove the installed application from the deployment manager
- I cannot display the syntax for the `removeNode` command.

If none of these steps fixes your problem:
- Make sure that the application and its Web and EJB modules, are in a stopped state before uninstalling.

- If you are uninstalling or installing an application using **wsadmin**, make sure that you are using the **`-conntype NONE`** option to invoke **wsadmin** and enable local mode. To use the **`-conntype NONE`** option, stop the hosting application server before uninstalling the application.
- Check to see if the problem has been identified and documented by looking at the available online support (hints and tips, technotes, and fixes).

After uninstalling application through the wsadmin tool, the application throws "DocumentIOException"

If this exception occurs after the application was uninstalled using wsadmin with the `-conntype NONE` option:
- Restart the server or,
- Rerun the uninstall command without the `-conntype NONE` option.

The removeNode command does not remove the installed application from the deployment manager

If the applications were installed indirectly using the **addNode** program with the **-includeapps** option, then `removeNode` will not uninstall them, since they may be in use by other nodes. These applications must be explicitly uninstalled, for example through the administrative console.

I cannot display the syntax for the removeNode command

Unlike the `addNode` command, the `removeNode` command is valid with no parameters, so executing it will execute the operation, that is, remove the node, without displaying the command syntax.

To see the valid options for removeNode, execute `removeNode -?` or `removeNode -help`.

Problems creating or using HTTP sessions

Note: To view and update the Session Manager settings discussed here, use the administrative console. Select the application server that hosts the problem application, then under **Additional properties**, select **Web Container**, then **Session manager**.

What kind of problem are you having?
- HTTP Sessions are not getting created, or are lost between requests.
- HTTP Sessions are not persistent (session data lost when application server restarts, or not shared across cluster).
- Session is shared across multiple browsers on same client machine.
- Session is not getting invalidated immediately after specified Session timeout interval.
- Unwanted sessions are being created by jsps.

If your problem is not described here, or none of these steps fixes the problem:
-

HTTP Sessions are not getting created, or are lost between requests

By default, the Session Manager uses cookies to store the session ID on the client between requests. Unless you intend to avoid cookie-based session tracking, ensure that cookies are flowing between WebSphere Application Server and the browser:
- Make sure the **Enable cookies** checkbox is checked under the **Session tracking Mechanism** property.
- Make sure cookies are enabled on the browser you are testing from or from which your users are accessing the application.
- Check the Cookie domain specified on the SessionManager (to view the or update the cookie settings, in the **Session tracking mechanism->enable cookies** property, click **Modify**).

  – For example, if the cookie domain is set as ".myCom.com", resources should be accessed using that domain name, e.g. http://www.myCom.com/myapp/servlet/sessionservlet.
  – If the domain property is set, make sure it begins with a dot (.). Certain versions of Netscape do not accept cookies if domain name doesn't start with a dot. Internet Explorer honors the domain with or without a dot. For example, if the domain name is set to *mycom.com*, change it to *.mycom.com* so that both Netscape and Internet Explorer honor the cookie.
- Check the **Cookie path** specified on the SessionManager. Check whether the problem url is hierarchially below the Cookie path specified. If not correct the Cookie path.
- If the `Cookie maximum age` property is set, ensure that the client (browser) machine's date and time is the same as the server's, including the time zone. If the client and the server time difference is over the "Cookie maximum age" then every access would be a new session, since the cookie will "expire" after the access.
- If you have multiple web modules within an enterprise application that track sessions:
  – If you want to have different session settings among web modules in an enterprise application, ensure that each web module specifies a different cookie name or path, or
  – If Web modules within an enterprise application use a common cookie name and path, ensure that the HTTP session settings, such as Cookie maximum age, are the same for all Web modules. Otherwise cookie behavior will be unpredictable, and will depend upon which application creates the session. Note that this does not affect session data, which is maintained separately by Web module.
- Check the cookie flow between browser and server:
  1. On the browser, enable "cookie prompt". Hit the servlet and make sure cookie is being prompted.
  2. Access the session servlet from the browser.
  3. The browser will prompt for the cookie; note the jsessionid.
  4. Reload the servlet, note down the cookie if a new cookie is sent.
  5. Check the session trace and look for the session id and trace the request by the thread. Verify that the session is stable across web requests:
     – Look for **getIHttpsession(...)** which is start of session request.
     – Look for **releaseSesson(..)** which is end of servlet request.
- If you are using URL rewriting instead of cookies:
  – Ensure there are no static HTML pages on your application's navigation path.
- If you are using SSL as your session tracking mechanism:
  – Ensure that you have SSL enabled on your IBM HTTP Server or iPlanet HTTP server.
- If you are in a clustered (multiple node) environment, ensure that you have session persistence enabled.

HTTP Sessions are not persistent

If your HTTP sessions are not persistent, that is session data is lost when the application server restarts or is not shared across the cluster:
- Check the Datasource.
- Check the SessionManager's Persistence Settings properties:
  – If you intend to take advantage of Session Persistence, verify that Persistence is set to **Database**.
  – If you are using **Database-based persistence**:
    - Check the jndi name of the datasource specified correctly on SessionManager.
    - Specify correct userid and password for accessing the database.

      Note that these settings have to be checked against the properties of an existing Data Source in the admin console. The Session Manager does not automatically create a session database for you.
    - The Datasource should be non-JTA, i.e. non XA enabled.
    - Check the logs for appropriate database error messages.
    - With DB2, for row sizes other than 4k make sure specified row size matches the DB2 page size. Make sure tablespace name is specified correctly.

Session is shared across multiple browsers on same client machine

This behavior is browser-dependent. It varies between browser vendors, and also may change according to whether a browser is launched as a new process or as a subprocess of an existing browser session (for example by hitting Ctl-N on Windows).

The Cookie maximum age property of the Session Manager also affects this behavior, if cookies are used as the session-tracking mechanism. If the maximum age is set to some positive value, all browser instances share the cookies, which are persisted to file on the client for the specifed maximum age time.

Session is not getting invalidated immediately after specified Session timeout interval

The SessionManager invalidation process thread runs every *x* seconds to invalidate any invalid sessions, where x is determined based on the Session timeout interval specified in the Session manager properties. For the default value of 30 minutes , *x* is around 300 seconds. In this case, it could take up to 5 minutes (300 seconds) beyond the timeout threshold of 30 minutes for a particular session to become invalidated.

Unwanted sessions are being created by jsps

As required by the JavaServer Page specification, jsps by default perform a request.getSession(true), so that a session is created if none exists for the client. To prevent jsps from creating a new session, set the session scope to **false** in the jsp file using the page directive as follows:

<% @page session=″false″ %>

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

JSP source code shown by the Web server

Problem

If you share the document root of the WebSphere Application Server within the Web server document root, a security exposure can result as the Web server might display the JSP source file as plain text.

You can use the WebSphere Web server plug-in set of rules to determine whether a given request will be handled by the WebSphere Application Server. When an incoming request fails to match those rules, the Web server plug-in returns control to the Web server so that the Web server can fulfill the request. In this case, the unknown host header causes the Web server plug-in to return control to the Web server because the rules do not indicate that the WebSphere Application Server should handle it. Therefore, the Web server looks for the request in the Web server document root. Since the JSP source file is stored in the document root of the Web server, the Web server finds the file and displays it as plain text.

Suggested solution

Move the WebSphere Application Server JSP source file outside of the Web server document root. Then, when this request comes in with the unknown host header, the plug-in returns control to the Web server and the JSP source file is not found in the document root. Therefore, the Web server returns a `404 File Not Found` error rather than the JSP source file.

Problems using tracing, logging or other troubleshooting features

What kind of problem are you having?
- Netscape browser fails when trying to enable a component trace

Netscape browser fails when trying to enable a component trace

On systems using AIX, the Netscape browser fails when you try to enable trace on a component.

To work around this problem, do one of the following:
- Disable JavaScript on the browser and continue setting trace.
- Administer the AIX server from a remote machine running another browser and operating system.
- Change the trace manually in the *server.xml* file.

Errors connecting to the administrative console from a Netscape browser

What kind of problem are you having?
- Resizing the Netscape browser results in an error (nresize)
- Resizing the Netscape browser causes an error 404 message (404)
- Netscape screen blanks out while using the administrative console (152339)
- Resizing Netscape Version 4.7 causes errors (resizing47)
- Enabling Netscape Version 4.7 to display double-byte character set correctly (netscape479)
- Using Netscape Version 4.79 on a Solaris Operating Environment causes problems (479)
- Limitations occur when using Netscape with Solaris Operating Environment (nhas)
- Netscape browser must be capable of launching from a terminal window on all UNIX platforms (151829)

If you are able to bring up the browser page, but the console's behavior is inconsistent, error-prone, or unresponsive, try upgrading the browser you are using. Older browsers may not support the administrative console's features. For a listing of supported Web browsers, see http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html.

Resizing the Netscape browser results in an error (nresize)

If you resize your Netscape browser, you could get a ″Data Missing″ error. The error message disappears in 60 seconds.

Resizing the Netscape browser causes an error 404 message (404)

When connecting to the IBM WebSphere Application Server administrative console from a Netscape browser, resizing the browser can cause an error 404 message to occur. This situation occurs because the browser reloads the frame when resizing the window.

To avoid getting the error message, you can refrain from resizing the Netscape browser window, or you can connect to the IBM WebSphere Application Server administrative console using an Internet Explorer browser.

Netscape screen blanks out while using the administrative console (152339)

While working with the right-hand panel of the administrative console to do regular administrative tasks, the browser screen blanks out intermittently.

To work around this problem, do one of the following:
- After the problem occurs, close the Netscape browser, log in again, and continue working.
- Use the Internet Explorer browser from a Windows machine.

- Use Netscape 7.x, Mozilla 1.x, Opera 5, or Konquerer browsers on the platform, depending on which is available. Although there is not formal support for these browsers, they have all been used successfully with the product and in many cases work better than the previous 4.7.x series of Netscape browsers

Resizing Netscape Version 4.7 causes errors (resizing47)

You receive the following error messages when resizing Netscape Version 4.7:

```
Error 0
       An error occurred while processing request: http://localhost:9090/admin/upload.do
       message:
       Details
       com.ibm.webshpere.servlet.error.ServletErrorReport:
         at java.lang.Class.newInstance0(Native Method)
         ...
```

After resizing Netscape 4.7, Netscape has to reload the page just as it initially loads the page on the first request. For pages that do not expect POST data, it is not a problem. But for pages that do, Netscape 4.7 cannot retain the data.

Enabling Netscape Version 4.7 to display double-byte character set correctly (netscape479)

When using Netscape on AIX platforms without the translated package, the English version of Netscape is available for all locale environments as the default package. However, the English version of Netscape does not display double-byte character set (DBCS) characters on the browser radio buttons and title bars because the fonts are mismatched.

To work around this problem, you can install the message resource to make the translated version of Netscape available on a DBCS environment. Use the translated version of Netscape to display the corrupted DBCS correctly. Change the locale from English to the expected DBCS locale before starting Netscape. For example, issue the following commands to display the Japanese contents on Ja_JP (AIX Shift JIS locale):

```
$ export LANG=Ja_JP
  $ netscape&
```

Using Netscape Version 4.79 on a Solaris Operating Environment causes problems (479)

Using Netscape Version 4.79 on a Solaris Operating Environment to access the administrative console causes problems with some key text translations with the zh_TW.EUC locale. This situation is not a problem when you use Netscape Version 4.7. The officially supported version of Netscape on a Solaris Operating Environment is Version 4.79, but in this case the workaround is to use Netscape Version 4.7.

Limitations occur when using Netscape with Solaris Operating Environment (nhas)

If you click Troubleshooting > Logs and Trace >*server* > Diagnostic Trace > Modify, the window that pops up allowing you to select the Components and Groups to trace might not display a scroll bar, preventing you from viewing all the components and groups.

The text area displaying the selected components, groups, and trace levels does not have a vertical scroll bar. This omission is a limitation of Netscape on a Solaris Operating Environment.

To work around this problem, refresh the window to show the scroll bar.

Netscape browser must be capable of launching from a terminal window on all UNIX platforms (151829)

To make sure the Netscape browser can launch from a terminal window, edit or create a file called `profile` in the `/etc` directory and add the Netscape directory to the system path. For example:

```
PATH=$PATH:/opt/Netscape export PATH
```

Resolving timeout conditions

This file gives an overview of how to resolve 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.

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.

Understanding how timers work

This file gives an overview of understanding timers

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 later in this article; the specific timers themselves are described in Controlling behavior through timeout values. 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.

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 J2EE 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:
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:
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

This file gives an overview of how to enable and use the System Management Facilities (SMF) to collect and record system and job-related information.

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.

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

This file lists common timer variables and tools for monitoring these 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.

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.

General type of timer	Possible causes	Possible solutions
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.
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

This file lists common timer variables and tools for monitoring these timeout conditions

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.

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.	
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.

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:
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 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.	• 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.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

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 any of the various WebSphere console settings that controls the type of trace data collected.

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

All of these publications can be accessed from the IBM Publications Center.

Chapter 8. Troubleshooting by component: what is not working?

This section provides troubleshooting information based on the task you were trying to accomplish when the problem occurred. To find more information about your problem, select a task category from the list below.

Installation component troubleshooting tips

If you are having problems installing the WebSphere Application Server, follow these steps to resolve the problem:

- Browse the relevant log files for clues:
  - The main installation log file: `*install_dir*/log.txt`.
  - IBM Http Server log: `*install_dir*/ihs_log.txt`.
  - The log file produced when the default application `.ear` file is installed is: `*install_dir*/logs/installDefaultApplication.log`.
- Ensure that you have installed the correct level of dependent software, such as operating system version and revision level, by reviewing `http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html`.

If none of these steps solves the problem, check to see if the problem is identified and documented using the links in Diagnosing and fixing problems: Resources for learning.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Administration and administrative console troubleshooting tips

In WebSphere Application Server products, administrative functions are supported by:

- The application server (such as server1) process in the base product
- The deployment manager (dmgr) process in the Network Deployment product

The process must be running to use the administrative console. The **wsadmin** command line utility has a local mode that you can use to perform administrative functions, even when the server process is not running.

If you have problems starting or using the administrative console or wsadmin utility, verify that the supporting server process is started and that it is healthy.

- For the base product, look at these files:
  - `*install_root*/logs/server/startServer.log` for the message that indicates that the server started successfully: **ADMU3000I: Server server1 open for e-business; process id is *nnnn*.**.
  - the server log files for the message that indicates that the server started successfully: **WSVR0001I: Server *server* open for e-business**.
- For the Network Deployment product, look at these files:
  - `*install_root*/logs/dmgr/startServer.log` for the message that indicates that the server started successfully: **ADMU3000I: Server dmgr open for e-business; process id is *nnnn*.**.
  - the server log files for this message indicating that the server started successfully: **WSVR0001I: Server dmgr open for e-business**.
- For the z/OS product, check the job output.
- Look up any error messages in these files in the message reference table. Select the **Reference** view in the InfoCenter, then click **Messages**.

- A message like **WASX7213I: This scripting client is not connected to a server process** when trying to start wsadmin indicates that either the server process is not running, the host machine where it is running is not accessible, or that the port or server name used by wsadmin is incorrect.
- Verify that you are using the right port number to communicate with the administrative console or wsadmin server using the following steps:
  - Look in the the server log files.
  - The line **ADMC0013I: SOAP connector available at port *nnnn*** indicates the port that the server is using to listen for wsadmin functions.
  - The property **com.ibm.ws.scripting.port** in the `*install_root*/properties/wsadmin.properties` file controls the port used by wsadmin to send requests to the server. If it is different from the value shown in the the server log files, either change the port number in the `wsadmin.properties` file, or specify the correct port number when starting wsadmin by using the **-port *port_number*** property on the command line.
  - The message **SRVE0171I: Transport http is listening on port *nnnn* (default 9090)** indicates the port the server uses to listen for administrative console requests. If it is different than the one specified in the URL for the administrative console, change the URL in the browser to the correct value. The default value is http://localhost:9090/admin.
- Use the TCP/IP **ping** command to test that the hostname where the application server or deployment manager is executing, is reachable from the system where the browser or wsadmin program are being used. If you are able to ping the hostname, this indicates that there are no firewall or connectivity issues.
- If the host where the application server or deployment manager is running is remote to the machine from which the client browser or wsadmin command is running, ensure that the appropriate hostname parameter is correct:
  - The hostname in the browser URL for the console.
  - The **-host *hostname*** option of the wsadmin command that is used to direct wsadmin to the right server

If none of these steps solves the problem, see if the specific problem you are having is addressed in the Installation completes but the administrative console does not start topic. Check to see if the problem has been identified and documented using the links in the Diagnosing and fixing problems: Resources for learning topic.

For current information available from IBM Support on known problems and their resolution, see the following topics on the IBM support page:

- Administrative interface
- System management

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Application Assembly Tool troubleshooting tips

If you are having problems installing the WebSphere Application Server Application Assembly Tool (AAT), follow these steps:

- If a problem occurs using this component, the first thing to do is to enable the printing of messages and exceptions to the screen.
  - Modify the `assembly.bat` file located in the bin directory of the product installation. Change the statement "`start javaw`" to just "`java`".
  - Restart the AAT and a hanging command prompt window will appear through the lifetime of the Java process and display messages and exceptions.
  - Look up any error or warning messages you see in the message reference table.
- With a problem application open in the AAT, use the **Verify** menu command. This command will go through all components of the application and validate them for any XML errors or invalid entries such as missing fields, invalid bean or class references.

- To verify the integrity of an EAR (enterprise archive resource) file, expand it manually (outside of the AAT) by running the WebSphere Application Server `*install_root*\bin\EARExpander.bat` or `EARExpander.sh` file and supplying the name of the EAR file as a parameter. Browse the directory structure of the expanded EAR file to see if contains all the expected files.

  Here is an example using the Windows command prompt: `EARExpander -ear my.ear -expandDir c:\tmp\myear -operation expand`
- Contact the developer of the EAR file or its component files and ensure that they comply with J2EE specification level 1.3 and that any enterprise beans it contains conform to the EJB 2.0 Specification level.

If none of these steps solves the problem, check to see if the problem is identified and documented using the links in Diagnosing and fixing problems: Resources for learning.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Web Container troubleshooting tips

If you are having problems starting a Web module, or accessing resources within a particular Web module:
- For specific problems that can cause servlets, HTML files, and JavaServer Pages (JSP) files not to be served, seeWeb resource (JSP file, servlet, HTML file, image) does not display .

If none of these steps fixes your problem, check to see if the problem has been identified and documented by looking at the available online support (hints and tips, technotes, and fixes).

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

JDBC and data source troubleshooting tips

To see whether your specific problem has been addressed, review the Cannot access a data source topic.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

HTTP plug-in component troubleshooting tips

If you are having problems with the HTTP plug-in component - the component which sends requests from your HTTP server, such as IBM HTTP Server, Apache, Domino, iPlanet, or IIS, to the WebSphere Application Server, try these steps:
- Review the file *install_dir*`/logs/http_plugin.log` for clues. Look up any error or warning messages in the message table.
- Review your HTTP server's error and access logs to see if the HTTP server is having a problem:
  – IBM HTTP Server and Apache: `access.log` and `error.log`.
  – Domino web server: `httpd-log` and `httpd-error`.
  – iPlanet: `access` and `error`.

– IIS: *timedatestamp*`.log`.

If these files don't reveal the cause of the problem, follow these additional steps.

Plug-in Problem Determination Steps

The plug-in provides very readable tracing which can be beneficial in helping to figure out the problem. By setting the **LogLevel** attribute in the `config/plugin-cfg.xml` file to **Trace**, you can follow the request processing to see what is going wrong. At a high level:

1. The plug-in gets a request.
2. The plug-in checks the routes defined in the `plugin-cfg.xml` file.
3. It finds the server group.
4. It finds the server.
5. It picks the transport protocol, usually HTTP.
6. It sends the request.
7. It reads the response.
8. It writes it back to the client.

You can see this very clearly by reading through the trace for a single request:

- The first step is to determine if the plug-in has loaded into the HTTP server successfully.
  – Check to make sure the`http_plugin.log` has been created.
  – If it has, look in it to see if any error messages indicate some sort of failure that took place during plug-in initialization. If no errors are found look for the following stanza, which indicates that the plug-in started normally. Ensure that the timestamps for the messages correspond to the time you started the Web server:

```
[Thu Jul 11 10:59:15 2002] 0000009e 000000b1 - PLUGIN: ------------System Information----------
[Thu Jul 11 10:59:15 2002] 0000009e 000000b1 - PLUGIN: Bld date: Jul  3 2002, 15:35:09
[Thu Jul 11 10:59:15 2002] 0000009e 000000b1 - PLUGIN: Web server: IIS
[Thu Jul 11 10:59:15 2002] 0000009e 000000b1 - PLUGIN: Hostname = SWEETTJ05
[Thu Jul 11 10:59:15 2002] 0000009e 000000b1 - PLUGIN: OS version 4.0, build 1381, 'Service Pack 6'
[Thu Jul 11 10:59:15 2002] 0000009e 000000b1 - PLUGIN: -------------------------------------------
```

  – Some common errors are:

    lib_security: loadSecurityLibrary: Failed to load gsk library
    The GSKit did not get installed or the installation is corrupt. If the GSKit did not get installed you can determine this by searching for the file `gsk5ssl.dll` on all drives for Win32 or see if there are any `libgsk5*.so` files in `/usr/lib` on UNIX. Try reinstalling the plug-in to see if you can get the GSKit to install in order to fix this.

    ws_transport: transportInitializeSecurity: Keyring wasn't set
    The HTTPS transport defined in the configuration file was prematurely terminated and did not contain the Property definitions for the keyring and stashfile. Check your XML syntax for the line number given in the error messages that follow this one to make sure the Transport element contains definitions for the keyring and stashfiles before it is terminated.

  – If the`http_plugin.log` is not created, check the Web server error log to see if any plug-in related error messages have been logged there that indicate why the plug-in is failing to load. Typical causes of this can include failing to correctly configure the plug-in with the Web server environment. Check the documentation for configuring the Web server you are trying to use with the Web server plug-in.
- Determine whether there are network connection problems with the plug-in and the various application servers defined in the configuration. Typically you will see the following message when this is the case:

  ws_common: websphereGetStream: Failed to connect to app server, OS err=%d

  Where %d is an OS specific error code related to why the connect() call failed. This can happen for a variety of reasons.

  – Ping the machines to make sure they are properly connected to the network. If the machines can't be pinged then there is no way the plug-in will be able to contact them. Possible reasons for this include:
    - Firewall policies limiting the traffic from the plug-in to the app server.
    - The machines are not on the same network.
  – If you are able to ping the machines then the likely cause of the problem is that the port is not active. This could be because the application server or cluster has not been started or the application server has gone down for some reason. You can test this by hand by trying to telnet into the port that the connect is failing on. If you cannot telnet into the port the application server is not up and that problem needs to be resolved before the plug-in will be able to connect successfully.
- Determine whether other activity on the machines where the servers are installed is impairing the server's ability to service a request. Check the processor utilization as measured by the task manager, processor ID, or some other outside tool to see if it:
  – Is not what was expected.
  – Is erratic rather than a constant.
  – Shows that a newly added member of the cluster is not being utilized.
  – Shows that a failing member that has been fixed is not being utilized.
- Check the administrative console to ensure that the application servers are started. View the administrative console for error messages or look in the JVM logs.
- Check the administrative console to ensure that the application servers are started. View the administrative console for error messages.
- In the administrative console, select the problem application server and view its installed applications to verify that they are started.

If none of these steps solves the problem:
- For specific problems that can cause web pages and their contents not to display, seeWeb resource (JSP file, servlet, html file, image, etc) will not display in this InfoCenter.
- Check to see if the problem has been identified and documented using the links in Diagnosing and fixing problems: Resources for learning.
- If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

For current information available from IBM Support on known problems and their resolution, see the following topics on the IBM support page.
- HTTP transport
- HTTP plugin
- HTTP plugin remote install

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

HTTP session manager troubleshooting tips

If you are having problems creating or using HTTP sessions with your Web application hosted by WebSphere Application Server, here are some steps to take:
- View the logs for the application server which hosts the problem application:
  – first, look at messages written while each application is starting. They will be written between the following two messages:

    Starting application: <application>
    ....
    Application started: <application>
  – Within this block, look for any errors or exceptions containing a package name of com.ibm.ws.webcontainer.httpsession. If none are found, this is an indication that the session manager started successfully.

  - Error ″**SRVE0054E: An error occurred while loading session context and Web application**″ indicates that SessionManager didn't start properly for a given application.
  - Look within the logs for any Session Manager related messages. These messages will be in the format SESNxxxxE and SESNxxxxW for errors and warnings, respectively, where xxxx is a number identifying the precise error. Look up the extended error definitions in the Session Manager message table.
- See Best practices for using HTTP Sessions.
- Alternatively, a special servlet can be invoked that displays the current configuration and statistics related to session tracking.
  - Servlet name: **com.ibm.ws.webcontainer.httpsession.IBMTrackerDebug**.
  - It can be invoked from any web module which is enabled to serve by class name. For example, using default_app, **http://localhost:9080/servlet/com.ibm.ws.webcontainer.httpsession.IBMTrackerDebug**.
  - If you are viewing the module via the serve-by-class-name feature, be aware that it may be viewable by anyone who can view the application. You may wish to map a specific, secured URL to the servlet instead and disable the serve-servlets-by-classname feature.
- If you are using **database-based persistent sessions**, look for problems related to the **data source** the Session Manager relies on to keep session state information. For details on diagnosing database related problems see Errors accessing a datasource or connection pool

If none of these steps fixes your problem, check to see if the problem has been identified and documented by looking at the available online support (hints and tips, technotes, and fixes).

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Naming services component troubleshooting tips

″Naming″ is a J2EE service which publishes and provides access to resources such as connection pools, enterprise beans, message listeners, etc, to client processes. If you have problems in accessing a resource which otherwise appears to be healthy, the naming service might be involved. To investigate problems with the WebSphere Application Server Naming service:
- With WebSphere Application Server running, run the **`dumpNameSpace`** command for Windows systems, or the **`dumpNameSpace.sh`** command for UNIX and z/OS systems, and pipe, redirect, or ″more″ the output so that it is easily viewed. This command results in a display of all objects in the WebSphere Application Server namespace, including the directory path and object name.
- If the object a client needs to access does not appear, use the administrative console to verify that:
  - The server hosting the target resource is started.
  - The Web module or EJB container, if applicable, hosting the target resource is running.
  - The JNDI name of the target resource is correct and updated.
  - If the problem resource is remote, that is, not on the same node as the Name Server node, that the JNDI name is fully qualified, including the host name. This is especially applicable to Network Deployment configurations
- If you see an exception that appears to be CORBA related (″CORBA″ appears as part of the exception name) look for a naming-services-specific CORBA minor code, further down in the exception stack, for information on the real cause of the problem. For a list of naming service exceptions and explanations, see the class com.ibm.websphere.naming.WsnCorbaMinorCodes in the Javadoc topic in the InfoCenter.

If none of these steps solve the problem:
- For specific problems that can cause access to named object hosted in WebSphere Application Server to fail, see Cannot look up an object hosted by WebSphere Application Server from a servlet, JSP file, or other client.

- Check to see if the problem has been identified and documented using the links in Diagnosing and fixing problems: Resources for learning.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Messaging (JMS) component troubleshooting tips

If you are having problems deploying or executing applications which use the WebSphere Application Server messaging capabilities, review these articles in the WebSphere Application Server InfoCenter:
- Troubleshooting WebSphere Messaging
- Troubleshooting message-driven beans
- Troubleshooting transactions

If none of these steps solves the problem, check to see if the problem has been identified and documented using the links in Diagnosing and fixing problems: Resources for learning.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Universal Discovery, Description, and Integration, Web Service, and SOAP component troubleshooting tips

If you are having problems deploying or executing applications that use WebSphere Application Server Web Services, Universal Discovery, Description, and Integration (UDDI), or SOAP, try these steps:
- Review the troubleshooting documentation for messaging in this InfoCenter:
  - WSIF troubleshooting tips
- Investigate the following areas for SOAP-related problems:
  - View the error log of the HTTP server to which the SOAP request is sent.
  - Browse the Web site `http://xml.apache.org/soap/` for FAQs and known SOAP issues.

If none of these steps solves the problem, check to see if the problem has been identified and documented using the links in Diagnosing and fixing problems: Resources for learning.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Enterprise bean and EJB container troubleshooting tips

If you are having problems starting an EJB container, or encounter error messages or exceptions that appear to be generated on by an EJB container, follow these steps to resolve the problem:
- Browse the relevant log files for clues:
  - Use the Administrative Console to verify that the application server which hosts the container is running.

– Browse the logs for the application server which hosts the container. Look for the message **`server`** ***`server_name`*** **`open for e-business`** in the server log files. If it does not appear, or if you see the message **`problems occurred during startup`**, browse the server log files for details.

If none of these steps solves the problem, check to see if the problem is identified and documented using the links in Diagnosing and fixing problems: Resources for learning.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Cannot restart the Deployment Manager monitoring policy

The Deployment Manager monitoring policy is permanently set to STOPPED and cannot be changed. Therefore, if the Deployment Manager fails it will never be restarted by WebSphere Application Server monitoring.

To correct the problem, use the z/OS Automatic Restart Facility (ARM) to monitor and restart the Deployment Manager. See z/OS Automatic Restart Facility for information on ARM.

Security components troubleshooting tips

This document explains basic resources and steps for diagnosing security related issues in the WebSphere Application Server, including:
- What log files to look at and what to look for in them.
- A general approach to isolating and resolving security problems.
- When and how to enable security-related trace.
- An overview and table of security-related CORBA minor codes.

The following security-related problems are addressed elsewhere in this InfoCenter:
- Errors and access problems after enabling security
- Errors after enabling SSL, or SSL-related error messages
- Errors trying to configure and enable security

If none of these steps solves the problem, check to see if the problem has been identified and documented using the links in Diagnosing and fixing problems: Resources for learning. If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

Note: for an overview of WebSphere Application Server security components such as SASzSAS and how they work, see Getting started with security.

Log filesSDSF output logs

When troubleshooting the security component, browse the JVM logs SDSF logs for the server that hosts the resource you are trying to access. The following is a sample of messages you would expect to see from a server in which the security service has started successfully:

```
SASRas          A  JSAS0001I:  Security  configuration  initialized.
SASRas          A  JSAS0002I:  Authentication  protocol:  CSIV2/IBM
SASRas          A  JSAS0003I:  Authentication  mechanism:  SWAM
SASRas          A  JSAS0004I:  Principal  name:  MYHOSTNAME/aServerID
SASRas          A  JSAS0005I:  SecurityCurrent  registered.
SASRas          A  JSAS0006I:  Security  connection  interceptor  initialized.
SASRas          A JSAS0007I: Client request interceptor registered.
SASRas          A JSAS0008I: Server request interceptor registered.
SASRas          A JSAS0009I: IOR interceptor registered.
NameServerImp I NMSV0720I: Do Security service listener registration.
SecurityCompo A SECJ0242A: Security service is starting
UserRegistryI A SECJ0136I: Custom Registry:com.ibm.ws.security.registry.nt.
NTLocalDomainRegistryImpl has been initialized
SecurityCompo A SECJ0202A: Admin application initialized successfully
SecurityCompo A SECJ0203A: Naming application initialized successfully
SecurityCompo A SECJ0204A: Rolebased authorizer initialized successfully
SecurityCompo A SECJ0205A: Security Admin mBean registered successfully
SecurityCompo A SECJ0243A: Security service started successfully
SecurityCompo A SECJ0210A: Security enabled true

Messages begining with BBOM0001I are related to zOS specific implementations
of zSAS and CSIv2. They appear in both the controller and servant
but are only applicable in the controller.

 BBOM0001I com_ibm_Server_Security_Enabled: 1.
 BBOM0001I com_ibm_CSI_claimTLClientAuthenticationSupported: 1.
 BBOM0001I com_ibm_CSI_claimTLClientAuthenticationRequired: 0.
 BBOM0001I com_ibm_CSI_claimTransportAssocSSLTLSSupported: 1.
 BBOM0001I com_ibm_CSI_claimTransportAssocSSLTLSRequired: 0.
 BBOM0001I com_ibm_CSI_claimMessageConfidentialityRequired: 0.
 BBOM0001I com_ibm_CSI_claimClientAuthenticationSupported: 1.
 BBOM0001I com_ibm_CSI_claimClientAuthenticationRequired: 0.
 BBOM0001I com_ibm_CSI_claimClientAuthenticationtype:
 SAFUSERIDPASSWORD.
 BBOM0001I com_ibm_CSI_claimIdentityAssertionTypeSAF: 0.
 BBOM0001I com_ibm_CSI_claimIdentityAssertionTypeDN: 0.
 BBOM0001I com_ibm_CSI_claimIdentityAssertionTypeCert: 0.
 BBOM0001I com_ibm_CSI_claimMessageIntegritySupported: NOT SET,DEFAULT=1.
 BBOM0001I com_ibm_CSI_claimMessageIntegrityRequired: NOT SET,DEFAULT=1.
 BBOM0001I com_ibm_CSI_claimStateful: 1.
 BBOM0001I com_ibm_CSI_claimSecurityLevel: HIGH.
 BBOM0001I com_ibm_CSI_claimSecurityCipherSuiteList: NOT SET.
 BBOM0001I com_ibm_CSI_claimKeyringName: WASKeyring.
 BBOM0001I com_ibm_CSI_claim_ssl_sys_v2_timeout: NOT SET, DEFAULT=100.
 BBOM0001I com_ibm_CSI_claim_ssl_sys_v3_timeout: 600.
 BBOM0001I com_ibm_CSI_performTransportAssocSSLTLSSupported: 1.
 BBOM0001I security_sslClientCerts_allowed: 0.
 BBOM0001I security_kerberos_allowed: 0.
 BBOM0001I security_userid_password_allowed: 0.
 BBOM0001I security_userid_passticket_allowed: 1.
 BBOM0001I security_assertedID_IBM_accepted: 0.
 BBOM0001I security_assertedID_IBM_sent: 0.
 BBOM0001I nonauthenticated_clients_allowed: 1.
 BBOM0001I security_remote_identity: WSGUEST.
 BBOM0001I security_local_identity: WSGUEST.
 BBOM0001I security_EnableRunAsIdentity: 0.

 Messages beginning with BBOO0222I are common to java WebSphere security.
They appear in both the controller and servant but are applicable to the servant.

 BBOO0222I SECJ0240I: Security service initialization completed successfully
 BBOO0222I SECJ0215I: Successfully set JAAS login provider
 configuration class to com.ibm.ws.security.auth.login.Configuration.
 BBOO0222I SECJ0136I: Custom
 Registry:com.ibm.ws.security.registry.zOS.SAFRegistryImpl has been   initialized
 BBOO0222I SECJ0157I: Loaded Vendor AuthorizationTable:
 com.ibm.ws.security.core.SAFAuthorizationTableImpl
 BBOO0222I SECJ0243I: Security service started successfully
 BBOO0222I SECJ0210I: Security enabled true
```

The following is an example of messages from a server which cannot start the security service, in this case because the administrative user ID and password given to communicate with the user registry is wrong, or the user registry itself is down or misconfigured:

```
SASRas         A JSAS0001I: Security configuration initialized.
SASRas         A JSAS0002I: Authentication protocol: CSIV2/IBM
SASRas         A JSAS0003I: Authentication mechanism: SWAM
SASRas         A JSAS0004I: Principal name: MYHOSTNAME/aServerID
SASRas         A JSAS0005I: SecurityCurrent registered.
SASRas         A JSAS0006I: Security connection interceptor initialized.
SASRas         A JSAS0007I: Client request interceptor registered.
SASRas         A JSAS0008I: Server request interceptor registered.
SASRas         A JSAS0009I: IOR interceptor registered.
NameServerImp I NMSV0720I: Do Security service listener registration.

SecurityCompo A SECJ0242A: Security service is starting
UserRegistryI A SECJ0136I: Custom Registry:com.ibm.ws.security.
registry.nt.NTLocalDomainRegistryImpl has been initialized
Authenticatio E SECJ4001E: Login failed for badID/<null>
javax.security.auth.login.LoginException: authentication failed: bad user/password
```

The following is an example of messages from a server for which LDAP has been specified as the security mechanism, but the LDAP keys have not been properly configured:

```
SASRas         A JSAS0001I: Security configuration initialized.
SASRas         A JSAS0002I: Authentication protocol: CSIV2/IBM
SASRas         A JSAS0003I: Authentication mechanism: LTPA
SASRas         A JSAS0004I: Principal name: MYHOSTNAME/anID
SASRas         A JSAS0005I: SecurityCurrent registered.
SASRas         A JSAS0006I: Security connection interceptor initialized.
SASRas         A JSAS0007I: Client request interceptor registered.
SASRas         A JSAS0008I: Server request interceptor registered.
SASRas         A JSAS0009I: IOR interceptor registered.
NameServerImp I NMSV0720I: Do Security service listener registration.
SecurityCompo A SECJ0242A: Security service is starting
UserRegistryI A SECJ0136I: Custom Registry:com.ibm.ws.security.registry.nt.
NTLocalDomainRegistryImpl has been initialized
SecurityServe E SECJ0237E: One or more vital LTPAServerObject configuration
attributes are null or not available. The attributes and values are password :
LTPA password does exist, expiration time 30, private key <null>, public key <null>,
and shared key <null>.
```

A problem with the SSL configuration might lead to the following message. You should ensure that the keystore location and keystore passwords are valid. Also, ensure the keystore has a valid personal certificate and that the personal certificate public key or CA root has been extracted on put into the truststore.

```
SASRas          A  JSAS0001I:  Security  configuration  initialized.
SASRas          A  JSAS0002I:  Authentication  protocol:  CSIV2/IBM
SASRas          A  JSAS0003I:  Authentication  mechanism:  SWAM
SASRas          A  JSAS0004I:  Principal  name:  MYHOSTNAME/aServerId
SASRas          A  JSAS0005I:  SecurityCurrent  registered.
SASRas          A  JSAS0006I:  Security  connection  interceptor  initialized.
SASRas          A  JSAS0007I:  Client  request  interceptor  registered.
SASRas          A  JSAS0008I:  Server  request  interceptor  registered.
SASRas          A  JSAS0009I:  IOR  interceptor  registered.
SASRas          E  JSAS0026E:  [SecurityTaggedComponentAssistorImpl.register]
Exception  connecting  object  to  the  ORB.   Check  the  SSL  configuration  to  ensure
 that  the  SSL  keyStore  and  trustStore  properties  are  set  properly.   If  the  problem
persists,  contact  support  for  assistance.  org.omg.CORBA.OBJ_ADAPTER:
ORB_CONNECT_ERROR  (5)  -  couldn't  get  Server  Subcontract   minor  code:
4942FB8F   completed:  No
```

General approach for troubleshooting security-related issues

When troubleshooting security-related problems, the following questions are very helpful and should be considered:

Does the problem occur when security is disabled?
: This is a good litmus test to determine that a problem is security related. However, just because a problem only occurs when security is enabled does not always make it a security problem. More troubleshooting is necessary to ensure the problem is really security-related.

Did security appear to initialize properly?
: A lot of security code is visited during initialization. So you will likely see problems there first if the problem is configuration related. The following sequence of messages generated in the SystemOut.logSDSF active log indicate normal code initialization of an application server. Non-security messages have been removed from the sequence that follows. This sequence will vary based on the configuration, but the messages are similar:

```
SASRas          A  JSAS0001I:  Security  configuration  initialized.
SASRas          A  JSAS0002I:  Authentication  protocol:  CSIV2/IBM
SASRas          A  JSAS0003I:  Authentication  mechanism:  SWAM
SASRas          A  JSAS0004I:  Principal  name:  BIRKT20/pbirk
SASRas          A  JSAS0005I:  SecurityCurrent  registered.
SASRas          A  JSAS0006I:  Security  connection  interceptor  initialized.
SASRas          A  JSAS0007I:  Client  request  interceptor  registered.
SASRas          A  JSAS0008I:  Server  request  interceptor  registered.
SASRas          A  JSAS0009I:  IOR  interceptor  registered.
NameServerImp  I  NMSV0720I:  Do  Security  service  listener  registration.
SecurityCompo  A  SECJ0242A:  Security  service  is  starting
UserRegistryI  A  SECJ0136I:  Custom  Registry:com.ibm.ws.security.registry.nt.
NTLocalDomainRegistryImpl  has  been  initialized
SecurityCompo  A  SECJ0202A:  Admin  application  initialized  successfully
SecurityCompo  A  SECJ0203A:  Naming  application  initialized  successfully
SecurityCompo  A  SECJ0204A:  Rolebased  authorizer  initialized  successfully
SecurityCompo  A  SECJ0205A:  Security  Admin  mBean  registered  successfully
SecurityCompo  A  SECJ0243A:  Security  service  started  successfully

SecurityCompo  A  SECJ0210A:  Security  enabled  true

 Trace:  2003/08/25  13:06:31.034  01  t=9EA930  c=UNK  key=P8  (13007002)
   FunctionName:  com.ibm.ws.security.auth.login.Configuration
   SourceId:  com.ibm.ws.security.auth.login.Configuration
   Category:  AUDIT
```

```
    ExtendedMessage: SECJ0215I: Successfully set JAAS login provider
    configuration class to com.ibm.ws.security.auth.login.Configuration.
  Trace: 2003/08/25 13:06:31.085 01 t=9EA930 c=UNK key=P8 (13007002)
    FunctionName: com.ibm.ws.security.core.SecurityDM
    SourceId: com.ibm.ws.security.core.SecurityDM
    Category: INFO
    ExtendedMessage: SECJ0231I: The Security component's
    FFDC Diagnostic Module com.ibm.ws.security.core.SecurityDM
    registered success
  fully: true.
  Trace: 2003/08/25 13:06:31.086 01 t=9EA930 c=UNK key=P8 (0000000A)
    Description: Log Boss/390 Error
    from filename: ./bborjtr.cpp
    at line: 812
    error message: BBOO0222I SECJ0231I: The Security component's
FFDC Diagnostic Module com.ibm.ws.security.core.SecurityDM registered
 successfully: true.
  Trace: 2003/08/25 13:06:32.426 01 t=9EA930 c=UNK key=P8 (13007002)
    FunctionName: com.ibm.ws.security.core.SecurityComponentImpl
    SourceId: com.ibm.ws.security.core.SecurityComponentImpl
    Category: INFO
    ExtendedMessage: SECJ0309I: Java 2 Security is disabled.
  Trace: 2003/08/25 13:06:32.427 01 t=9EA930 c=UNK key=P8 (0000000A)
    Description: Log Boss/390 Error
    from filename: ./bborjtr.cpp
    at line: 812
    error message: BBOO0222I SECJ0309I: Java 2 Security is disabled.
  Trace: 2003/08/25 13:06:32.445 01 t=9EA930 c=UNK key=P8 (13007002)
    FunctionName: com.ibm.ws.security.core.SecurityComponentImpl
    SourceId: com.ibm.ws.security.core.SecurityComponentImpl
    Category: INFO
    ExtendedMessage: SECJ0212I: WCCM JAAS configuration information
successfully pushed to login provider class.
  Trace: 2003/08/25 13:06:32.445 01 t=9EA930 c=UNK key=P8 (0000000A)
    Description: Log Boss/390 Error
    from filename: ./bborjtr.cpp
    at line: 812
    error message: BBOO0222I SECJ0212I: WCCM JAAS configuration
  information successfully pushed to login provider class.
  Trace: 2003/08/25 13:06:32.459 01 t=9EA930 c=UNK key=P8 (13007002)
    FunctionName: SecurityComponentImpl
    SourceId: SecurityComponentImpl
    Category: WARNING
    ExtendedMessage: BBOS1000W  LTPA or ISCF are configured as the
authentication mechanism but SSO is disabled.
  Trace: 2003/08/25 13:06:32.459 01 t=9EA930 c=UNK key=P8 (0000000A)
    Description: Log Boss/390 Error
    from filename: ./bborjtr.cpp
    at line: 824
    error message: BBOS1000W  LTPA or ISCF are configured as the
authentication mechanism but SSO is disabled.
  Trace: 2003/08/25 13:06:32.463 01 t=9EA930 c=UNK key=P8 (13007002)
    FunctionName: com.ibm.ws.security.core.SecurityComponentImpl
    SourceId: com.ibm.ws.security.core.SecurityComponentImpl
    Category: INFO
    ExtendedMessage: SECJ0240I: Security service initialization completed
```

```
successfully
 Trace: 2003/08/25 13:06:32.463 01 t=9EA930 c=UNK key=P8 (0000000A)
   Description: Log Boss/390 Error
   from filename: ./bborjtr.cpp
   at line: 812
   error message: BBOO0222I SECJ0240I: Security service initialization
completed successfully
 Trace: 2003/08/25 13:06:39.718 01 t=9EA930 c=UNK key=P8 (13007002)
   FunctionName: com.ibm.ws.security.registry.UserRegistryImpl
   SourceId: com.ibm.ws.security.registry.UserRegistryImpl
   Category: AUDIT
   ExtendedMessage: SECJ0136I: Custom Registry:
com.ibm.ws.security.registry.zOS.SAFRegistryImpl has been initialized
 Trace: 2003/08/25 13:06:41.967 01 t=9EA930 c=UNK key=P8 (13007002)
   FunctionName: com.ibm.ws.security.core.WSAccessManager
   SourceId: com.ibm.ws.security.core.WSAccessManager
   Category: AUDIT
   ExtendedMessage: SECJ0157I: Loaded Vendor AuthorizationTable:
com.ibm.ws.security.core.SAFAuthorizationTableImpl
 Trace: 2003/08/25 13:06:43.136 01 t=9EA930 c=UNK key=P8 (13007002)
   FunctionName: com.ibm.ws.security.role.RoleBasedAuthorizerImpl
   SourceId: com.ibm.ws.security.role.RoleBasedAuthorizerImpl
   Category: AUDIT
   ExtendedMessage: SECJ0157I: Loaded Vendor AuthorizationTable:
com.ibm.ws.security.core.SAFAuthorizationTableImpl
 Trace: 2003/08/25 13:06:43.789 01 t=9EA930 c=UNK key=P8 (13007002)
   FunctionName: com.ibm.ws.security.core.SecurityComponentImpl
   SourceId: com.ibm.ws.security.core.SecurityComponentImpl
   Category: INFO
   ExtendedMessage: SECJ0243I: Security service started successfully
 Trace: 2003/08/25 13:06:43.789 01 t=9EA930 c=UNK key=P8 (0000000A)
   Description: Log Boss/390 Error
   from filename: ./bborjtr.cpp
   at line: 812
   error message: BBOO0222I SECJ0243I: Security service started successfully
 Trace: 2003/08/25 13:06:43.794 01 t=9EA930 c=UNK key=P8 (13007002)
   FunctionName: com.ibm.ws.security.core.SecurityComponentImpl
   SourceId: com.ibm.ws.security.core.SecurityComponentImpl
   Category: INFO
   ExtendedMessage: SECJ0210I: Security enabled true
 Trace: 2003/08/25 13:06:43.794 01 t=9EA930 c=UNK key=P8 (0000000A)
   Description: Log Boss/390 Error
   from filename: ./bborjtr.cpp
   at line: 812
   error message: BBOO0222I SECJ0210I: Security enabled true
 Trace: 2003/08/25 13:07:06.474 01 t=9EA930 c=UNK key=P8 (13007002)
   FunctionName: com.ibm.ws.security.core.WSAccessManager
   SourceId: com.ibm.ws.security.core.WSAccessManager
   Category: AUDIT
   ExtendedMessage: SECJ0157I: Loaded Vendor AuthorizationTable:
com.ibm.ws.security.core.SAFAuthorizationTableImpl
 Trace: 2003/08/25 13:07:09.315 01 t=9EA930 c=UNK key=P8 (13007002)
   FunctionName: com.ibm.ws.security.core.WSAccessManager
   SourceId: com.ibm.ws.security.core.WSAccessManager
   Category: AUDIT
```

ExtendedMessage: SECJ0157I: Loaded Vendor AuthorizationTable: com.ibm.ws.security.core.SAFAuthorizationTableImpl

A single stack trace tells a lot about the problem. What code initiated the code that failed? What is the failing component? Which class did the failure actually come from? Sometimes the stack trace is all that is needed to solve the problem and it can pinpoint the root cause. Other times, it can only give us a clue, and could actually be misleading. When support analyzes a stack trace, they may request additional trace if it is not clear what the problem is. If it appears to be security related and the solution cannot be determined from the stack trace or problem description, you will be asked to gather the following trace specification:
`SASRas=all=enabled:com.ibm.ws.security.*=all=enabled` from all processes involved.

Is this a distributed security problem or a local security problem?

- If the problem is local, that is the code involved does not make a remote method invocation, then troubleshooting is isolated to a single process. It is important to know when a problem is local versus distributed since the behavior of the ORB, among other components, is different between the two. Once a remote method invocation takes place, an entirely different security code path is entered.
- When you know that the problem involves two or more servers, the techniques of troubleshooting change. You will need to trace all servers involved simultaneously so that the trace shows the client and server sides of the problem. Try to make sure the timestamps on all machines match as closely as possible so that you can find the request and reply pair from two different processes. Enable both SASzSAS and Security trace using the trace specification: `SASRas=all=enabled:com.ibm.ws.security.*=all=enabled`.

Is the problem related to authentication or authorization?

Most security problems fall under one of these two categories. Authentication is the process of determing who the caller is. Authorization is the process of validating that the caller has the proper authority to invoke the requested method. When authentication fails, typically this is related to either the authentication protocol, authentication mechanism or user registry. When authorization fails, this is usually related to the application bindings from assembly and/or deployment and to the caller's identity who is accessing the method and the roles required by the method.

Is this a Web or EJB request?

Web requests have a completely different code path than EJB requests. Also, there are different security features for Web requests than for EJB requests, requiring a completely different body of knowledge to resolve. For example, when using the LTPA authentication mechanism, the Single SignOn feature is available for Web requests but not for EJB requests. Web requests involve HTTP header information not required by EJB requests due to the protocol differences. Also, the Web container (or servlet engine) is involved in the entire process. Any of these components could be involved in the problem and all should be considered during troubleshooting, based on the type of request and where the failure occurs.

Secure EJB requests heavily involve the ORB and Naming components since they flow over the RMI/IIOP protocol. In addition, when work flow management (WLM) is enabled, other behavior changes in the code can be observed. All of these components interact closely for security to work properly in this environment. At times, trace in any or all of these components might be necessary to troubleshoot problems in this area. The trace specification to begin with is `SASRas=all=enabled:com.ibm.ws.security.*=all=enabled`. ORB trace is also very beneficial when the SASzSAS/Security trace does not seem to pinpoint the problem.

Secure EJB requests are passed from the controller to the servant. Web requests are mostly ignored by the controller. As a result, EJB requests are first processed and authenticated by the zSAS or CSIv2 layers of security. Authorization is done by the servant. If an authentication failure occurs, the zSAS type level of tracing must be turned on to diagnose the problem. Other problems can be diagnosed using the WebSphere Application Server component tracing (CTRACE) facility.

Does the problem seem to be related to the Secure Sockets Layer (SSL)?

The Secure Socket Layer (SSL) is a totally distinct separate layer of security. Troubleshooting SSL problems are usually separate from troubleshooting authentication and/or authorization problems. There are many things to consider. Usually, SSL problems are first time setup problems because the configuration can be difficult. Each client must contain the server's signer certificate. During mutual authentication, each server must contain the client's signer certificate. Also, there can be protocol differences (SSLv3 vs. TLS), and listener port problems related to stale IORs (i.e., IORs from a server reflecting the port prior to the server restarting).

In z/OS, two variations of SSL are used. To determine the cause of an SSL problem on z/OS, you will have to be aware of what protocol is being used. System SSL is used by the IIOP and HTTPS protocols. Java Secure Socket Extension (JSSE) is used by all other protocols, for example, Simple Object Access Protocol (SOAP). System SSL requests are handled in the controller and are used by zSAS and CSIv1 security. JSSE is predominately used by the servant, but there are cases where it is used in the controller as well.

For SSL problems, we sometimes request an SSL trace to determine what is happening with the SSL handshake. The SSL handshake is the process which occurs when a client opens a socket to a server. If anything goes wrong with the key exchange, cipher exchange, etc. the handshake will fail and thus the socket is invalid. Tracing JSSE (the SSL implementation used in WebSphere Application Server) involves the following steps:

- Ensure that the client and server processes contain an `ibmjsse-debug.jar` file in the`java/jre/lib/ext` directory. The `ibmjsse-debug.jar` is shipped with the product. You can locate the file under *`installation_directory`*`\web\docs\jsse`. Make sure you remove the existing `ibmjsse.jar` file from this directory after putting in the `ibmjsse-debug.jar`. If both exist in the `/ext` directory, the JSSE trace will not be complete.
- Set the following system property on the client and server processes: `-Djavax.net.debug=true`. For the server, add this to the Generic JVM Arguments property of the Java virtual machine settings page.
- Turn on ORB trace as well.
- Recreate the problem. The `SystemOut.log`SDSF active log of both processes should contain the JSSE trace. You will find trace similar to the following:

```
SSLConnection: install <com.ibm.sslite.e@3ae78375>
>> handleHandshakeV2 <com.ibm.sslite.e@3ae78375>
>> handshakeV2 type = 1
>> clientHello: SSLv2.
SSL client version: 3.0
...
...
...
JSSEContext: handleSession[Socket[addr=null,port=0,localport=0]]

<< sendServerHello.
SSL version: 3.0
SSL_RSA_WITH_RC4_128_MD5
HelloRandom
...
...
...
<< sendCertificate.
<< sendServerHelloDone.
>> handleData <com.ibm.sslite.e@3ae78375>
>> handleHandshake <com.ibm.sslite.e@3ae78375>
>> handshakeV3 type = 16

>> clientKeyExchange.
>> handleData <com.ibm.sslite.e@3ae78375>
>> handleChangeCipherSpec <com.ibm.sslite.e@3ae78375>
>> handleData <com.ibm.sslite.e@3ae78375>
>> handleHandshake <com.ibm.sslite.e@3ae78375>
>> handshakeV3 type = 20
>> finished.
<< sendChangeCipherSpec.
<< sendFinished.
```

```
 JSSEContext: handleConnection[Socket
[addr=boss0106.plex1.l2.ibm.com/9.38.48.108,port=2139,localport=8878]]
 JSSEContext: handleConnection[Socket
[addr=boss0106.plex1.l2.ibm.com/9.38.48.108,port=2140,localport=8878]]
 TrustManagerFactoryImpl: trustStore is :
 /WebSphere/V5R0M0/AppServer/etc/DummyServerTrustFile.jks
 TrustManagerFactoryImpl: trustStore type is : JKS
 TrustManagerFactoryImpl: init truststore
 JSSEContext: handleConnection[Socket
[addr=boss0106.plex1.l2.ibm.com/9.38.48.108,port=2142,localport=8878]]
 KeyManagerFactoryImpl: keyStore is :
/WebSphere/V5R0M0/AppServer/etc/DummyServerKeyFile.jks
 KeyManagerFactoryImpl: keyStore type is : JKS
 KeyManagerFactoryImpl: init keystore
 KeyManagerFactoryImpl: init keystore
 JSSEContext: handleConnection[Socket
[addr=boss0106.plex1.l2.ibm.com/9.38.48.108,port=2143,localport=8878]]
 JSSEContext: handleSession[Socket
[addr=BOSSXXXX.PLEX1.L2.IBM.COM/9.38.48.108,port=8879,localport=2145]]
 JSSEContext:  confirmPeerCertificate
[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM/9.38.48.108,port=8879,
  localport=2145]]
 X509TrustManagerImpl: checkServerTrusted
 X509TrustManagerImpl: Certificate [
 [
   Version: V3
   Subject: CN=jserver, OU=SWG, O=IBM, C=US
   Signature Algorithm: MD5withRSA, OID = 1.2.840.113549.1.1.4
0  Key:  IBMJCE RSA Public Key:
 modulus:
 100949966922395090747968287561185391075683695663138899555538950668
6622953008589748001058216362638201577071902071311277365773252660799
 128781182947273802312699983556527878615792292244995317112436562491
4899043818842651193550377312654086540073888633031017463144383376 01
 26454073567994420539169324292133155134224789 1
 public exponent:
 65537
0  Validity: [From: Fri Jun 21 20:08:18 GMT 2002,
               To: Thu Mar 17 20:08:18 GMT 2005]
   Issuer: CN=jserver, OU=SWG, O=IBM, C=US
   SerialNumber: [    3d1387b2 ]
0]
   Algorithm: [MD5withRSA]
   Signature:
 0000: 54 DC B5 FA 64 C9 CD FE   B3 EF 15 22 3D D0 20 31  T...d......"=. 1
 0010: 99 F7 A7 86 F9 4C 82 9F   6E 4B 7B 47 18 2E C6 25  .....L..nK.G...%
 0020: 5B B2 9B 78 D8 76 5C 82   07 95 DD B8 44 62 02 62  [..x.v\.....Db.b
 0030: 60 2A 0A 6D 4F B9 0A 98   14 27 E9 BB 1A 84 8A D1  `*.mO....'......
 0040: C2 22 AF 70 9E A5 DF A2   FD 57 37 CE 3A 63 1B EB  .".p.....W7.:c..
 0050: E8 91 98 9D 7B 21 4A B5   2C 94 FC A9 30 C2 74 72  .....!J.,...0.tr
 0060: 95 01 54 B1 29 E7 F8 9E   6D F3 B5 D7 B7 D2 9E 9B  ..T.)...m.......
 0070: 85 D8 E4 CF C2 D5 3B 64   F0 07 17 9E 1E B9 2F 79  ......;d....../y
0]
 X509TrustManagerImpl: Certificate [
 [
   Version: V3
   Subject: CN=jserver, OU=SWG, O=IBM, C=US
   Signature Algorithm: MD5withRSA, OID = 1.2.840.113549.1.1.4
0  Key:  IBMJCE RSA Public Key:
 modulus:
 1009499669223950907479682875611853910756836956631388995553895066866
22953008589748001058216362638201577071902071311277365773252660799
 1287811829472738023126999835565278786157922922449953171124365624914
8990438188426511935503773126540865400738886330310174631443833760 1
 26454073567994420539169324292133155134224789 1
 public exponent:
 65537
0  Validity: [From: Fri Jun 21 20:08:18 GMT 2002,
               To: Thu Mar 17 20:08:18 GMT 2005]
   Issuer: CN=jserver, OU=SWG, O=IBM, C=US
   SerialNumber: [    3d1387b2 ]
0]
   Algorithm: [MD5withRSA]
   Signature:
 0000: 54 DC B5 FA 64 C9 CD FE   B3 EF 15 22 3D D0 20 31  T...d......"=. 1
 0010: 99 F7 A7 86 F9 4C 82 9F   6E 4B 7B 47 18 2E C6 25  .....L..nK.G...%
 0020: 5B B2 9B 78 D8 76 5C 82   07 95 DD B8 44 62 02 62  [..x.v\.....Db.b
 0030: 60 2A 0A 6D 4F B9 0A 98   14 27 E9 BB 1A 84 8A D1  `*.mO....'......
 0040: C2 22 AF 70 9E A5 DF A2   FD 57 37 CE 3A 63 1B EB  .".p.....W7.:c..
 0050: E8 91 98 9D 7B 21 4A B5   2C 94 FC A9 30 C2 74 72  .....!J.,...0.tr
 0060: 95 01 54 B1 29 E7 F8 9E   6D F3 B5 D7 B7 D2 9E 9B  ..T.)...m.......
 0070: 85 D8 E4 CF C2 D5 3B 64   F0 07 17 9E 1E B9 2F 79  ......;d....../y
0]
 JSSEContext: handleConnection[Socket[addr=boss0106.plex1.l2.ibm.com
/9.38.48.108,port=2144,localport=8878]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2145]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2146]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2147]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2148]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2149]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2150]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2151]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2152]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2153]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2154]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2155]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2156]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2157]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2158]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2159]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2160]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2161]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2162]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2163]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2164]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2165]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2166]]

 JSSEContext: handleSession[Socket[addr=boss0106.plex1.l2.ibm.com
/9.38.48.108,port=9443,localport=2167]]
 JSSEContext:  confirmPeerCertificate[Socket[addr=boss0106.plex1.l2.ibm.com
/9.38.48.108,port=9443,localport=2167]]
```

```
 X509TrustManagerImpl: checkServerTrusted
 X509TrustManagerImpl: Certificate [
 [
   Version: V3
   Subject: CN=WAS z/OS Deployment Manager, O=IBM
   Signature Algorithm: SHA1withRSA, OID = 1.2.840.113549.1.1.5
0  Key:  IBMJCE RSA Public Key:
 modulus:
 128409482671196514693124865480209574419464134944983704395586039015825 89
8755033448419534105183133064366466828741516428176579440511007
 6258795528749232737808897160958348495006972731464152299032614592135114
1936153996255599713608514059109825934562585361738939634066476 6
 6499577495278411071215903524293486342870315 01
 public exponent:
 65537
0  Validity: [From: Fri Jul 25 05:00:00 GMT 2003,
                To: Mon Jul 26 04:59:59 GMT 2004]
   Issuer: CN=WAS CertAuth, C=US
   SerialNumber: [    02]
0Certificate Extensions: 3
 [1]: ObjectId: 2.16.840.1.113730.1.13 Criticality=false
 Extension unknown: DER encoded OCTET string =
 0000: 04 3C 13 3A 47 65 6E 65   72 61 74 65 64 20 62 79  .<.:Generated by
 0010: 20 74 68 65 20 53 65 63   75 72 65 57 61 79 20 53   the SecureWay S
 0020: 65 63 75 72 69 74 79 20   53 65 72 76 65 72 20 66  ecurity Server f
 0030: 6F 72 20 7A 2F 4F 53 20   28 52 41 43 46 29        or z/OS (RACF)
-[2]: ObjectId: 2.5.29.14 Criticality=false
 SubjectKeyIdentifier [
 KeyIdentifier [
 0000: 05 6A CD 7F AE AF 89 78   99 A8 F1 5B 64 8B 9F AF  .j.....x...[d...
 0010: 73 1B 58 65                                        s.Xe
 ]
 ]
0[3]: ObjectId: 2.5.29.35 Criticality=false
 AuthorityKeyIdentifier [
 KeyIdentifier [
 0000: 7E D1 7B 17 74 D3 AD D1   7D D8 F8 33 85 19 04 F8  ....t......3....
 0010: 36 51 57 16                                        6QW.
 ]
0]
0]
   Algorithm: [SHA1withRSA]
   Signature:
 0000: 73 0D FC E1 8A B3 42 E1   04 73 72 B1 C6 C9 87 54  s.....B..sr....T
 0010: 87 57 02 FA 41 32 D8 B0   39 09 86 CB 6B 03 B6 F9  .W..A2..9...k...
 0020: 62 8D 95 36 56 0E D4 D2   F7 7A 8D 4B FB 0B FD 91  b..6V....z.K....
 0030: 89 A8 08 41 30 E2 27 DC   15 5F 2C F4 CD 2F 6B 8E  ...A0.'.._,../k.
 0040: 21 2A 88 53 46 27 68 9B   55 14 38 8E 1F 50 95 BC  !*.SF'h.U.8..P..
 0050: A8 46 F6 68 97 9E 7B 65   9E E8 A7 34 B2 C8 63 CF  .F.h...e...4..c.
 0060: 73 C8 4E 25 0A EF C5 8F   04 A4 EB 8C CC 33 84 26  s.N%.........3.&
 0070: 5D FD 7C AD 7B 02 13 5A   86 A1 89 93 1E A4 93 63  ]......Z.......c
0]
 X509TrustManagerImpl: Certificate [
 [
   Version: V3
   Subject: CN=WAS CertAuth, C=US
   Signature Algorithm: SHA1withRSA, OID = 1.2.840.113549.1.1.5
0  Key:  IBMJCE RSA Public Key:
 modulus:
 1167408593733331602218385578183389496484587418638676352829560040529918
4055868120819997783340160989574822236906623032978514888325114 4
 2382911186804921983976695395381692334250582278359056431484427844566504
4149179995259286489524298703792940845345562755277231738207701 5
 8287135852202125028395464960718394963084303 93
 public exponent:
 65537
0  Validity: [From: Fri Jul 25 05:00:00 GMT 2003,
               To: Sat Jul 24 04:59:59 GMT 2010]
   Issuer: CN=WAS CertAuth, C=US
   SerialNumber: [  0  ]
0Certificate Extensions: 4
 [1]: ObjectId: 2.16.840.1.113730.1.13 Criticality=false
 Extension unknown: DER encoded OCTET string =
 0000: 04 3C 13 3A 47 65 6E 65   72 61 74 65 64 20 62 79  .<.:Generated by
 0010: 20 74 68 65 20 53 65 63   75 72 65 57 61 79 20 53   the SecureWay S
 0020: 65 63 75 72 69 74 79 20   53 65 72 76 65 72 20 66  ecurity Server f
 0030: 6F 72 20 7A 2F 4F 53 20   28 52 41 43 46 29        or z/OS (RACF)
-[2]: ObjectId: 2.5.29.14 Criticality=false
 SubjectKeyIdentifier [
 KeyIdentifier [
 0000: 7E D1 7B 17 74 D3 AD D1   7D D8 F8 33 85 19 04 F8  ....t......3....
 0010: 36 51 57 16                                        6QW.
 ]
 ]
0[3]: ObjectId: 2.5.29.15 Criticality=true
 KeyUsage [
   Key_CertSign
   Crl_Sign
 ]
0[4]: ObjectId: 2.5.29.19 Criticality=true
 BasicConstraints:[
 CA:true
 PathLen:2147483647
 ]
0]
   Algorithm: [SHA1withRSA]
   Signature:
 0000: 43 88 AB 19 5D 00 54 57   5E 96 FA 85 CE 88 4A BF  C...].TW^.....J.
 0010: 6E CB 89 4C 56 BE EF E6   8D 2D 74 B5 83 1A EF 9C  n..LV....-t.....
 0020: B3 82 F2 16 84 FA 5C 50   53 2A B4 FD EB 27 98 5D  ......\PS*...'.]
 0030: 43 48 D3 74 85 21 D1 E1   F2 63 9E FB 58 2A F3 6A  CH.t.!...c..X*.j
 0040: 44 D2 F5 7D B2 55 B9 5E   32 11 78 B6 34 8E 4B 1D  D....U.^2.x.4.K.
 0050: F3 82 1D C1 5F 7B 3F AD   C9 29 FA FF D1 D1 13 2C  ...._.?..).....,
 0060: 57 F7 7B 51 02 99 6F ED   54 E1 51 34 B8 51 BE 97  W..Q..o.T.Q4.Q..
 0070: 30 AC 4F 89 AB AA 8A B2   E1 40 89 2E 18 C7 0E 15  0.O......@......
0]
 JSSEContext: handleConnection[Socket[addr=boss0106.plex1.l2.ibm.com
/9.38.48.108,port=9443,localport=2167]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2168]]

 JSSEContext: handleConnection[Socket[addr=boss0106.plex1.l2.ibm.com
/9.38.48.108,port=2235,localport=8878]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2236]]
 JSSEContext: handleSession[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8880,localport=2238]]
 JSSEContext:  confirmPeerCertificate[Socket
[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8880,localport=2238]]
 X509TrustManagerImpl: checkServerTrusted
 X509TrustManagerImpl: Certificate [

 [
   Version: V3
   Subject: CN=jserver, OU=SWG, O=IBM, C=US
   Signature Algorithm: MD5withRSA, OID = 1.2.840.113549.1.1.4
0  Key:  IBMJCE RSA Public Key:
 modulus:
 100949966922395090747968287561185391075683695663138899555389506686622953
0085897480010582163626382015770719020713112773657732526607 99
 128781182947273802312699983556527878615792292244995317112436562491 4
899043818842651193550377312654086540073888633031017463144383376 01
 26454073567994420539169324292133155134224789 1
 public exponent:
 65537
0  Validity: [From: Fri Jun 21 20:08:18 GMT 2002,
               To: Thu Mar 17 20:08:18 GMT 2005]
   Issuer: CN=jserver, OU=SWG, O=IBM, C=US
   SerialNumber: [    3d1387b2 ]
0]
   Algorithm: [MD5withRSA]
   Signature:
 0000: 54 DC B5 FA 64 C9 CD FE   B3 EF 15 22 3D D0 20 31  T...d......"=. 1
 0010: 99 F7 A7 86 F9 4C 82 9F   6E 4B 7B 47 18 2E C6 25  .....L..nK.G...%
 0020: 5B B2 9B 78 D8 76 5C 82   07 95 DD B8 44 62 02 62  [..x.v\.....Db.b
 0030: 60 2A 0A 6D 4F B9 0A 98   14 27 E9 BB 1A 84 8A D1  `*.mO....'......
 0040: C2 22 AF 70 9E A5 DF A2   FD 57 37 CE 3A 63 1B EB  .".p.....W7.:c..
 0050: E8 91 98 9D 7B 21 4A B5   2C 94 FC A9 30 C2 74 72  .....!J.,...0.tr
 0060: 95 01 54 B1 29 E7 F8 9E   6D F3 B5 D7 B7 D2 9E 9B  ..T.)...m.......
 0070: 85 D8 E4 CF C2 D5 3B 64   F0 07 17 9E 1E B9 2F 79  ......;d....../y
0]
 X509TrustManagerImpl: Certificate [
 [
   Version: V3
   Subject: CN=jserver, OU=SWG, O=IBM, C=US
   Signature Algorithm: MD5withRSA, OID = 1.2.840.113549.1.1.4
0  Key:  IBMJCE RSA Public Key:
 modulus:
 100949966922395090747968287561185391075683695663138899555389506
686622953008589748001058216362638201577071902071311277365773252660799
 128781182947273802312699983556527878615792292244995317112436562 49
148990438188426511935503773126540865400738886330310174631443833 7601
 26454073567994420539169324292133155134224 7891
 public exponent:
 65537
0  Validity: [From: Fri Jun 21 20:08:18 GMT 2002,
               To: Thu Mar 17 20:08:18 GMT 2005]
   Issuer: CN=jserver, OU=SWG, O=IBM, C=US
   SerialNumber: [    3d1387b2 ]
0]
   Algorithm: [MD5withRSA]
   Signature:
 0000: 54 DC B5 FA 64 C9 CD FE   B3 EF 15 22 3D D0 20 31  T...d......"=. 1
 0010: 99 F7 A7 86 F9 4C 82 9F   6E 4B 7B 47 18 2E C6 25  .....L..nK.G...%
 0020: 5B B2 9B 78 D8 76 5C 82   07 95 DD B8 44 62 02 62  [..x.v\.....Db.b
 0030: 60 2A 0A 6D 4F B9 0A 98   14 27 E9 BB 1A 84 8A D1  `*.mO....'......
 0040: C2 22 AF 70 9E A5 DF A2   FD 57 37 CE 3A 63 1B EB  .".p.....W7.:c..
 0050: E8 91 98 9D 7B 21 4A B5   2C 94 FC A9 30 C2 74 72  .....!J.,...0.tr
 0060: 95 01 54 B1 29 E7 F8 9E   6D F3 B5 D7 B7 D2 9E 9B  ..T.)...m.......
 0070: 85 D8 E4 CF C2 D5 3B 64   F0 07 17 9E 1E B9 2F 79  ......;d....../y
0]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM/
9.38.48.108,port=8880,localport=2238]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM/
9.38.48.108,port=8880,localport=2239]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM/
9.38.48.108,port=8880,localport=2240]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM/
9.38.48.108,port=8880,localport=2241]]
```

Tracing security

The classes which implement WebSphere Application Server security are:
- com.ibm.ws.security.*
- com.ibm.websphere.security.*
- com.ibm.WebSphereSecurityImpl.*
- SASRas

To view detailed information on the runtime behavior of security, enable trace on the following components and review the output:

- `com.ibm.ws.security.*=all=enabled:com.ibm:WebSphereSecurityImpl.*=all=enabled:com.ibm.websphere.security.*=all=enabled`. This trace statement collects the trace for the security runtime.
- `com.ibm.ws.console.security.*=all=enabled`. This trace statement collects the trace for the security center GUI.
- `SASRas=all=enabled`. This trace statement collects the trace for SAS (low-level authentication logic).

Fine tuning SASzSAS traces:

If a subset of classes need to be traced for the SASzSAS/CSIv2 component, a system property can be specified with the class names comma separated: `com.ibm.CORBA.securityTraceFilter=SecurityConnectionInterceptorImpl, VaultImpl, ...`

Fine tuning Security traces:

If a subset of packages need to be traced, specify a trace specification more detailed than `com.ibm.ws.security.*=all=enabled`. For example, to trace just dynamic policy code, you can specify `com.ibm.ws.security.policy.*=all=enabled`. To disable dynamic policy trace, you can specify `com.ibm.ws.security.policy.*=all=disabled`.

Configuring CSIv2 or SASzSAS Trace Settings

Situations arise where reviewing trace for the CSIv2 or SASzSAS authentication protocols can assist in troubleshooting difficult problems. This section decribes how to enable to CSIv2/SASzSAS trace.

Enabling Client-Side CSIv2/SASzSAS Trace

To enable CSIv2 and SASzSAS trace on a pure client, the following steps need to be taken:

- Edit the file TraceSettings.properties in the **`/WebSphere/AppServer/properties` directory**.
- In this file, change `traceFileName=` to point to the path in which you want the ouput file created. Make sure you put a double backslash (\\) between each subdirectory. For example, `traceFileName=c:\\WebSphere\\AppServer\\logs\\sas_client.log`
- In this file, add the trace specification string: `SASRas=all=enabled`. Any additional trace strings can be added on separate lines.
- Point to this file from within your client application. On the Java command line where you launch the client, add the following system property: `-DtraceSettingsFile=TraceSettings.properties`.

  Note: Do not give the fully qualified path to the `TraceSettings.properties` file. Make sure that the `TraceSettings.properties` file is in your classpath.

Enabling Server-Side CSIv2/SASzSAS Trace

To enable SASzSAS trace in an application server, complete the following:

- Add the trace specification, `SASRas=all=enabled`, to the `server.xml` file or add it to the Trace settings within the WebConsole GUI.
- Typically it is best to also trace the authorization security runtime in addition to the authentication protocol runtime. To do this, use the following two trace specifications in combination: `SASRas=all=enabled:com.ibm.ws.security.*=all=enabled`.
- When troubleshooting a connection type problem, it is beneficial to trace both SASzSAS/CSIv2 and the ORB. To do this, use the following three trace specifications: `SASRas=all=enabled:com.ibm.ws.security.*=all=enabled:ORBRas=all=enabled`.
- In addition to adding these trace specifications, for ORB trace there are a couple of system properties that also need to be set. Go to the ORB settings in the GUI and add the following two properties: `com.ibm.CORBA.Debug=true` and `com.ibm.CORBA.CommTrace=true`.

CSIv2 CORBA Minor Codes

Whatever exceptions might occur within the security code on either the client or server, the eventual exception will become a CORBA exception. So any exception that occurs gets ″wrapped″ by a CORBA exception, because the CORBA architecture is used by the security service for its own inter-process communication. CORBA exceptions are generic, and indicate a problem in communication between two components. CORBA minor codes are more specific, and indicate the underlying reason that a component could not complete a request.

The following shows the CORBA Minor codes which a client can expect to receive after executing a security-related request such as authentication. It also includes the CORBA exception type that the minor code would appear in.

The following exception shows an example of a CORBA exception where the minor code is 49424300. From the table below, this minor code indicates Authentication Failure. Typically, a descriptive message is also included in the exception to assist in troubleshooting the problem. Here, the detailed message is ″Exception caught invoking authenticateBasicAuthData from SecurityServer for user jdoe. Reason: com.ibm.WebSphereSecurity.AuthenticationFailedException″ which indicates that the authentication failed for user ″jdoe″.

The completed field in the exception indicates whether the method was completed or not. In the case of a `NO_PERMISSION`, the method should never get invoked, so it will always be ″completed:No″. Other exceptions which are caught on the server side could have a completed status of ″Maybe″ or ″Yes″.

```
org.omg.CORBA.NO_PERMISSION: Caught WSSecurityContextException in
WSSecurityContext.acceptSecContext(),
reason: Major Code[0] Minor Code[0] Message[Exception caught invoking
authenticateBasicAuthData from SecurityServer for user jdoe.  Reason:
com.ibm.WebSphereSecurity.AuthenticationFailedException]  minor code: 49424300
completed: No

at com.ibm.ISecurityLocalObjectBaseL13Impl.PrincipalAuthFailReason.
map_auth_fail_to_minor_code(PrincipalAuthFailReason.java:83)
        at com.ibm.ISecurityLocalObjectBaseL13Impl.CSIServerRI.receive_request
               (CSIServerRI.java:1569)
        at com.ibm.rmi.pi.InterceptorManager.iterateReceiveRequest
              (InterceptorManager.java:739)
        at com.ibm.CORBA.iiop.ServerDelegate.dispatch(ServerDelegate.java:398)
        at com.ibm.rmi.iiop.ORB.process(ORB.java:313)
        at com.ibm.CORBA.iiop.ORB.process(ORB.java:1581)
        at com.ibm.rmi.iiop.GIOPConnection.doWork(GIOPConnection.java:1827)
        at com.ibm.rmi.iiop.WorkUnitImpl.doWork(WorkUnitImpl.java:81)
        at com.ibm.ejs.oa.pool.PooledThread.run(ThreadPool.java:91)
        at com.ibm.ws.util.CachedThread.run(ThreadPool.java:149)
```

The following table shows the CORBA Minor codes which a client can expect to receive after executing a security-related request such as authentication. It also includes the CORBA exception type that the minor code would appear in.

Minor code name	Minor code value (in hex)	Exception type (all in the package of org.omg.CORBA .*)	Minor code description	Retry performed (when authenticationRe tryEnabled=true)
AuthenticationFailed	49424300	NO_PERMISSION	This is a generic authentication failed error. It does not give any details about whether the userid or password is invalid. Some registries can choose to use this type of error code, others might choose to use the next three types which are more specific.	Yes

InvalidUserid	49424301	NO_PERMISSION	This occurs when the registry returns bad userid.	Yes
InvalidPassword	49424302	NO_PERMISSION	This occurs when the registry returns bad password.	Yes
InvalidSecurityCredentials	49424303	NO_PERMISSION	This is a generic error indicating that the credentials are bad for whatever reason. It could be that they don't have the right attributes set.	Yes, if client has BasicAuth credential (token based credential was rejected in the first place).
InvalidRealm	49424304	NO_PERMISSION	This occurs when the REALM in the token received from the client does not match the server's current realm.	No
ValidationFailed	49424305	NO_PERMISSION	A validation failure occurs when a token is sent from the client or server to a target server but the token format or the expiration is invalid.	Yes, if client has BasicAuth credential (token based credential was rejected in the first place).
CredentialTokenExpired	49424306	NO_PERMISSION	This is more specific about why the validation failed. In this case, the token has a absolute lifetime, and this lifetime has expired. Therefore, it is no longer a valid token and cannot be used.	Yes, if client has BasicAuth credential (token based credential was rejected in the first place).
InvalidCredentialToken	49424307	NO_PERMISSION	This is more specific about why the validation failed. In this case, the token cannot be decrypted or the data within it is not readable.	Yes, if client has BasicAuth credential (token based credential was rejected in the first place).
SessionDoesNotExist	49424308	NO_PERMISSION	This indicates that the CSIv2 session does not exist on the server. Typically, a retry occurs automatically and will successfully create a new session.	Yes
SessionConflictingEvidence	49424309	NO_PERMISSION	This indicates that a session already exists on the server which matches the context_id sent over by the client, however, the information provided by the client for this EstablishContext message is different from the information originally provided to establish the session.	Yes
SessionRejected	4942430A	NO_PERMISSION	This indicates that the session referenced by the client has been previously rejected by the server.	Yes
SecurityServerNotAvailable	4942430B	NO_PERMISSION	This error occurs when the server cannot contact the security server (whether local or remote) in order to authenticate or validate.	No
InvalidIdentityToken	4942430C	NO_PERMISSION	This error indicates that identity cannot be obtained from the identity token when Identity Assertion is enabled.	No

IdentityServerNotTrusted	4942430D	NO_PERMISSION	This indicates that the server id of the sending server is not on the target server's trusted principal list.	No
InvalidMessage	4942430E	NO_PERMISSION	This indicates that the CSIv2 message format is invalid for the receiving server.	No
AuthenticationNotSupported	49421090	NO_PERMISSION	This error occurs when a mechanism does not support authentication (very rare).	No
InvalidSecurityMechanism	49421091	NO_PERMISSION	This is used to indicate that the specified security mechanism is not known.	No
CredentialNotAvailable	49421092	NO_PERMISSION	This indicates a credential is not available when it is required.	No
SecurityMechanismNotSupported	49421093	NO_PERMISSION	This error occurs when a security mechanism specified in the CSIv2 token is not implemented on the server.	No
ValidationNotSupported	49421094	NO_PERMISSION	This error occurs when a mechanism does not support validation (such as LocalOS). This error should not occur since the LocalOS credential is not a forwardable credential, therefore, validation should never need to be called on it.	No
CredentialTokenNotSet	49421095	NO_PERMISSION	This is used to indicate the token inside the credential is null.	No
ServerConnectionFailed	494210A0	COMM_FAILURE	This error is used when a connection attempt fails.	Yes (via ORB retry)
CorbaSystemException	494210B0	INTERNAL	This is a generic CORBA specific exception in system code.	No
JavaException	494210B1	INTERNAL	This is a generic error that indicated an unexpected Java exception occurred.	No
ValueIsNull	494210B2	INTERNAL	This is used to indicate that a value or parameter passed in was null.	No
EffectivePolicyNotPresent	494210B3	INTERNAL	This indicates that an effective policy object for CSIv2 is not present. This object is used to determine what security configuration features have been specified.	No
NullPointerException	494210B4	INTERNAL	This is used to indicate that a NullPointerException was caught in the runtime.	No
ErrorGettingClassInstance	494210B5	INTERNAL	This indicates a problem loading a class dynamically.	No
MalFormedParameters	494210B6	INTERNAL	This indicates parameters are not valid.	No
DuplicateSecurityAttributeType	494210B7	INTERNAL	A duplicate credential attribute has been specified during the set_attributes operation.	No
MethodNotImplemented	494210C0	NO_IMPLEMENT	A method invoked has not been implemented.	No

GSSFormatError	494210C5	BAD_PARAM	This indicates that a GSS encoding or decoding routine has thrown an exception.	No
TagComponentFormatError	494210C6	BAD_PARAM	This indicates that a tag component cannot be read properly.	No
InvalidSecurityAttributeType	494210C7	BAD_PARAM	This indicates an attribute type specified during the set_attributes operation is an invalid type.	No
SecurityConfigError	494210CA	INITIALIZE	A problem exists between the client and server configuration.	No

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

JSP engine troubleshooting tips

If you are having difficulty using the JSP engine, try these steps:

1. Determine whether other resources such as `.html` files or servlets are being requested and displayed correctly. If they are not, the problem probably lies at a deeper level, such as with the HTTP server.
2. If other resources are being displayed correctly, determine whether the JSP engine has started normally:
   - Browse the logs of the server hosting the JSP files you are trying to access. A message such as JSP 1.2 Processor: init a server log file indicates that the JSP engine has started normally. If the JSP processor fails to load, you may see a message such as **`Did not realize init() exception thrown by servlet JSP 1.2 Processor`** in the server log files.
3. If the JSP engine has started normally, the problem may be with the JSP file itself.
   - Copy a simple JavaServer Pages file file (such as the WebSphere Application Server sample ″HelloHTML.jsp″) to the Web application's document root and attempt to serve it.
   - If that works, examine the target application server's server log files for invalid JSP directive syntax . Errors similar to the following in a browser indicate this kind of problem: **`Message: /`*`jspname`*`.jsp(9,0) Include: Mandatory attribute page missing`**. This example indicates that line 9, column 0 of the named JavaServer Pages file is missing a mandatory page attribute. Similar messages are displayed for other syntax errors.
   - Examine the target application server's server log files files for problems with invalid Java syntax. Errors similar to **`Message: Unable to compile class for JSP`** in a browser indicate this kind of problem.

     The error message output from the Javac compiler will be found in the server log files. It might look like:

     ```
     C:\WASROOT\temp\ ... test.war\_myJsp.java:14: Duplicate variable declaration:
           int myInt was int myInt
           int myInt = 122; String myString = ″number is 122″;
           static int myStaticInt=22;
           int myInt=121;
                               ^
        1 error
     ```

     Correct the error in the JSP file and retry the file.

If none of these steps solves the problem, check to see if the problem is identified and documented using the links in Diagnosing and fixing problems: Resources for learning.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Object request broker component troubleshooting tips

Enabling tracing for the Object Request Broker component

The object request broker (ORB) service is one of the WebSphere Application Server run time services. Tracing of messages sent and received by the ORB is a useful starting point for troubleshooting the ORB service. You can selectively enable or disable tracing of ORB messages for each server in a WebSphere Application Server installation, and for each application client.

For instructions on how to set trace controls so that tracing occurs for the ORB subcomponent, see Setting trace controls for IBM service.

Log files and messages associated with Object Request Broker

For a summary of how messages get routed in WebSphere Application Server, see Message routing.

Java packages containing the Object Request Broker service

The ORB service resides in the following Java packages:
- com.ibm.com.CORBA.*
- com.ibm.rmi.*
- com.ibm.ws.orb.*
- com.ibm.ws.orbimpl.*
- com.ibm.ws390.orb.*
- org.omg.CORBA.*
- javax.rmi.CORBA.*

JAR files that contain the previously mentioned packages include:
- `*java_install_dir*/jre/lib/ext/ibmorb.jar`
- `*java_install_dir*/jre/lib/ext/iwsorbutil.jar`
- `*install_dir*/lib/iwsorb.jar`
- `*install_dir*/lib/runtimews390.jar`
- `*install_dir*/lib/bootstrap.jar`

Tools used with Object Request Broker

The tools used to compile Java remote interfaces to generate language bindings used by the ORB at runtime reside in the following Java packages:
- com.ibm.tools.rmic.*
- com.ibm.idl.*

The JAR file that contains the packages is `*install_dir*/java/lib/ibmtools.jar`.

Object Request Broker properties

The ORB service requires a number of ORB properties for correct operation. It is not necessary for most users to modify these properties, and it is recommended that only your system administrator modify them when required. Consult IBM Support personnel for assistance. The properties reside in the `orb.properties` file, located at *`install_dir`*`/java/jre/lib/orb.properties`.

If none of these steps fixes your problem, check to see if the problem has been identified and documented by looking at the available online support (hints and tips, technotes, and fixes).

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Web services security troubleshooting tips

Troubleshooting Web services security is best done by reviewing the configurations in the Assembly Toolkit so that you can match up the client and server request and response configurations. These configurations must match. A client request sender configuration must match a server request receiver configuration. For encryption to successfully occur, the public key of the receiver must be exported to the sender and this key must be configured properly in the encryption information. For authentication, you must specify the method used by the client in the login mapping of the server. Also, you must correctly specify the actor URI at each point in the configuration with the same URI string. The following is a list of generic troubleshooting steps that you can perform. A listing of specific symptoms and solutions is provided after these steps.

- General troubleshooting tips
  - Verify that the client security extensions and server security extensions match on each downstream call for the following:
    - Request Sender and Request Receiver
    - Response Sender and Response Receiver
  - Verify that when the **Add Created Time Stamp** option is enabled on the client-side that the server has the **Add Received Time Stamp** option configured. You must configure the security extensions in the Assembly Toolkit
  - Verify that the client security bindings and server security bindings are correctly configured. When the client authentication method is signature, make sure that the server has a login mapping to handle it. When the client uses the public key `cn=Bob,o=IBM,c=US` to encrypt the body, verify that this Subject is a personal certificate in the server keystore so that it can decrypt the body with the private key. You can configure the security bindings using either the Assembly Toolkit or the WebSphere Application Server Administrative Console.
  - Check the `SystemOut.log` file in the `${`*`USER_INSTALL_ROOT`*`}/logs/`*`server1`* directory (*server1* changes depending upon the server name) for messages that might provide information about the problem.
  - Enable trace for Web Services Security by using the following trace specification:
    ```
    com.ibm.xml.soapsec.*=all=enabled:com.ibm.ws.webservices.*=all=enabled:
    com.ibm.wsspi.wssecurity.*=all=enabled:com.ibm.ws.security.*=all=enabled:
    SASRas=all=enabled
    ```
    Type the previous three lines as one continuous line.
- **Symptom:** WSEC5061E: The SOAP Body is not signed

  Solution: This error usually occurs whenever the SOAP security handler does not load properly, thus causing the SOAP body not to be signed. The SOAP security handler is typically the first validation that occurs on the server-side, so a multitude of problems can cause this message to display. The error might be caused by invalid actor URI configurations. You can configure the actor URI at the following locations within the Assembly Toolkit:

  From the Web Services Client Editor within the Assembly Toolkit for client configurations:
  - Click **Security Extensions > Client Service Configuration Details** and indicate the actor information in the **ActorURI** field.

  – Click **Security Extensions > Request Sender Configuration section > Details** and indicate the actor information in the **Actor** field.

  From the Web Services Editor within the Assembly Toolkit for server configurations:

  – Click **Security Extensions > Server Service Configuration** section. Verify that the Actor URI has the same actor string as the client-side.
  – Click **Security Extensions > Response Sender Service Configuration Details > Details** and indicate the actor information in the **Actor** field.

  The actor information on both the client and server must refer to the same exact string. When the actor fields on the client and server match, the request or response is acted upon instead of being forwarded downstream. The actor fields might be different when you have Web services acting as a gateway to other Web services. However, in all other cases, verify that the actor information matches on the client and server. When the Web services implementation is acting as a gateway and it does not have the same actor configured as the request passing through the gateway, this Web services implementation does not process the message from the client. Instead, it sends the request downstream. The downstream process that contains the correct actor string processes the request. The same situation occurs for the response. Therefore, it is important that you verify that the appropriate client and server actor fields are synchronized.

  Additionally, the error can appear when you do not specify that the body is to be signed in the client configuration. To sign the body part of the message using the Web Service Client Editor in the Assembly Toolkit, click **Security Extensions > Request Sender Configuration > Integrity** and select the message parts to be signed.

- **Symptom:** WSEC5075E: No security token found which satisfies any one of AuthMethods

  Solution: Verify that the client and server login configuration information matches in the security extensions. Also, verify that the client has a valid login binding and that the server has a valid login mapping in the security bindings. You can check this by looking at the following locations in the Assembly Toolkit:

  From the Web Services Client Editor within the Assembly Toolkit for client configurations:

  – Click **Security Extensions > Request Sender Configuration > Login Config** and check the authentication method.
  – Click **Port Binding > Security Request Sender Binding Configuration > Login Binding** and check the authentication method and other parameters.

  From the Web Services Editor within the Assembly Toolkit for server configurations:

  – Click **Security Extensions > Request Receiver Service Configuration Details > Login Config** and check the authentication method.
  – Click **Binding Configurations > Request Receiver Binding Configuration Details > Login Mapping** and check the authentication method and other parameters.

  Also, make sure that the actor URI specified on the client and server matches. You can configure the actor URI at the following locations within the Assembly Toolkit:

  From the Web Services Client Editor within the Assembly Toolkit for client configurations:

  – Click **Security Extensions > Client Service Configuration Details** and indicate the actor information in the **ActorURI** field.
  – Click **Security Extensions > Request Sender Configuration section > Details** and indicate the actor information in the **Actor** field.

  From the Web Services Editor within the Assembly Toolkit for server configurations:

  – Click **Security Extensions > Server Service Configuration** section. Make sure that the **Actor URI** field has the same actor string as the client side.

  – Click **Security Extensions > Response Sender Service Configuration Details > Details** and indicate the actor information in the **Actor** field.

- **Symptom:** WSEC5094E: No UsernameToken of trusted user was found or the login failed for the user while TrustMode is BasicAuth.

  Solution: This situation occurs when you have IDAssertion configured in the login config as the authentication method. On the sending Web Service, configure a trusted basic authentication entry in the login binding. Then, on the server side, verify that the trusted ID evaluator has a property set that contains the user name of this basic authentication entry. To configure the client for IDAssertion, see:

  – Configuring the client for identity assertion authentication: specifying the method

  – Configuring the client for identity assertion authentication: collecting the authentication information

  To configure the server for IDAssertion, see:

  – Configuring the server to handle identity assertion authentication

  – Configuring the server to validate identity assertion authentication information

- **Symptom:** The following authorization error occurs with UNAUTHENTICATED as the security name: SECJ0053E: Authorization failed for /UNAUTHENTICATED while invoking (Home)com/ibm/wssvt/tc/pli/ejb/Beneficiary findBeneficiaryBySsNo(java.lang.String):2 securityName: /UNAUTHENTICATED;accessID: null is not granted any of the required roles: AgentRole

  Solution: This situation is usually due to a login config not being configured or Web services Security is not configured from a client to a server. When the request arrives at the server and authentication information is not received, the UNAUTHENTICATED user is set on the thread. Authorization returns this error if there are any roles assigned to the resource except for the special ″Everyone″ role, which allows access by anyone. If the client successfully authenticates to an EJB file but the EJB file calls a downstream EJB file that is not configured with Web services security or transport security, such as HTTP user ID and password, an error can occur for this downstream request. Using the Assembly Toolkit, verify that the enterprise archive (EAR) file for both client and server has the correct security extensions and security bindings. For more information, see:

  – Configuring the client security bindings using the Assembly Toolkit

  – Configuring the security bindings on a server acting as a client using the administrative console

  – Configuring the server security bindings using the Assembly Toolkit

  – Configuring the server security bindings using the administrative console

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter 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
500 Columbus Avenue
Thornwood, New York 10594 USA

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
522 South Road
Poughkeepsie, NY 12601-5400
USA
Attention: Information Requests

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

Trademarks and service marks

The following terms are trademarks of IBM Corporation in the United States, other countries, or both:

- AIX
- CICS
- Cloudscape
- DB2
- DFSMS
- Everyplace
- iSeries
- IBM
- IMS
- Informix
- iSeries
- Language Environment
- MQSeries
- MVS
- OS/390