Sun Gathering Debug Data for Sun Java System Application Server

Sun Java™ Enterprise System Technical Note

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 820–2974–10
October 2007

081023@21288

Contents

◆ ◆ ◆ CHAPTER 1

Introduction

This technical note describes how to collect data that the Sun Support Center requires in order to debug problems with a Sun Java™ System Application Server system. By collecting this data before you open a Service Request, you can reduce substantially the time needed to analyze and resolve the problem. For more information on how this document and associated scripts can help you in better dealing with Application Server problems, see:

`http://www.sun.com/service/gdd/index.xml`

This document is intended for anyone who needs to open a Service Request about Application Server with the Sun Support Center.

This chapter contains the following sections:

After reading this chapter, proceed to Chapter 2, "Gathering Debug Data."

1.1 Technical Note Revision History

Version	Date	Description of Changes
10	August 2007	Initial release of this technical note.

1.2 About This Technical Note

This document covers the following versions of Sun Java System Application Serveron the Solaris™ Operating System, HP-UX, Linux, and Microsoft Windows platforms:

- Sun Java System Application Server 9.1
- Sun Java System Application Server 9.0
- Sun Java System Application Server 8.2
- Sun Java System Application Server 8.1

You can use this document in all types of environments, including test, pre-production, and production. Verbose debugging is not used (to reduce performance impact), except when it is deemed necessary. At the same time, it is possible that the problem could disappear when you configure logging for debug mode. However, this is the minimum to understand the problem. In the majority of cases, the debug data described in this document is sufficient to analyze the problem.

This document does not provide workarounds, techniques, or tools to analyze debug data. It provides some troubleshooting, but you should not use this guide as an approach to troubleshooting Application Server problems.

If your problem does not conveniently fit into any of the specific categories, supply the general information described in "2.1 What Application Server Debug Data Should You Collect?" on page 11 and clearly explain your problem.

If the information you initially provide is not sufficient to find the root cause of the problem, Sun will ask for more details, as needed.

1.2.1 Prerequisites for Collecting Application Server Debug Data

The prerequisites for collecting debug data for Application Server are as follows:

- Make sure you have `superuser` privileges.
- For the Solaris OS platform, obtain the `appserver_8_dbhang` and `pkg_app` scripts from the following location:

  `http://www.sun.com/bigadmin/scripts/indexSjs.html`
- On the Windows platform, download the free Debugging Tools for Windows to help in analyzing process hang problems. The debugger Dr. Watson is not useful for process hang problems because it cannot generate a crash dump on a running process. Download the free Debugging Tools from the following location:

  `http://www.microsoft.com/whdc/devtools/debugging/default.mspx`

  Install the last version of Debugging Tools and the OS Symbols for your version of Windows. Also, you must add the environment variable `NT_SYMBOL_PATH`.

Use the command `drwtsn32 -i` to select Dr. Watson as the default debugger. Use the command `drwtsn32`, check all options, and choose the path for crash dumps.

1.2.2 Variables Used in This Technical Note

The following describes the variables used in the procedures in this document. Gather the values of the variables if you don't already know them before you try to do the procedures.

- *appserv-pid*: Process ID of a Application Server daemon.
- *windbg-root*: The directory on the Windows Application Server machine dedicated to holding the WinDebugger program, and configuration, maintenance, and information files.
- *server-root*: The directory on the Application Server machine dedicated to holding the server program, configuration, maintenance, and information files. The default location for the Solaris OS version of the Sun Java System Application Server is `/opt/SUNWappserver`. See "To Obtain the *server-root* Variable" on page 7 for instructions on determining the *server-root*.

▼ To Obtain the *server-root* Variable

- **Use one of the following commands to obtain the value of the *server-root* variable for Application Server.**

  Solaris `pkgparam –v SUNWappserver`

  Linux `rpm -q --qf '%{INSTALLPREFIX}' sun-application-server`

  Windows Look in the `C:\windows\system32\appservregistry.inf` file.

1.3 Overview of Collecting Debug Data for Application Server

Collecting debug data for a Application Server problem involves these basic operations:

1. Collecting basic problem and system information.
2. Collecting specific problem information (installation problem, process hang, process crash, and so on).
3. Creating a `tar.gz` or `zip` file of all the information and uploading it for the Sun Support Center.
4. Creating a Service Request with the Sun Support Center.

1.4 Creating a Service Request with the Sun Support Center

When you create a Service Request with the Sun Support Center, either online or by phone, provide the following information:

- A clear problem description
- Details of the state of the system, both before and after the problem started
- Impact on end users
- All recent software and hardware changes
- Any actions already attempted
- Whether the problem is reproducible; when reproducible, provide the detailed test case
- Whether a pre-production or test environment is available
- Name and location of the archive file containing the debug data

Upload your debug data archive file to one of the following locations:

- `http://supportfiles.sun.com/upload`
- `https://supportfiles.sun.com/upload`

For more information on how to upload files to this site, see:
`http://supportfiles.sun.com/show?target=faq`

Note – When opening a Service Request by phone with the Dispatch Team, provide a summary of the problem, then give the details in a text file named `Description.txt`. Be sure to include `Description.txt` in the archive along with the rest of your debug data.

1.5 Reporting Problems

Use the following email aliases to report problems with this document and its associated scripts:

- To provide feedback: `gdd-feedback@sun.com`
- To report problems: `gdd-issue-tracker@sun.com`

1.6 Accessing Sun Resources Online

The docs.sun.com web site enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for a specific book title or subject. Books are available as online files in PDF and HTML formats. Both formats are readable by assistive technologies for users with disabilities.

To access the following Sun resources, go to http://www.sun.com:

- Downloads of Sun products
- Services and solutions
- Support (including patches and updates)
- Training
- Research
- Communities (for example, Sun Developer Network)

1.7 Third-Party Web Site References

Third-party URLs are referenced in this document and provide additional, related information.

Note – Sun is not responsible for the availability of third-party web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

1.8 Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. To share your comments, go to http://docs.sun.com and click Send Comments. In the online form, provide the full document title and part number. The part number is a 7-digit or 9-digit number that can be found on the book's title page or in the document's URL. For example, the part number of this book is XXX-XXXX-XX.

◆ ◆ ◆ CHAPTER 2

Gathering Debug Data

This chapter describes a number of procedures to help you gather debugging information that you can use to solve problems you may encounter when using Application Server. This information is also useful if you contact the Sun Support Center.

This chapter includes the following topics:

2.1 What Application Server Debug Data Should You Collect?

This section describes the kinds of debug data that you need to provide based on the kind of problem you are experiencing.

This section contains the following tasks:

▼ To Collect Required Debug Data for Any Application Server Problem

All problems described in this technical note need basic information collected about when the problem occurred and about the system having the problem. Use this task to collect that basic information.

1. **Note the day(s) and time(s) the problem occurred.**

2. **Provide a graphical representation of your deployment. Include all hosts and IP addresses, host names, operating system versions, role they perform, and other important systems such as load balancers, firewalls, and so forth.**

3. **Note the version of the operating system.**

Solaris OS	`uname -a`
HP-UX	`uname -r`
Linux	`uname -a`
Windows	`C:\Program Files\Common files\Microsoft Shared\MSInfo\msinfo32.exe /report C:\report.txt`

4. **Note the patch level.**

Solaris OS	`patchadd -p`
HP-UX	`swlist`
Linux	`rpm -qa`
Windows	Already provided in the `C:\report.txt` file above.

5. **Note the version of Application Server.**

   If a configured JDK is used instead of the default JDK (usually installed in `$AS_HOME/jdk`) then provide the output of the command `java -version`. The Application Server version can be shown with the `asadmin` version command:

   ```
   asadmin version --verbose=true
   ```

6. **Create a tar file of the Application Server main domain configuration directory and their node agents.**

   This step varies depending on the type of Application Server configuration you are debugging:

   - Domain Administration Server (DAS)
   - Node Agent (lightweight agent that is required on every machine that hosts server instances, including the machine that hosts the DAS)
   - Instance (an individual Application Server instance, either clustered or standalone)

Note – The locations referenced in these instructions are the default locations for Application Server and are provided for illustrative purposes only. If you have started a domain with the `--domaindir` argument or a node agent with the `--agentdir` argument, be sure to use your custom directory paths for *server-root*.

- **DAS**

  On the main domain machine:

Solaris, HP-UX, Linux	`cd` *server-root*`/domains/`*domainname*`/config`
	Create a tar file of the *server-root*`domains/`*domainname*`/config` directory.
Windows	`cd` *server-root*`\config`
	Create a zip file of the *server-root*`\config` directory.

- **Node Agents**

  On the node agent machine:

Solaris, HP-UX, Linux	`cd` *server-root*`/nodeagents/`*nodeagentname*`/agent/config`
	Create a tar file of the *server-root*`/nodeagents/`*nodeagentname*`/agent/config` directory.
Windows	`cd` *server-root*`\nodeagents\`*nodeagentname*`\agent\config`
	Create a zip file of the *server-root*`\nodeagents\`*nodeagentname*`\agent\config` directory.

- **Instance**

  On the instance machine:

Solaris, HP-UX, Linux	`cd` *server-root*`/nodeagents/`*nodeagentname*`/`*instancename*`/config`
	Create a tar file of the *server-root*`/nodeagents/`*nodeagentname*`/`*instancename*`/config` directory.
Windows	`cd` *server-root*`\nodeagents\`*nodeagentname*`\`*instancename*`\config`
	Create a zip file of the *server-root*`\nodeagents\`*nodeagentname*`\`*instancename*`\config` directory.

7 Get the log files from the Application Server main domain configuration directory and their node agents.

- **DAS**

On the main domain machine:

Solaris, HP-UX, Linux	`cd` *server-root*`/domains/`*domainame*`/logs` Get the `access.log` and `server.log` files from the *server-root*`/domains/`*domainame*`/logs` directory.
Windows	`cd` *server-root*`\domains\`*domainname*`\logs` Get the `access.log` and `server.log` files from the *server-root*`\domains\`*domainname*`\logs` directory.

- **Node Agents**

  On the node agent machine:

Solaris, HP-UX, Linux	`cd` *server-root*`/nodeagents/`*nodeagentname*`/agent/logs` Get the `server.log` file from the *server-root*`/nodeagents/`*nodeagentname*`/agent/logs` directory.
Windows	`cd` *server-root*`\nodeagents\`*nodeagentname*`\agent\logs` Get the `server.log` file from the *server-root*`\nodeagents\`*nodeagentname*`\agent\logs` directory.

- **Instance**

  On the instance machine:

Solaris, HP-UX, Linux	`cd` *server-root*`/nodeagents/`*nodeagentname*`/`*instancename*`/logs` Get the `server.log` file from the *server-root*`/nodeagents/`*nodeagentname*`/`*instancename*`/logs` directory.
Windows	`cd` *server-root*`\nodeagents\`*nodeagentname*`\`*instancename*`\logs` Get the `server.log` file from the *server-root*`\nodeagents\`*nodeagentname*`\`*instancename*`\logs` directory.

Tip – If possible, provide just the relevant extracts of log files for the same time period that show the problem, with sufficient context to see what else was occurring during the error occurrence and shortly before. Thus for relatively short log files, send the entire log file, whereas for long-running (hence large) log files, an extract might be more appropriate. In either case, be sure to include all the material from the time of the error as well as at least some lead-in logging from before the error apparently occurred.

▼ To Collect Debug Data for Application Server Installation Problems

Follow these steps if you are unable to complete the installation or if you get a "failed" installation status for Application Server.

1. **Gather the general system information, as explained in "To Collect Required Debug Data for Any Application Server Problem" on page 11.**

2. **Specify what kind of installation you have on your server:**

   - Package-based or file-based
   - First-time (clean) or upgrade

3. **Gather the installation logs.**

   - **Application Server 9.1**

Solaris	`/var/sadm/install/logs` The log file names start with `Install_Application_Server_*`*datetime* and `Sun_Java_System_Application_Server_*`*datetime*, where date and time correspond to the failing installing (for example, `12161532`).
HP-UX and Llinux	`/var/sadm/install/logs` The log file names start with `Install_Application_Server_*`*datetime* and `Sun_Java_System_Application_Server_*`*datetime*, where date and time correspond to the failing installing (for example, `12161532`).
Windows	`C:\DocumentsandSettings\`*current-user*`\LocalSettings\Temp`

The log file names start with `MSI*.log` (usually a text file). The asterisk (*) represents a random number in the `Temp` directory for each MSI based setup.

- **Application Server 8.1 2005Q1**

  Solaris
  ```
  truss -ealf -rall -wall -vall -o /tmp/install-appserver.truss
  ./sjsas_ee-8_1_02_2005Q2-solaris-sparc.bin
  ```

  HP-UX
  ```
  tusc -v -fealT -rall -wall -o /tmp/install-appserver.tusc
  ./sjsas_ee-8_1_02_2005Q2-hpux.bin
  ```

  Linux
  ```
  strace -fv -o /tmp/install-appserver.strace
  ./sjsas_ee-8_1_02_2005Q2-linux.bin
  ```

  Windows Use `DebugView` tool. You can download this tool from `http://www.sysinternals.com/Utilities/DebugView.html`.

▼ To Collect Debug Data for Application Server Startup Problems

1. **Check the product Troubleshooting Guides on `http://docs.sun.com`.**
   - *Sun Java System Application Server Enterprise Edition 8.1 2005Q1 Troubleshooting Guide*
   - *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Troubleshooting Guide*
   - *Sun Java System Application Server Enterprise Edition 8.2 Troubleshooting Guide*
   - *Sun Java System Application Server 9.1 Troubleshooting Guide*

2. **If none of the above guides solves your problem, follow the instructions in "To Collect Required Debug Data for Any Application Server Problem" on page 11, then gather the following information:**

   a. **Run the `netstat` command and save the output.**

   Solaris, HP-UX, Linux
   ```
   netstat -an | grep application_server_port
   ```

   Windows
   ```
   netstat -an
   ```

   b. **Identify which part of the Application Server is not starting: node agent, DAS, or instance.**

   c. **Run the following command on the Web Server start script and save the resulting file.**

   Solaris *Node Agent*

   ```
   truss -eafl -wall -vall -rall -o /tmp/as-start.truss \
   ./asadmin start-nodeagent -u admin nodeagentname
   ```

Instance

```
truss -eafl -wall -vall -rall -o /tmp/instance-start.truss \
./asadmin start-instance -u admin instancename
```

DAS

```
truss -eafl -wall -vall -rall -o /tmp/das-start.truss \
./asadmin start-domain -u admin domainname
```

HP-UX *Node Agent*

```
tusc -v -fealT -rall -wall -o /tmp/as-start.tusc \
./asadmin start-nodeagent -u admin nogeagentname
```

Instance

```
tusc -v -fealT -rall -wall -o /tmp/as-start.tusc \
./asadmin start-instance -u admin instancename
```

DAS

```
tusc -v -fealT -rall -wall -o /tmp/as-start.tusc \
./asadmin start-domain -u admin domainname
```

Linux *Node Agent*

```
strace -fv -o /tmp/as-start.strace ./asadmin \
start-nogeagent -u admin nogeagentname
```

Instance

```
strace -fv -o /tmp/as-start.strace ./asadmin \
start-instance -u admin instancename
```

DAS

```
strace -fv -o /tmp/as-start.strace ./asadmin \
start-domain -u admin domainname
```

Windows Use `DebugView` tool. You can download this tool from `http://www.sysinternals.com/Utilities/DebugView.html`.

d. Collect log files from the following locations, depending on which Application Server component is failing.

- **DAS** – *server-root*`/domains/`*domainname*`/logs`
- **Instance** – *server-root*`/nodeagents/`*nodeagentname*`/`*instancename*`/logs`

- **Node agent** – *server-root*`/nodeagents/`*nodeagentname*`/agent/logs`

e. If the log files do not contain an error message about the problem, use the logging facility from the DAS admin console.

3 Set logging on the failing component to the `FINEST` level.

If you cannot access the Application Server Admin GUI, you can directly edit the `module-log-levels` in the following configuration files:

- UNIX (Solaris and HP-UX) and Linux

  Edit the *server-root*`/domains/`*domainname*`/config/domain.xml` file and set logging for the failing component to `FINEST`.

- Windows

  Edit the *server-root*`\domains\`*domainname*`\config\domain.xml` file and set logging for the failing component to `FINEST`.

▼ To Collect Debug Data for the Load Balancer Plug-in

The locations of the following libraries referenced in this procedure depend on the OS and Web server version you are using:

- `libpassthrough.so` for the Sun Web server on UNIX platforms, `passthrough.dll` on Windows
- `mod_loadbalancer.so` for the Apache Web server on UNIX platforms, `mod_loadbalancer.dll` on Windows
- `sun-passthrough.dll` for Microsoft IIS (Windows only)

1 Check the plug-in version with the following command (UNIX only):

- Web Server 6.0 or Web Server 6.1

```
strings libpassthrough.so | grep BuildId
```

- Apache Web server

```
strings mod_loadbalancer.so | grep BuildId
```

2 Verify the checksum of the plug-in library using one of the following commands:

- `cksum`
- `elfdump –k`
- `sum`
- `md5sum`

3. **Edit and add the relevant configuration files to generate more debug information.**

   - Sun Java Web Server 6.0

     Solaris, HP-UX, Linux: Edit the *server-root*`/web-identifier/config/magnus.conf` file, adding the following line at the end:

     ```
     LogVerbose on
     ```

     Windows: Edit the *server-root*`\web-identifier\config\magnus.conf` file, adding the following line at the end:

     ```
     LogVerbose on
     ```

   - Sun Java Web Server 6.1

     Solaris, HP-UX, Linux: Edit the *server-root*`/web-identifier/config/server.xml` file, setting the `loglevel` property of the `LOG` element to `FINEST`.

     Windows: Edit the *server-root*`\web-identifier\config\server.xml` file, setting the `loglevel` property of the `LOG` element to `FINEST`.

   - Apache

     Solaris, HP-UX, Linux: Edit the *server-root*`/config/httpd.conf` file, setting the `LogLevel` option to `debug`.

     Windows: Edit the *server-root*`\confighttpd.conf` file, setting the `LogLevel` option to `debug`.

4. **Edit the** *server-root*`\web-identifier\config\loadbalancer.xml` **file and set the** `require-monitor-data` **property to** `true`**.**

   ```
   <property name="require-monitor-data" value="true"/>
   ```

   This generates debug information for the Load Balancer plug-in in the Web server error log.

▼ To Collect Debug Data on a Hung or Unresponsive Application Server Process

A process hang is defined as one of the Application Server processes not responding to requests while the `appserv` process is still running. On Solaris systems, you can use the `appserver_8_hang` script to gather debug data in hang situations. See "2.3.1 Running the `appserver_8_hang.sh` Debugging Script" on page 26 for detailed information about this script.

Before You Begin Make sure that you collect all the data over the same time frame in which the problem occurs. See "2.2 Configuring Solaris OS to Generate Core Files" on page 25 if a core file is not generated.

Collect the following information for process hang problems. Run the commands in the order listed below when the problem occurs. Be sure to specify the time when the process hang happened and affected processes, if possible.

1 **Collect the general system information as explained in "To Collect Required Debug Data for Any Application Server Problem" on page 11.**

2 **If your Application Server uses JDK 1.4.2 or higher, verify that the `-XX:+PrintClassHistogram` Application Server JVM option is enabled.**

For more information, see `http://docs.sfbay.sun.com/source/819-0215/jvm.html`.

3 **For the Solaris platform, use the `ptree` command on any of the following processes, depending on the component for which you want to collect data.**

DAS	`ps -ef | grep appservDAS`
Node agent or instance	`ps -ef | grep appserv`

- Sample DAS Output

```
ptree 21293
21293 /export/home/as81ur2-02/lib/appservDAS domain1
 21299 /bin/sh /export/home/as81ur2-02/imq/bin/imqbrokerd -javahom
  21309 /export/home/pablo/as81ur2-02_caixa/jdk/bin/java -server -cp /export/
```

Note – Collect data on the lowest PID process for the DAS, which in this example is 21293, or the highest PID for the instance, which is in this example is 178.

- Sample Instance Output

```
ptree 171
171 ./bin/sh /export/home/as82p1/nodeagents/draco.spain.sun.com/instac
 173 /export/home/as82p1/lib/appservLauncher /export/home/as82
  178 /export/home/as82p1/lib/appserv instacia_draco
```

4 **Run the `netstat` command and save the output.**

Solaris, HP-UX, Linux	`netstat -an | grep` *appserv-port*
Windows	`netstat -an`

5 For Solaris and Application Server 8, run the `appserv_8_hang.sh` script.

The `appserv_8_hang.sh` script captures the debug data for Solaris and Application Server 8. After running this script, run the `pkg_app` script on one of the core files generated by the `appserv_8_hang`.

6 Run the following commands and save the output.

Solaris `ps -aux | server-root`

`vmstat 5 5`

`iostat -xtopuptime`

HP-UX `ps -aux | server-root`

`vmstat 5 5`

`iostat -xtopsar`

Linux ps -aux | *server-root*

`vmstat 5 5`

Windows Obtain the `appserv` or `appservDAS` process PID:

`C:\`*windbg-root*`>tlist.exe`

Obtain the process details of the `appserv` or `appserveDAS` running process PID:

`C:\`*windbg-root*`>tlist.exe` *appserv_PID*

7 Get the swap information.

Solaris `swap -l`

HP-UX `swapinfo`

Linux `free`

Windows Already provided in `C:\report.txt`.

8 *(Solaris only)* **If you are able to isolate the hung process, collect the following debug data for that process.**

Using the PID obtained in Step 3, above, get a series of five of the following commands (one every ten seconds):

`pstack` *appserv-pid*
`pmap -x` *appserv-pid*

Additionally, collect the outputs from the following commands:

```
prstat -L -p appserv-pid
pfiles appserv-pid
pmap appserv-pid
```

9 Get a Java Stack trace from the either `appserv` or `appservDAS` process.

```
ptree 171
171 ./bin/sh /export/home/as82p1/nodeagents/draco.spain.sun.com/instac
 173 /export/home/as82p1/lib/appservLauncher /export/home/as82
  178 /export/home/as82p1/lib/appserv instacia_draco
kill -3 178
```

The `kill -3` dumps a Java stack trace for the `server.log` file.

10 Gather `core` files and the output of the following commands.

If a process hangs, it is helpful to compare several `core` files to review the state of the threads over time. Rename the `core` file, wait for approximately one minute, then rerun the following commands. In this way you can collect a series of `core` files without subsequent files overwriting each other. Do this three times to obtain `three` core files.

Note – For HP-UX, you need the `PHKL_31876` and `PHCO_32173` patches to use the `gcore` command. If you cannot install these patches, use the HP-UX `/opt/langtools/bin/gdb` command from version 3.2 or later, or use the `dumpcore` command.

- **Solaris**

  ```
  gcore -o /tmp/appserver-core appserv-pid
  pstack /tmp/appserver-core
  ```

- **HP-UX**

  ```
  # cd server-root/lib
  gcore -p appserv-pid
  (gdb) attach appserv-pid
  Attaching to process appserv-pid
  No executable file name was specified
  (gdb) dumpcore
  Dumping core to the core file core.appserv-pid
  (gdb) quit
  The program is running. Quit anyway (and detach it)? (y or n) y
  Detaching from program: , process appserv-pid
  ```

- **Linux**

  ```
  # cd server-root/lib
  gdb
  ```

```
(gdb) attach appserv-pid
Attaching to process appserv-pid
No executable file name was specified
(gdb) gcore
Saved corefile core.appserv-pid
(gdb)backtrace
(gdb)quit
```

- **Windows**

  Get the `appserv` process PID:

  ```
  C:\windbg-root>tlist.exe
  ```

  Generate a crash dump on the `appserv` running process PID:

  ```
  C:\windbg-root>adplus.vbs -hang -p appserv-pid -o C:\crashdump_dir
  ```

  Note – For Windows, provide the complete generated folder under `C:\crashdump_dir`.

11 For Solaris, archive the result of the `pkg_app` script (at least one `core` file is required).

```
./pkg_app.ksh application_pid corefile
```

Note – Make sure that the appropriate limitations are set by using the `ulimit` command, and that the user ID is not `nobody`. Also check the `coreadm` command for additional control. See "2.2 Configuring Solaris OS to Generate Core Files" on page 25 if a core file is not generated.

▼ To Collect Debug Data on a Application Server Crashed Process

Use this task to collect data when a Application Server process has stopped (crashed) unexpectedly. Run all the commands on the actual machine where the `core` file(s) were generated.

1 Collect the general system information as explained in "To Collect Required Debug Data for Any Application Server Problem" on page 11.

2 Collect the swap information

Solaris	`swap -l`
HP-UX	`swapinfo`
Linux	`free`

Windows Already provided in `C:\report.txt`.

3 Collect the system logs.

Solaris, Linux	`/var/adm/messages/var/log/syslog`
HP-UX	`/var/adm/syslog/syslog.log`
Windows	Go to *Start->Settings->Control Panel->Event Viewer->Select Log*, and then click *Action->Save Log File As* and enter a name for the resulting file.

4 Collect the `core` files (called "Crash Dumps" in Windows).

- **Solaris**

  See "2.2 Configuring Solaris OS to Generate Core Files" on page 25 if a `core` file was not generated.

- **Linux**

  Core dumps are turned off by default in the `/etc/profile` file. You can make user-specific changes by editing your `~/.bash_profile` file. Look for the following line:

  ```
  ulimit -S -c 0 > /dev/null 2>&1
  ```

  You can either comment out the entire line to set no limit on the size of the `core` files or set your own maximum size.

- **Windows**

  Generate a crash dump during a crash of Application Server by using the following commands:

  Get the `appserv` process PID:

  ```
  C:\windbg-root>tlist.exe
  ```

  Generate a crash dump when the `appserv` process crashes by executing the following commands:

  ```
  C:\windbg-root>adplus.vbs -crash -FullOnFirst -p appserv-pid -o C:\crashdump_dir
  ```

  The `adplus.vbs` command monitors *appserv-pid* until it crashes and generates the `dmp` file. Provide the complete generated folder under `C:\crashdump_dir`.

Note – If you have not installed the Debugging Tools for Windows, you can use the `drwtsn32 -i` command to select Dr. Watson as the default debugger. Use the `drwtsn32` command, check all options, and choose the path for crash dumps. Then provide the dump and the `drwtsn32.log` files.

---

5 **(Solaris only) For each core file, provide the output of the following commands.**

```
cd server-root/bin/https/bin
file corefile
pstack corefile
pmap corefile
pflags corefile
```

6 **(Solaris only) Archive the result of the `pkg_app` script (one core file is sufficient).**

```
./pkg_app.ksh application-pid corefile
```

Note – The Sun Support Center must have the output from the `pkg_app` script to properly analyze the `core` file(s). For more information on how to run the `pkg_app` script, see "2.3.2 Running the `pkg_app` Script" on page 28.

All these commands must be executed on the machine on which the `core` file(s) are generated.

2.2 Configuring Solaris OS to Generate Core Files

Core files are generated when a process or application terminates abnormally. Core files are managed with the `coreadm` command. This section describes how to use the `coreadm` command to configure a system so that all process core files are placed in a single system directory. This enables you to track problems by examining the core files in a specific directory whenever a Solaris OS process or daemon terminates abnormally.

Before configuring your system for core files, make sure the `/var` file system has sufficient space. Once you configure Solaris OS to generate core files, a core file is written to the `/var/cores` directory every time a process crashes.

▼ To Configure Solaris OS to Generate Core Files

1 **Run the following commands as `root`.**

```
mkdir -p /var/cores
coreadm -g /var/cores/%f.%n.%p.core -e global -e process -e \
global-setid -e proc-setid -e log
```

2 **View the core configuration.**

```
# coreadm
           global core file pattern:
                init core file pattern: %f.%n.%p.core
                  global core dumps: enabled
```

```
          per-process core dumps: enabled
          global setid core dumps: enabled
 per-process setid core dumps: enabled
       global core dump logging: enabled
```

See the `coreadm` man page for further information.

3 Set the size of the core dumps to unlimited.

```
# ulimit -c unlimited
# ulimit -a

        coredump(blocks) unlimited
```

See the `ulimit` man page for further information.

4 Verify core file creation.

```
# cd /var/cores
# sleep 100000 &
[1] PID
# kill -8 PID
# ls
```

2.3 Running the Application Server Debugging Scripts

This section describes the `appserver_8_hang.sh` and `pkg_app` scripts.

- "2.3.1 Running the `appserver_8_hang.sh` Debugging Script" on page 26
- "2.3.2 Running the `pkg_app` Script" on page 28

2.3.1 Running the `appserver_8_hang.sh` Debugging Script

The goal of the `appserver_8_hang.sh` script is to automate the collection of debug data if the Application Server gets into an unresponsive state when running on a Solaris system.

▼ To Run the `appserver_8_hang.sh` Script

All data generated by the script is compiled under a single `tar.gz` file named `APPSERVD_DATA_`*ddmmyyhhmmss*`.tar.gz`, where *ddmmyyhhmmss* is the date and timestamp.

In addition to general debug data, this script gathers data three times on the specific instance (`prstat -L -p`, `pflags`, `pstack`, `pmap`, `pldd`, `pfiles`), separated by the time specified in the variable `INTERVAL` (see below).

- The script also enables you to optionally get:

- `gcore`
- Java thread dumps by sending a `SIGQUIT` signal to the process (note that this output is sent to the Application Server `server.log` file)
- Truss
- HTTP 1.0 and 1.1 requests
- Java 5–specific information (`jps`, `jinfo`, `jmap`, `jstack`)

- Known limitations:
  - Only data from DAS instances or specific instances can be gathered.
  - If there are two or more Application Server domains with the same name running on the same machine, there is no guarantee that the data gathered will correspond to the intended domain.

1 The following variables must be defined before running the script:

`DAS_OR_INSTANCE`	Either domain name, if gathering DAS instance information, or instance name, if gathering non-DAS instance information
`DAT_DIR`	Output directory where temporary and final data will be stored
`HOSTNAME`	Host name/IP of the interface on which the HTTP requests will be made when using the `INTERVAL` variable (see below)
`DAS_OR_INSTANCE_HTTP_PORT`	HTTP port on which the HTTP requests will be made (see below)

2 Optionally, you can set the following at run time:

`INTERVAL`	Determines the interval between loops (see above); defaults to `10` seconds
`JAVA_5`	If the app server is running Java 5, choose "`yes`" to have the script run additional Java 5–specific commands (`jps`, `jinfo`, `jmap`, `jstack`)
`JAVA_HOME`	Required if `JAVA_5=yes`
`GCORE`	Determines whether a `gcore` of the Application Server process will be taken (`GCORE="yes"`) or not (any other value); running the `pkg_app.sh` script on the generated `core` file is required if this variable is enabled
`JAVA_THREADS`	Determines whether `kill -3` will be run on the Application Server process (to obtain Java thread dump)
`TRUSS`	Determines whether the Application Server process will be trussed for `TRUSS_DURATION`; defaults to `30` seconds

`HTTP_10_REQ`	Determines whether an HTTP 1.0 request will be made on the Application Server's HTTP interface (`DAS_OR_INSTANCE_HTTP_PORT`)
`HTTP_11_REQ`	Determines whether an HTTP 1.1 request will be made on the Application Server's HTTP interface (`DAS_OR_INSTANCE_HTTP_PORT`)
`HTTP_URI`	The URI used when running the HTTP requests (see above)

2.3.2 Running the `pkg_app` Script

This script packages an executable and all of its shared libraries into one compressed tar file, and named corresponding to the PID of the application and optionally the name of the `core` file to be opened. The files are stripped of their directory paths and are stored under a relative directory named `app/` with their name only, allowing them to be unpacked in one directory.

▼ To Run the `pkg_app` Script

On Solaris 9 OS or later, if the `core` file is specified, the list of files is derived from the `core` file rather than from the process image. You still must provide the PID of the running application to assist in path resolution.

Two scripts are created to facilitate opening the `core` file when the tar file is unpacked:

- `opencore` – The script to be executed once unpacked; it sets the name of the core file and the linker path to use the `app/` subdirectory and then invokes `dbx` with the `dbxrc` file as the argument.
- `dbxrc` – Contains the `dbx` initialization commands to open the core file.

Note – The `pkg_app` script asks for your Sun Case Number to name the `tar.gz` file.

---

1. **Copy the script to a temporary directory on the system where Application Server is installed.**
2. **Become superuser.**
3. **Execute the `pkg_app` script in one of the following three ways:**
   - `./pkg_app` *application-pid corefile*
   - `./pkg_app` *application-pid*
     (The *pkg_app* scripts prompts for the *corefile* name.)
   - `./pkg_app` *core file*