Sun Gathering Debug Data for Sun Java System Messaging Server

Sun Java™ Communications Suite Technical Note

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

Part No: 819–5355–12
August 2007

070809@18135

◆ ◆ ◆ CHAPTER 1

Sun Gathering Debug Data for Sun Java System Messaging Server

This technical note describes how to use Sun™ Gathering Debug Data (Sun GDD or GDD) to collect data that the Sun Support Center requires in order to debug problems with a Sun Java™ System Messaging 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 Messaging 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 Messaging Server with the Sun Support Center.

This technical note contains the following sections:

- "1.1 Technical Note Revision History" on page 4
- "1.2 About This Technical Note" on page 4
- "1.3 Overview of Collecting Debug Data for Messaging Server" on page 6
- "1.4 Creating a Service Request with the Sun Support Center" on page 6
- "1.5 What Messaging Server Debug Data Should You Collect?" on page 7
- "1.6 Configuring Solaris OS to Generate Core Files" on page 24
- "1.7 Running the Messaging Server Debugging Scripts" on page 26
- "1.8 Reporting Problems" on page 28
- "1.9 Accessing Sun Resources Online" on page 28

1.1 Technical Note Revision History

Version	Date	Description of Changes
12	August 2007	The `dbhang` command has been updated. See "1.7 Running the Messaging Server Debugging Scripts" on page 26 for details.
11	January 2007	Updated "To Configure Solaris OS to Generate Core Files" on page 24.
10	December 2006	Initial release of this technical note.

1.2 About This Technical Note

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

- Sun Java System Messaging Server 6.3
- Sun Java System Messaging Server 6 2005Q4
- Sun Java System Messaging Server 6 2005Q1
- Sun Java System Messaging Server 6 2004Q2
- Sun Java System Messaging Server 6 2003Q4
- Sun ONE Messaging Server 6
- iPlanet™ Messaging Server 5

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 nor techniques or tools to analyze debug data. It provides some troubleshooting, but you should not use this guide as an approach to troubleshooting Messaging Sever problems.

If your problem does not conveniently fit into any of the specific categories, supply the general information described in "1.5 What Messaging Server Debug Data Should You Collect?" on page 7 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 Messaging Server Debug Data

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

- Make sure you have `superuser` privileges.
- For the Solaris OS platform, obtain the `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.

- *channel*: The channel name where the messages are queued.
- *identifier*: The Messaging Server instance name used during installation. The installation program automatically added the prefix `msg-` to the name you specified. For example, if you named the identifier `tango`, the installation program created `msg-tango`. This only applies to iPlanet Messaging Server 5.
- *messaging-pid*: Process ID of a Messaging Server daemon.
- *messaging-service-port*: Port number used by a Messaging Service (IMAP, POP, HTTP, and so forth).
- *msg-instance*: The directory on the Messaging Server machine dedicated to holding configuration, maintenance, and information files for a specific instance. This directory is located under *server-root*. This only applies to iPlanet Messaging Server 5. In Sun Java System Messaging Server 6, it is the `config` directory.

- *server-root*: The directory on the Messaging Server machine dedicated to holding the server program, configuration, maintenance, and information files. The default location for the Solaris OS version of Sun Java System Messaging Server 6 is `/opt/SUNWmsgsr/`. See "To Obtain the server-root Variable" on page 6 for more information on determining the value of *server-root*.
- *windbg-root*: The directory on the Windows Messaging Server machine dedicated to holding the Win Debugger program, and configuration, maintenance, and information files.

▼ To Obtain the server-root Variable

- **Use the following to obtain the value of the *server-root* variable.**
  - Sun Java System Messaging Server (Messaging Server 6):

    Solaris `pkgparam -v SUNWmsgco`

    Linux `rpm -q --qf '%{INSTALLPREFIX}' sun-messsaging-server`
  - iPlanet Messaging Server (Messaging Server 5):

    Solaris Look in the `/etc/msgregistry.inf` file.

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

1.3 Overview of Collecting Debug Data for Messaging Server

Collecting debug data for a Messaging 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` 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 Sun Support Center, 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 What Messaging 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 Messaging Server Problem" on page 7
- "To Collect Debug Data on Messaging Server Installation Problems" on page 9
- "To Collect Debug Data on a Hung or Unresponsive Messaging Server Process" on page 11
- "To Collect Debug Data on a Messaging Server Crashed Process" on page 17
- "To Collect Debug Data on a Messaging Server Routing Problem" on page 19
- "To Collect Debug Data on a Messaging Server MTA Queue Problem" on page 21
- "To Collect Debug Data on a Messaging Server Webmail Problem" on page 24

▼ To Collect Required Debug Data for Any Messaging 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 operating system.**

   Solaris OS `uname -a`

   HP-UX `uname -r`

   Linux `more /etc/redhat-release`

   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 Messaging Server.**

   Be sure to send the entire screen output of the `imsimta version` command.

   - Sun Java System Messaging Server (Messaging Server 6):

     UNIX and Linux
     ```
     cd server-root/sbin
     ./imsimta version
     ```

     Windows
     ```
     cd server-root\sbin
     imsimta.exe version
     ```

   - iPlanet Messaging Server (Messaging Server 5):

     UNIX and Linux
     ```
     cd server-root/msg-identifier
     ./imsimta version
     ```

     Windows
     ```
     cd server-root\msg-identifier
     imsimta.exe version
     ```

6. **Create a tar file of the Messaging Server configuration directory.**

   - Sun Java System Messaging Server (Messaging Server 6):

UNIX and Linux	`cd` *server-root*`/sbin`
	`./configutil`
	Create a tar file of the *server-root*`/config` directory.

Windows	`cd` *server-root*`\config`
	`configutil.exe`
	Create a tar file of the *server-root*`\config` directory.

- iPlanet Messaging Server (Messaging Server 5):

UNIX and Linux	`cd` *server-root*`/msg-`*identifier* `./configutil` Create a tar file of the *server-root*`/msg-`*identifier*`/imta/config` directory.
Windows	`cd` *server-root*`\msg-`*identifier* `configutil.exe` Create a tar file of the *server-root*`\msg-`*identifier*`\imta\config` directory.

Note – 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 happening during the error occurrence and shortly before. Thus for relatively short log files (for example, MTA channel debug log files), send the entire log file, whereas for long-running hence large log files, an extract might be more appropriate, though 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.

However, when questions arise about message structure or content, or about notification or bounced messages, then send an entire sample message, including all the outermost header (not just an excerpt of the message).

▼ To Collect Debug Data on Messaging Server Installation Problems

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

1 Consult the following troubleshooting information:

- Messaging Server 6 2005Q4:
  Chapter 9, "Troubleshooting," in *Sun Java Enterprise System 2005Q4 Installation Guide for UNIX*
- Messaging Server 6 2005Q1:
  Chapter 13, "Troubleshooting," in *Sun Java Enterprise System 2005Q1 Installation Guide*

- Messaging Server 6 2004Q2:
  Chapter 11, "Troubleshooting," in *Sun Java Enterprise System 2004Q2 Installation Guide*
- Messaging Server 6 2003Q4:
  Chapter 9, "Troubleshooting Installation Problems," in *Sun Java Enterprise System 2003Q4 Installation Guide*

If the problem persists after using this troubleshooting information, then continue with this procedure to collect the necessary data for the Sun Support Center.

2 Collect the general system information as explained in "To Collect Required Debug Data for Any Messaging Server Problem" on page 7.

3 Specify if this is a first-time installation or a Hot Fix installation on a pre-existing Messaging Server instance.

4 Get the installation logs.

- Sun Java System Messaging Server (Messaging Server 6):

  Messaging Server 6 log files mostly reside in the *server-root*`/log` directory. However, the initial configuration log files reside in the *server-root*`/install` directory, which also contains information on the initial configuration.

Solaris OS	`/var/sadm/install/logs` The log file names start with `Java_Enterprise_System*_install.B`*datetime*, where *date* and *time* correspond to the failing installing (for example, `B12161532`).
HP-UX and Linux	`/var/opt/sun/install/logs` The log file names start with `Java_Enterprise_System*_install.B`*datetime*, where *date* and *time* correspond to the failing installing (for example, `B12161532`).
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.

- iPlanet Messaging Server (Messaging Server 5): Rerun the installation with the following command and save the resultant file.

Solaris OS	`truss -ealf -rall -wall -vall -o /tmp/install-messaging.truss ./setup`
HP-UX	`tusc -v -feaIT -rall -wall -o /tmp/install-messaging.tusc ./setup`
Linux	`strace -fv -o /tmp/install-messaging.strace ./setup`

Windows	Use Debug View: `http://www.sysinternals.com/Utilities/DebugView.html`

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

A process hang is defined as one of the Messaging Server processes not responding to requests anymore while the process is still running locally. Messaging Server's seven specific processes are:

- `imapd`—Provides access to IMAP services.
- `popd`—Provides access to POP services.
- `mshttpd`—Provides access to Webmail services.
- `dispatcher`—Permits multiple multithreaded server processes to share responsibility for SMTP connection services.
- `enpd`—Collects and dispatches events that occur to properties of resources (inboxes or calendars).
- `job_controller`—Ensures delivery of messages.
- `stored`—Performs a variety of important tasks such as deadlock and transaction operations of the message database, enforcing aging policies, and expunging and erasing messages stored on disk.

Additionally, Messaging Server uses these three processes:

- `tcp_smtp_server`—Handles SMTP sessions.
- `tcp_lmtp_server`—Handles LMTP sessions.
- `tcp_lmtpn_server`—Handles LMTP native sessions.

The system can be running more than one of each of these processes (especially `tcp_smtp_server`).

Other processes from the `job_controller.cnf` configuration file might be running (`autoreply`, `ims_master`, `l_master`, `conversion`, `reprocess`, and so on), as well as `reconstruct` and `imexpire` processes.

Before You Begin Make sure that you collect all the data over the same time frame in which the problem occurs. See "1.6 Configuring Solaris OS to Generate Core Files" on page 24 if a core file is not generated.

Collect the following information for process hang problems. Run the commands in order 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 Messaging Server Problem" on page 7.

2 Run the `netstat` command and save the output.

UNIX and Linux
```
netstat -an | grep messaging-service-port
```

Windows
```
netstat -an
```

3 Run the following commands and save the output.

Solaris OS
```
ps -ef | grep server-root
vmstat 5 5
iostat -x
top
uptime
```

HP-UX
```
ps -aux | grep server-root
vmstat 5 5
iostat -x
top
sar
```

Linux
```
ps -aux | grep server-root
vmstat 5 5
top
uptime
sar
```

Windows

Obtain the `MESSAGING` process PID: `C:\`*windbg-root*`>tlist.exe`

Obtain process details of the `MESSAGING` running process PID:
`C:\`*windbg-root*`>tlist.exe` *messaging-pid*

4 Get the swap information.

Solaris OS
```
swap -l
```

HP-UX
```
swapinfo
```

Linux
```
free
```

Windows — Already provided in `C:\report.txt` as described in "To Collect Required Debug Data for Any Messaging Server Problem" on page 7.

5 Get the system logs.

Solaris OS and Linux	`/var/adm/messages` `/var/log/syslog`
HP-UX	`/var/adm/syslog/syslog.log`
Windows	Event log files: Start-> Settings-> Control Panel —> Event Viewer-> Select Log Then click Action-> Save log file as

---

Note – For UNIX systems, depending on your site's configuration of the `SNDOPR_PRIORITY option.dat` option and your syslog configuration (`syslog.conf`), the MTA might be sending automatically generated syslog notices to a non-default location. Also, the `LOG_SNDOPR option.dat` option controls whether additional potential syslog notices are generated by the MTA message logging facility.

---

6 For UNIX and Linux systems, get the contents of the `/etc/nsswitch.conf` and `/etc/resolv.conf` files.

7 Get the most current log files for the hanging process, if known. Otherwise, get the log files for all processes.

You should also set `LOG_SNDOPR` in the `option.dat` file and check for syslog notices if some expected MTA log file doesn't seem to have been generated. When you set the `LOG_SNDOPR` option, then the MTA generates a syslog notice if it cannot write to its regular log files.

- Sun Java System Messaging Server (Messaging Server 6):

UNIX and Linux	*server-root*`/log/*`
Windows	*server-root*`\data\log\default\default` *server-root*`\data\log\http\http` *server-root*`\data\log\imap\imap` *server-root*`\data\log\pop\pop` *server-root*`\data\log\imta\*`

- iPlanet Messaging Server (Messaging Server 5):

UNIX and Linux	*server-root*`/msg-`*identifier*`/log/default/default` *server-root*`/msg-`*identifier*`/log/http/http` *server-root*`/msg-`*identifier*`/log/imap/imap` *server-root*`/msg-`*identifier*`/log/pop/pop` *server-root*`/msg-`*identifier*`/log/imta/*`
Windows	*server-root*`\msg-`*identifier*`\log\default\default` *server-root*`\msg-`*identifier*`\log\http\http` *server-root*`\msg-`*identifier*`\log\imap\imap`

server-root\msg-*identifier*\log\pop\pop
server-root\msg-*identifier*\log\imta*

8 (Solaris OS only) If you are able to isolate the hanging process, get the following debug data for that process. Otherwise, get the following data for each of the Messaging Server processes.

Using the PID obtained in Step 3, get a series of five of the following commands (one every 10 seconds):

```
pstack messaging-pid
pmap -x messaging-pid
```

9 Look for any core file that could have been dumped by one of the Messaging Server processes. If you find one, see "To Collect Debug Data on a Messaging Server Crashed Process" on page 17.

10 If any of the `ims_master`, `imapd`, `mshttpd`, `popd`, `mboxutil`, or `reconstruct` processes is hung, as the `mailsrv` user, get the outputs of the `db_stat` command as follows.

- UNIX and Linux:

```
cd msg-instance/data/store/mboxlist
server-root/lib/db_stat -Co -h msg-instance/data/store/mboxlist/ -N
server-root/lib/db_stat -t
```

If you have set the `dbtmpdir configutil` variable, use that location instead of *msg-instance*`/store/mboxlist/`.

Note – For Solaris OS systems only, the script `dbhang` captures all the following debug data for you. Edit the top of the script to match your system's configuration. Specifically, edit the `SRVROOT`, `INST`, `MAILUSER`, and `DBTMPDIR` parameters. Then run the script and collect the data. See "1.7 Running the Messaging Server Debugging Scripts" on page 26 for more information.

- Windows:

```
cd msg-instance\data\store\mboxlist
server-root\lib\db_stat -Co -h msg-instance\data\store\mboxlist\ -N
server-root\lib\db_stat -t
```

11 Get the output of the following command.

Solaris OS	`truss -ealf -rall -wall -vall -o /tmp/truss.out -p` *messaging-pid*
HP-UX	`tusc -v -fealT -rall -wall -o /tmp/tusc.out -p` *messaging-pid*
Linux	`strace -fv -o /tmp/strace.out -p` *messaging-pid*

Windows Use DebugView: `http://www.sysinternals.com/Utilities/DebugView.html`

Note – Wait one minute after launching the appropriate command (`truss`, `strace`, `tusc`, or DebugView) then stop it by pressing **`Control-C`** in the terminal where you launched the command.

---

12 Get core files and the output of the following commands.

In a process hang situation, it is helpful to compare several core files to review the state of the threads over time. To not overwrite a core file, copy that core file to a new name, wait approximately one minute then rerun the following commands. Do this three times to obtain three core files.

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

Solaris OS

```
cd server-root/bin/slapd/server
gcore -o /tmp/messaging_process-core messaging-pid
pstack /tmp/messaging_process-core
```

HP-UX

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

Linux

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

(gdb)backtrace
(gdb)quit
```

Windows Get the MESSAGING process PID:

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

Generate a crash dump on the MESSAGING running process PID:

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

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

---

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

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

The Sun Support Center must have the output from the `pkg_app` script to properly analyze the core file(s).

Note – Make sure the appropriate limitations are set by using the `ulimit` command, and that the user is not `nobody`. Also check the `coreadm` command for additional control. See "1.6 Configuring Solaris OS to Generate Core Files" on page 24 if a core file is not generated.

---

14 When you have collected all debug data, perform the following steps to restore the service.

Messaging Server processes usually hang because of an orphan lock left in one of the databases. Stopping the server (especially the `stored` process), and cleaning the temporary shared db files helps to resolve the problem.

a. Stop Messaging Server.

```
cd server-root/sbin
./stop-msg
```

b. Make sure that all Messaging Server processes stopped.

Wait one minute, then kill any remaining processes, except `tcp_smtp_server` processes (which do not use databases).

c. Restart Messaging Server.

```
./start-msg
```

d. After restarting the services, check the logs for any unexpected behavior.

▼ To Collect Debug Data on a Messaging Server Crashed Process

Use this task to collect data when a Messaging 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 Messaging Server Problem" on page 7.

2 Note whether you can you restart Messaging Server.

3 Get the output of the following commands.

Solaris OS
```
ps -ef | grep server-root
vmstat 5 5
iostat -x
top
uptime
```

HP-UX
```
ps -aux | grep server-root
vmstat 5 5
iostat -x
top
sar
```

Linux
```
ps -aux | grep server-root
vmstat 5 5
top
uptime
sar
```

Windows

Obtain the `MESSAGING` process PID: `C:\`*windbg-root*`>tlist.exe`

Obtain process details of the `MESSAGING` running process PID: `C:\`*windbg-root*`>tlist.exe` *messaging-pid*

4 Get the swap information.

Solaris OS
```
swap -l
```

HP-UX
```
swapinfo
```

Linux
```
free
```

Windows

Already provided in `C:\report.txt` as described in "To Collect Required Debug Data for Any Messaging Server Problem" on page 7.

5 Get the system logs.

Solaris OS and Linux	`/var/adm/messages` `/var/log/syslog`
HP-UX	`/var/adm/syslog/syslog.log`
Windows	Event log files: Start-> Settings-> Control Panel —> Event Viewer-> Select Log Then click Action-> Save log file as

---

Note – For UNIX systems, depending on your site's configuration of the `SNDOPR_PRIORITY option.dat` option and your syslog configuration (`syslog.conf`), the MTA might be sending automatically generated syslog notices to a pre—determined location. Also, the `LOG_SNDOPR option.dat` option controls whether additional potential syslog notices are generated by the MTA message logging facility.

---

6 Get core files (called "Crash Dumps" by Windows).

Solaris OS — See "1.6 Configuring Solaris OS to Generate Core Files" on page 24 if a core file was not generated.

Linux — Core dumps are turned off by default in the `/etc/profile` file. You can make per user 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 Messaging Server by using the following commands:

Get the MESSAGING process PID : `C:\`*windbg-root*`>tlist.exe`
Generate a crash dump when the MESSAGING process crashes:
`C:\`*windbg-root*`>adplus.vbs -crash -FullOnFirst -p` *messaging-pid* `-o C:\crashdump_dir`

The `adplus.vbs` command watches *messaging-pid* until it crashes and will generate the `dmp` file. Provide the complete generated folder under `C:\`*crashdump_dir*.

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

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

```
file corefile
pstack corefile
pmap corefile
pflags corefile
```

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

```
./pkg_app.ksh Pid-of-application corefile
```

Note – The Sun Support Center must have the output from the `pkg_app` script to properly analyze the core file(s).

▼ To Collect Debug Data on a Messaging Server Routing Problem

Use this task to collect data when Messaging Server is experiencing a routing problem.

A Messaging Server routing problem is defined as the inability of the system to correctly route a message. For example, a message might be ending up in the wrong Message Store, might be sent to the wrong channel, might be delivered to the wrong user, and so on.

1 Collect the general system information as explained in "To Collect Required Debug Data for Any Messaging Server Problem" on page 7.

2 Provide a detailed explanation of what do you want to obtain.

3 Get the output of the following command.

- Sun Java System Messaging Server (Messaging Server 6):

  UNIX and Linux
  ```
  cd server-root/sbin
  ./imsimta test -rewrite -debug=level=5 mailaddress
  ```

  Windows
  ```
  cd server-root\sbin
  imsimta.exe test -rewrite -debug=level=5 mailaddress
  ```

- iPlanet Messaging Server (Messaging Server 5):

UNIX and Linux	`cd` *server-root*`/msg-`*identifier* `./imsimta test -rewrite -debug=level=5` *mailaddress*
Windows	`cd` *server-root*`\msg-`*identifier* `imsimta.exe test -rewrite -debug=level=5` *mailaddress*

Note – When the problem is about Sieve filters, add the option `-filter` to the above command.

The `-noimage` qualifier to the `imsimta test -rewrite -debug` command enables you to test changes made to the configuration file prior to recompiling the new MTA configuration.

If possible, use the `-source_channel=`*source_channel* option to specify the incoming channel. This is sometimes necessary for testing the interactions with the mapping tables and the antirelay rules. By default, Internet mail arrives on the `tcp_local` channel whereas internal mail arrives on the `tcp_intranet` channel. If the connection is authenticated, use `tcp_auth`.

The command would then look like:
`# ./imsimta test -rewrite -debug -source_channel=tcp_intranet` *email-address*

4 Get the LDIF entry of an impacted user.

UNIX and Linux	*dir-root*`/shared/bin/ldapsearch -h` *hostname* `-p` *port* `-D "cn=Directory Manager" -w` *password* `-b "`*basedn*`" "(objectclass=*)" uid=user > /tmp/user.ldif`
Windows	*dir-root*`\shared\bin\ldapsearch.exe -h` *hostname* `-p` *port* `-D "cn=Directory Manager" -w` *password* `-b "`*basedn*`" "(objectclass=*)" uid=user > C:\user.ldif`

where:

dir-root	The directory on the Directory Server machine dedicated to holding the server program, and configuration, maintenance, and information files. The default location for UNIX and Linux versions of Messaging Server is `/var/opt/mps/serverroot/`.
hostname	Name of the host running Directory Server. The default value is `localhost`. You can omit `-h` *hostname* if the Directory Server is running locally.
port	Port number on which Directory Server is listening. The default is 389. You can omit *port* if the Directory Server is running on port 389.
basedn	The base dn for the search. Use *basedn* as the starting point for the search.

5 Get the LDIF entry of the domain where the impacted user resides.

UNIX and Linux	*dir-root*`/shared/bin/ldapsearch -h` *hostname* `-p` *port* `-D "cn=Directory Manager" -w` *password* `-s base -b "`*baseDN*`" "(objectclass=*)" > /tmp/domain.ldif`
Windows	*dir-root*`\shared\bin\ldapsearch.exe -h` *hostname* `-p` *port* `-D "cn=Directory Manager" -w` *password* `-s base -b "`*baseDN*`" "(objectclass=*)" > C:\domain.ldif`

where:

dir-root The directory on the Directory Server machine dedicated to holding the server program, and configuration, maintenance, and information files. The default location for UNIX and Linux versions of Messaging Server is `/var/opt/mps/serverroot/`.

hostname Name of the host running Directory Server. You can omit `-h` *hostname* if the Directory Server is running locally.

port Port number on which Directory Server is listening. The default is 389. You can omit *port* if the Directory Server is running on port 389.

▼ To Collect Debug Data on a Messaging Server MTA Queue Problem

Use this task to collect data when Messaging Server is experiencing a queue problem, for example, when a specific queue is growing and messages are not being dequeued.

1 **Collect the general system information as explained in "To Collect Required Debug Data for Any Messaging Server Problem" on page 7.**

2 **Is the "growing" queue full of `ZZ*.00` message files or full of `Z*.00` message files? How many `ZZ*.00` message files, and how many `Z*.00` message files are there? Or are there `.HELD` files?**

   a. **If the channel has lots of `ZZ*.00` message files and relatively few `Z*.00` message files (where "relatively" depends heavily on the specific channel and site usage), make sure that the Job Controller is running. For example:**

      ```
      ps -ef | grep job_controller
      ```

   b. **If a channel has lots of `Z*.00` message files and not very many `ZZ*.00` message files, then typically the MTA itself does not have any "problem," but rather than there is a problem with a separate destination host or a network problem.**

      In this case, look at the delivery history of the `Z*.00` messages. You need the message files themselves, or better yet, the output of the `imsimta qm history` command. Examine the

`imsimta qm history` output for old `mail.log*` records for those message files. They indicate what sort of SMTP or other error is occurring (and when) for these "old" message files.

For more information on the `imsimta qm` command, see the following:

- Messaging Server 6 2005Q4:
  "imsimta qm" in *Sun Java System Messaging Server 6 2005Q4 Administration Reference*
- Messaging Server 6 2005Q1:
  "imsimta qm" in Chapter 2, "Message Transfer Agent Command-line Utilities," in *Sun Java System Messaging Server 6 2005Q1 Administration Reference*
- Messaging Server 6 2004Q2:
  "imsimta qm" in Chapter 2, "Message Transfer Agent Command-line Utilities," in *Sun Java System Messaging Server 6 2004Q2 Administration Guide*

3 Look at the "Messages are Not Dequeued" section in the *Messaging Server Administration Guide*.

- Messaging Server 6 2005Q4:
  Chapter 22, "Troubleshooting the MTA," in *Sun Java System Messaging Server 6 2005Q4 Administration Guide*
- Messaging Server 6 2005Q1:
  Chapter 13, "Troubleshooting," in *Sun Java System Messaging Server 6 2005Q1 Administration Guide*
- Messaging Server 6 2004Q2:
  Chapter 11, "Troubleshooting," in *Sun Java System Messaging Server 6 2004Q2 Administration Guide*

4 Run the following `imsimta` command to see if messages will now get delivered.

If the messages do now get delivered, then whatever problem was preventing message delivery was probably transient (for example, a network or DNS problem) and is now resolved. Or else the "problem" is simply that you do not have the channel/Job Controller configured for enough simultaneous delivery jobs for that channel to keep up with the current load.

- Sun Java System Messaging Server (Messaging Server 6):

  UNIX and Linux
  ```
  cd server-root/sbin
  ./imsimta run channel
  ```

  Windows
  ```
  cd server-root\sbin
  imsimta.exe run channel
  ```

- iPlanet Messaging Server (Messaging Server 5):

  UNIX and Linux
  ```
  cd server-root/msg-identifier
  ./imsimta run channel
  ```

Note – The `imsimta run` command does not provide much helpful information for `Z*.00` message files.

5 Get the output of the following commands.

- Sun Java System Messaging Server (Messaging Server 6):

  UNIX and Linux

  ```
  cd server-root/sbin
  ./imsimta qm counters show
  ./imsimta qm summ
  ./imsimta qtop -database -domain_to
  ./imsimta qm messages channel
  ```

  Windows

  ```
  cd server-root\sbin
  imsimta.exe qm counters show
  imsimta.exe qm summ
  imsimta.exe qtop -database -domain_to
  imsimta.exe qm messages channel
  ```

- iPlanet Messaging Server (Messaging Server 5):

  UNIX and Linux

  ```
  cd server-root/msg-identifier
  ./imsimta qm counters show
  ./imsimta qtop -database -domain_to
  ```

  Windows

  ```
  cd server-root\msg-identifier
  imsimta.exe qm counters show
  imsimta.exe qm summ
  imsimta.exe qtop -database -domain_to
  ```

6 Get the current log files.

- Sun Java System Messaging Server (Messaging Server 6):

  UNIX and Linux *server-root*`/log/*`

  Windows *server-root*`\data\log\imta\*`

- iPlanet Messaging Server (Messaging Server 5):

  UNIX and Linux *server-root*`/msg-`*identifier*`/log/imta/*`

  Windows *server-root*`\msg-`*identifier*`\log\imta\*`

▼ To Collect Debug Data on a Messaging Server Webmail Problem

Use this task to collect data for a Webmail problem. The most common problems are related to incorrect translation of fields when using a localized Messaging Server interface.

1 **Collect the general system information as explained in "To Collect Required Debug Data for Any Messaging Server Problem" on page 7.**

2 **Take a snapshot of the problematic screen(s).**

3 **Note the step-by-step procedure to reproduce the problem with a test case.**

Note – The Sun Support Center does not support Webmail customizations. Contact your sales representative for those problems.

1.6 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 means it is easier 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 that the `/var` file system has sufficient space. Once you configure Solaris OS to generate core files, from now on all processes that crash will write a core file to the `/var/cores` directory.

▼ 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.%t.core -e global -e global-setid -e log -d process -d proc-setid
```

In this command:

`-g` Specifies the global core file name pattern. Unless a per-process pattern or setting overrides it, core files are stored in the specified directory with a name such as *program*`.`*node*`.`*pid*`.`*time*`.core`, for example: `mytest.myhost.1234.1102010309.core`.

`-e` Specifies options to enable. The preceding command enables:

- Use of the global (that is, system-wide) core file name pattern (and thereby location)
- Capability of `setuid` programs to also dump core as per the same pattern
- Generation of a syslog message by any attempt to dump core (successful or not)

`-d` Specifies options to disable. The preceding command disables:

- Core dumps per the per-process core file pattern
- Per-process core dumps of `setuid` programs

The preceding command stores all core dumps in a central location with names identifying what process dumped core and when. These changes only impact processes started after you run the `coreadm` command. Use the `coreadm -u` command after the preceding command to apply the settings to all existing processes.

2 Display the core configuration.

```
# coreadmglobal core file pattern: /var/cores/%f.%n.%p.%t.core
      init core file pattern: core
           global core dumps: enabled
      per-process core dumps: disabled
     global setid core dumps: enabled
per-process setid core dumps: disabled
    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
```

1.7 Running the Messaging Server Debugging Scripts

This section describes how to run the dbhang and pkg_app scripts.

Note – The dbhang command (version 3) now uses pkginfo to find the *server-root* directory and from that gathers other necessary information from configutil, thereby avoiding the need to edit the script. You can still use command-line switches to override these defaults. A side-effect of this enhancement is that the new version works only on Messaging Server 6.0 and later releases. If you are running iPlanet Messaging Server 5.2, continue to use dbhang version 2.10.

▼ To Run the dbhang Script

The dbhang script gathers data about problems where the Message Store database is the source of the hang. The dbhang script is not intended to be an all-purpose data gathering script for Messaging Server.

1 **Customize the** dbhang **script to reflect the specificities of your Messaging Server installation. Use an editor to modify the script then change the** SRVROOT, INST, **and** MAILUSER **variables in the top part of the script.**

2 **Using the** configutil **command, check if the** store.dbtmpdir **parameter is set. If it is set, change** DBTMPDIR **to the same value as** store.dbtmpdir, **otherwise it should be** $MBOXLIST.

3 **Leave the following variables set to their default values unless otherwise instructed by Sun Support:**

```
doDBSTAT=1
doMBOXTAR=0
doSHARES=0
doGCORE=0
doPFILES=0
doDISPSTATS=0
doGETCONFIG=0
doPMFCTL=0
dashN=""
```

You should not need to change anything below this comment:

```
#
# you should not need to edit anything below here
#
```

Note – Occasionally, the `dbhang` script cannot find the log files in their expected location, so you might need to change this as well.

4 **Perform a test run of the `dbhang` script after you have made your edits. Ensure that the script runs without errors.**

5 **Execute the `dbhang` script and collect the data when you have a problem.**

▼ To Run the pkg_app Script

This script packages an executable and all of its shared libraries into one compressed tar file given 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.

On Solaris 9 OS or greater, the list of files is derived from the core file rather than the process image if it is specified. 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`. This is 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`. This script contains the `dbx` initialization commands to open the core file.

1 **Copy the script to a temporary directory on the system where Portal Server is installed.**

2 **Become `superuser`.**

3 **Execute the `pkg_app` script in one of the following three ways:**

- `./pkg_app` *pid-of-running-application corefile*
- `./pkg_app` *pid-of-the-running-application*
  (The *pkg_app* scripts prompts for the *corefile* name.)
- `./pkg_app` *core file*

1.8 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

1.9 Accessing Sun Resources Online

The 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)

1.10 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.

1.11 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 819–5355–10.