Applix *TM1 Technical Bulletin*

Debugging the TM1 Server

Date: 11/28/06
Relevant TM1 Versions: All

Overview

This document describes how to monitor the TM1 server and generate dump files for debugging. This bulletin is written for Applix personnel who must work with customers to troubleshoot server hangs and other conditions that cannot be diagnosed through log files or other methods.

This bulletin describes how to monitor the TM1 server and generate dump files on Windows, Solaris, and HP/UX.

Debugging the TM1 Server on Windows, 32- and 64-bit

This section describes how to monitor the TM1 server and collect dump files on a Windows system, either 32-bit or 64-bit. For general information, see http://support.microsoft.com/kb/q286350/.

Your server must have Debugging Tools for Windows installed. This software is available from Microsoft at this location:

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

Be sure to download the software that is appropriate for your Windows platform.

Debugging Tools for Windows will install many scripts. One of those scripts is called adplus.vbs. This is executed in conjunction with command line switches that will attach it to the tm1s process.

Capturing Dump Files when a TM1 Server Crashes

This section describes how to capture dump files when the TM1 server crashes, either due to a forced crash or to an error during processing.

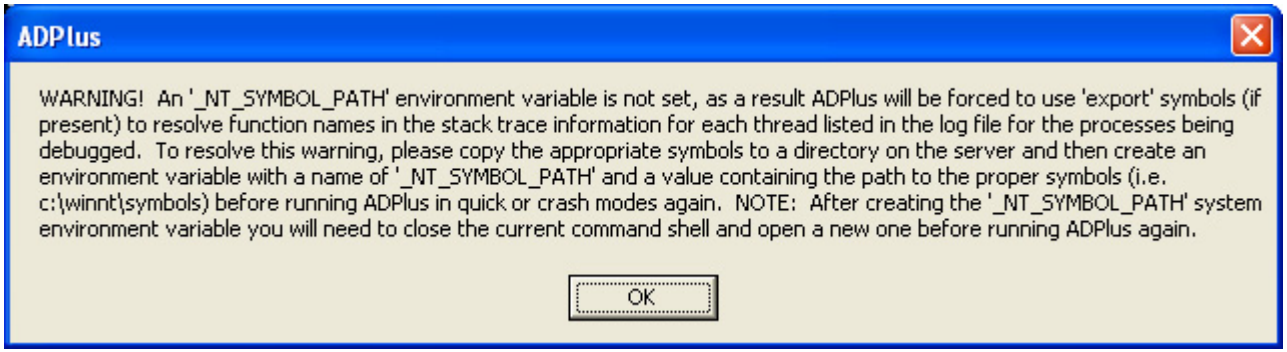
Preparing a Dirctory to Receive Dump Files

Follow these steps to set up a directory to receive the dump files that are generated when the TM1 server crashes:

1. Start the TM1 server.
2. Open a command prompt window and cd to the Debugging Tools for Windows directory (usually C:\Program Files\Debugging Tools for Windows).
3. Open the Windows Task Manager to retrieve the PID number for the TM1 server.
4. At the command prompt, type `adplus -crash -p [PID]`

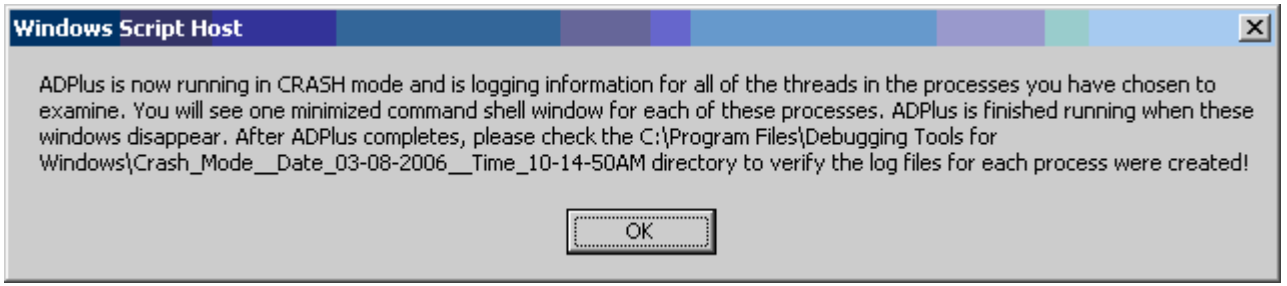
   For example, `adplus -crash -p 492`

   The following message appears:



5. Click **OK**.

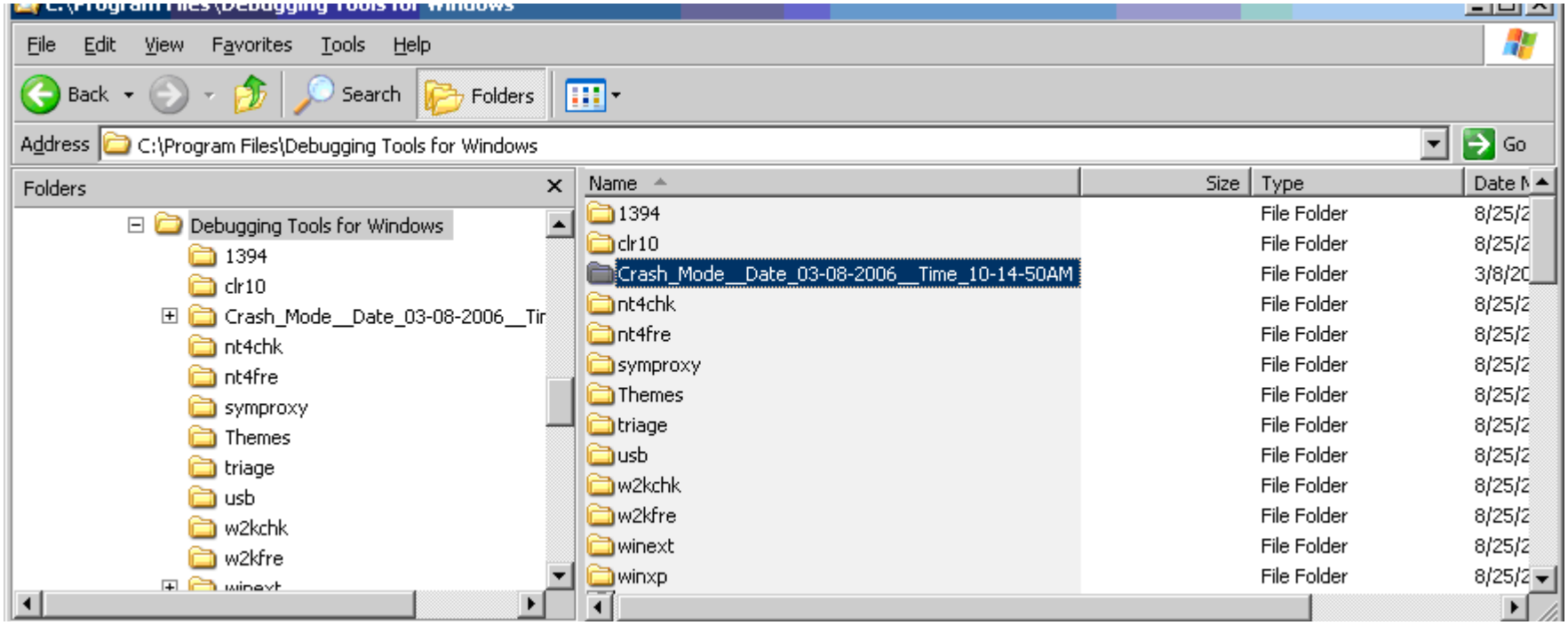
   The following message appears:



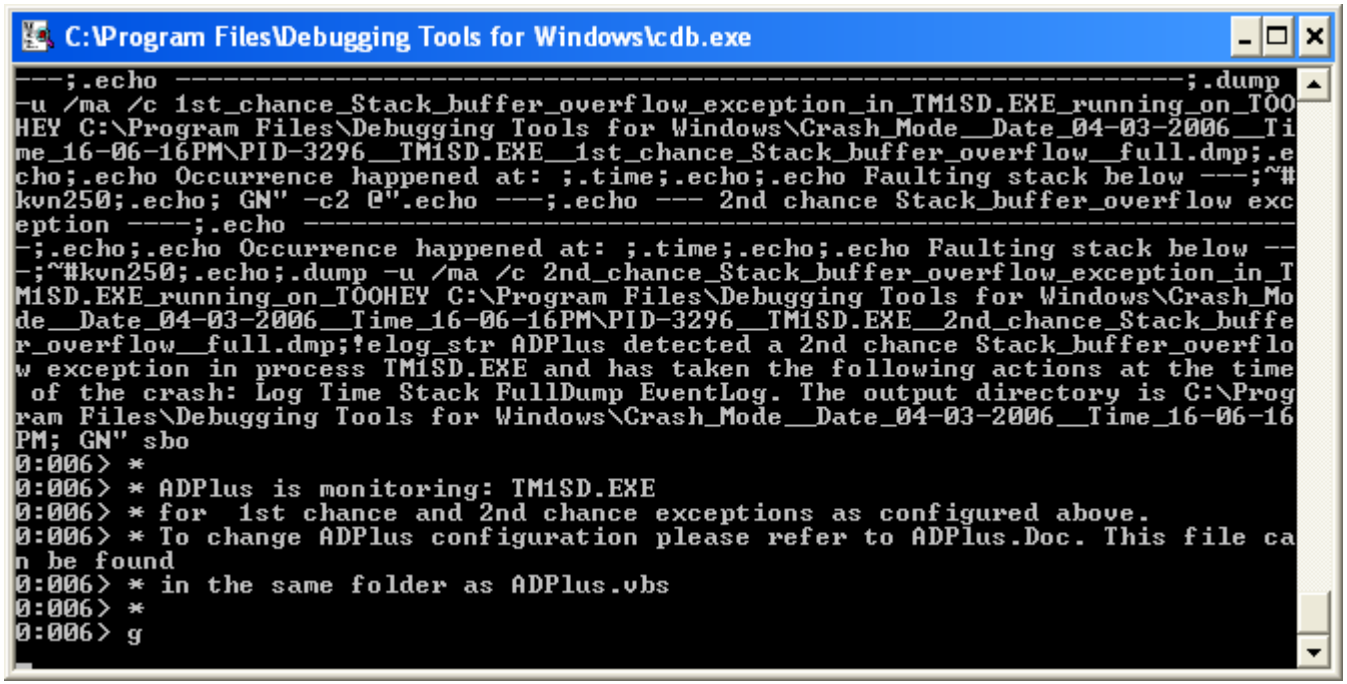
   Note that this message indicates that a new subdirectory will be created in the Debugging Tools for Window directory. The new subdirectory, which is named Crash_Mode_*DateStamp_TimeStamp*, receives the dump files that are generated when the TM1 server crashes.

6. Click **OK**.

   You can confirm the creation of the subdirectory in Windows Explorer.

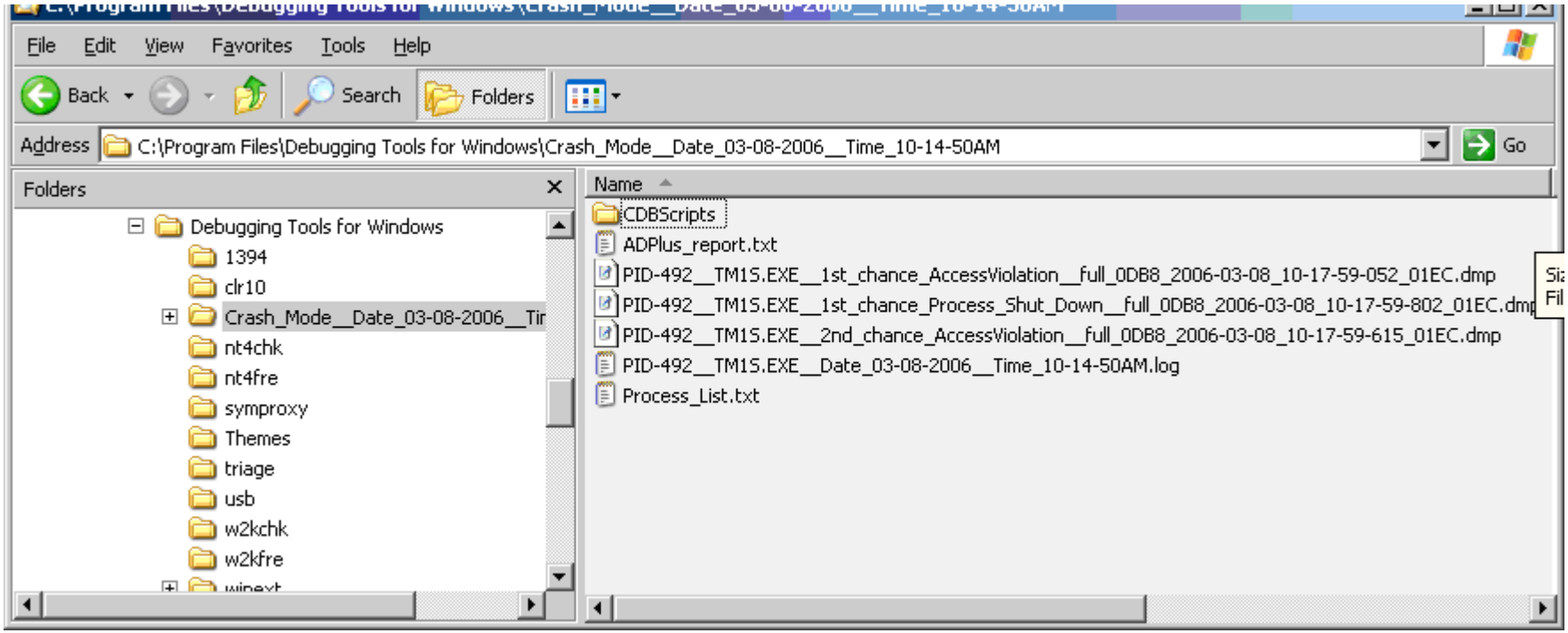


Also, note that a new command prompt window is now open on your desktop. The Microsoft Console Debugger (cdb.exe) runs in this window, as shown in the following image.



Monitoring the TM1 Server

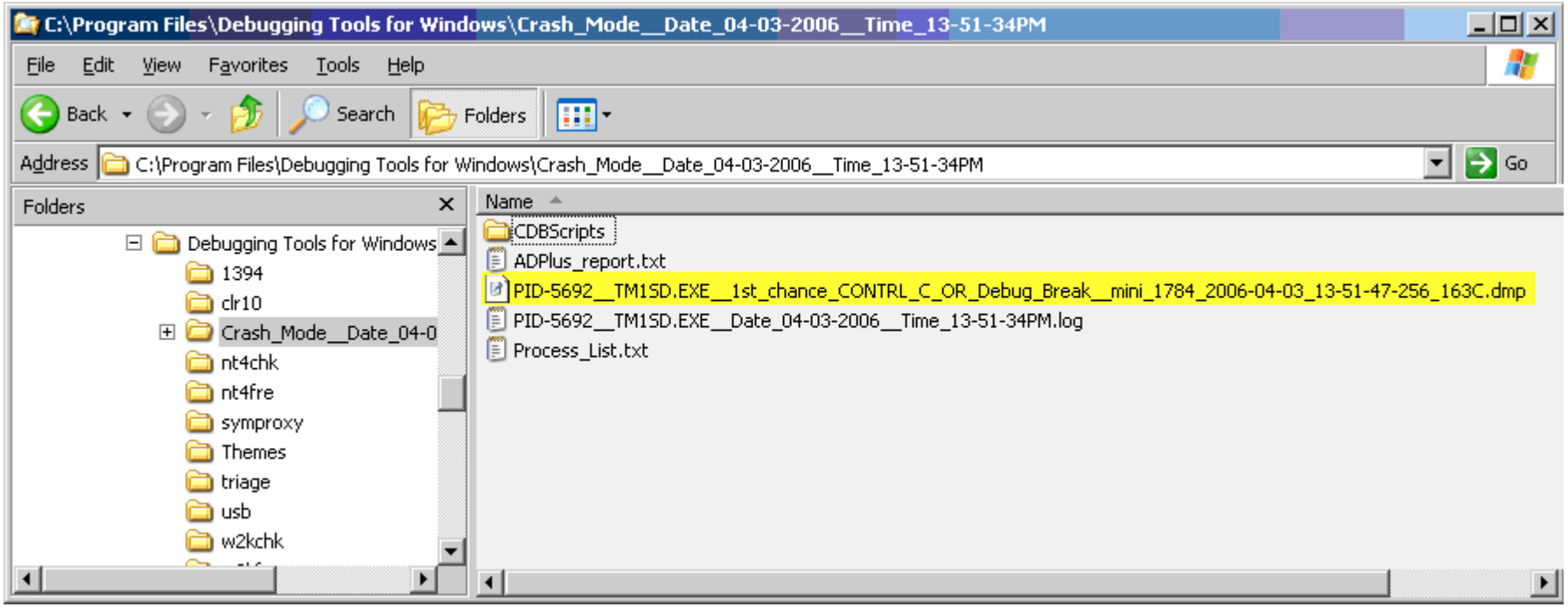
If you are monitoring the TM1 server, just continue your daily activity. When the TM1 server crashes, three dump files (.dmp) are created in the subdirectory. The only one Applix needs to debug the server crash is the ...1st chance_AccessViolation…file. This is typically the first dump file generated by adplus.



Using CTRL-C to Force a TM1 Server Crash

To force a server crash, enter CTRL-C in the cdb.exe command prompt window.

The resulting dump file will be named similar to the highlighted file below. This is the file Applix needs to debug the TM1 server.



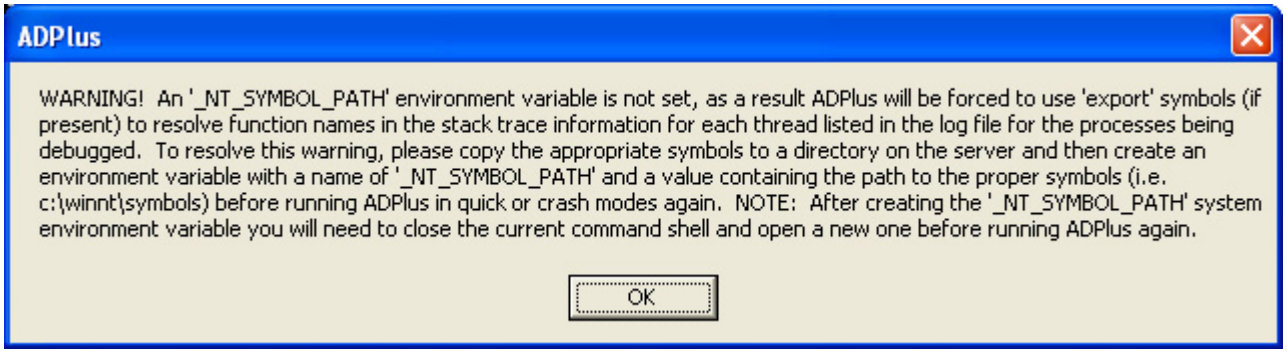
Taking a Snapshot of Current Server State

To take a snapshot of the current server state, but leave the TM1 server running:

1. Start the TM1 server.
2. Open a command prompt window and cd to the Debugging Tools for Windows directory (usually C:\Program Files\Debugging Tools for Windows).
3. Open the Windows Task Manager to retrieve the PID number for the TM1 server.
4. At the command prompt, type `adplus -hang -p [PID]`

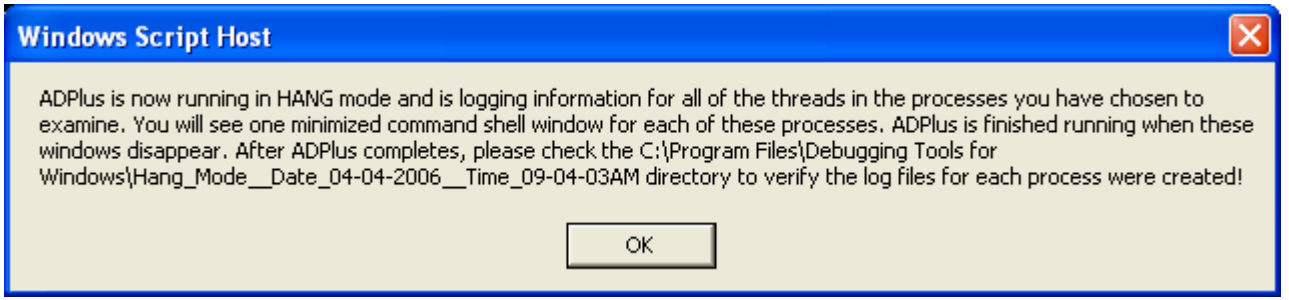
   For example, `adplus -hang -p 492`

   The following message appears:



5. Click **OK**.

   The following message appears:

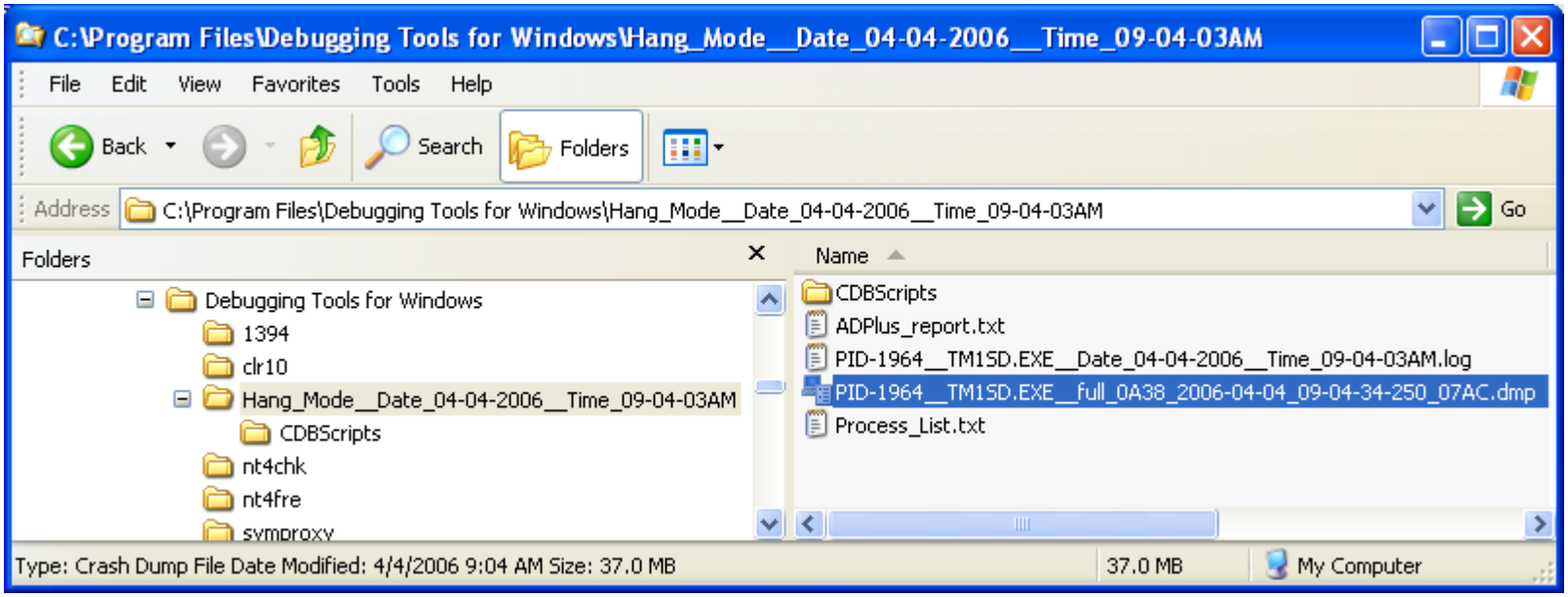


   Note that this message indicates that a new subdirectory will be created in the Debugging Tools for Window directory. The new subdirectory, which is named Hang_Mode_*DateStamp_TimeStamp*, receives the dump file that records the current TM1 server state.

6. Click **OK**.

   A new dump file is immediately generated, recording the current server state.

7. Open the new subdirectory in the Debugging Tools for Window directory and note the presence of a single dump file. This is the file Applix needs to debug your server.



Monitoring the TM1 Server on Solaris and HP/UX

At least one major customer has encountered conditions where the TM1 server hangs during processing. This section describes how to diagnose a hung TM1 server, and how to force a core dump when the condition occurs.

Identifying a Hung TM1 Server on Solaris

A TM1 Server that is hung displays the following symptoms:

- The TM1 server does not respond to user requests.
- The TM1 server process reports a 0% CPU load
- New client logons fail, and running clients freeze.

Use the Solaris command prstat -m -L -p <pid> to derive more information about your TM1 server. A TM1 server typically has 4 to 5 systems threads with thread Ids from 1 to 5. For instance, thread ID 4 is the 60 second heartbeat connection to the TM1 Admin Server. The server will also have plus multiple user threads. A TM1 Server in deadlock has all user threads in idle at 0% CPU load.

Methods of Generating a Core File

On Solaris and HP/UX, no special software is required. You can generate a core file using the following commands:

- Kill (both Solaris and HP/UX)
- gcore (Solaris command)
- gdb (HP/UX command)

Using Kill to Generate a Core File

You can use Kill to generate a core file on either a Solaris machine or an HP/UX machine. This command stops the TM1 server process. Follow these steps:

1. Start a TM1 Server.
2. At the shell prompt, perform a grep on the machine for TM1 processes.

   ```
   $Ps –def|grep tm1
   ```

3. At the shell prompt, force the crash and core dump with the kill command. For a bus error, use kill - 10. For a segmentation fault, use kill - 11.

   ```
   $kill –10 [PID]
   ```

   Or

   ```
   $kill –11 [PID]
   ```

A core file is generated that contains the dump.

Using gcore to Generate a Core File

You can use the gcore command on a running TM1 server. Like Kill, this command forces the generation of a core file, but does not bring down the server process. This Solaris command creates a coredump of a running process into the core dump directory set by the Solaris command coreadm.

Follow these steps to generate a core file with gcore.

1. Start a TM1 Server.
2. Use the following command to grep on the machine for TM1 processes.

   ```
   $Ps -def|grep tm1
   ```

3. Find the process id of the TM1 server. Enter the following command:

   ```
   gcore <pid>
   ```

Restarting the Server Process

The gcore command pauses the running server process. To restart the server, enter the following command:

```
prun <pid>
```

Setting Core File Locations in Solaris

Applix core files can be very large - a large TM1 model might leave a core file that is a gigabyte or more in size. Applix recommends using a global coredump directory on a non-essential partition, such as /opt or /export/home.

On a Solaris operating system, the coreadm command is used to specify the name and location of core files produced by abnormally terminating processes. Core file paths that include the zonename of the zone in which the process executed can be produced by specifying the %z variable. The path name is relative to a zone's root directory.

The coreadm utility allows you to redirect the location of the core file to a specific place, regardless of the current working directory of the process. This redirect of the core file can be done on a couple of levels. In the following example, assume that the directory /export/home/cores is on the local disk and is writeable to all users. All core files will be placed here.

All these commands take a core file pattern, which in this example is core.%f.%t. This writes core files of the form core.process_name.time_stamp. For example:

core.tm1s.exe.1113566515

The time_stamp means that cores can be collected in a single directory without overwriting one another.

Example 1

The first example shows how to redirect the core file for the TM1 server process only. To do this, start the TM1 server, get its process-id and enter the following command:

```
coreadm -p /export/home/cores/core.%f.%t process-id-of-TM1-server
```

If you enter the following core file command, you should get a result showing that the core location has been set, as shown here:

coreadm <process-id-of-TM1-server>

```
process-id-of-TM1-server :  /export/home/cores/core.%f.%t
```

Remember to do this operation every time the TM1 server is restarted.

Example 2

This example shows how to set the core location for all processes started by a particular user. Using this method, all children of a particular user inherit the same core file location. If a particular user is always used to start the TM1 server, place this

command in the startup script for that user to set the core file location whenever the TM1 server is started. For example:

```
coreadm -p /export/home/cores/core.%f.%t $$
```

You can check that this has been implemented correctly by checking the core file location of the running TM1 server with this command:

```
coreadm process-id-of-TM1-server
```

Example 3

This example shows how to set the global core file location, and enable global core files.

When a core file is written, either one or both of a 'per-process' or a 'global' core file is written also. The action to be taken is set with the coreadm utility. Check the current setting by using coreadm with no arguments.

To enable global core files:

```
coreadm -e global
```

To set the global core file location:

```
coreadm -g /export/home/cores/core.%f.%t
```

After these commands, coreadm with no arguments should produce something like this:

```
     global core file pattern: /export/home/cores/core.%f.%t
      init core file pattern:
           global core dumps: enabled
     per-process core dumps: enabled
    global setid core dumps: disabled
```

per-process setid core dumps: disabled

global core dump logging: disabled

If global core files are enabled, you might want to turn off per-process core files with:

```
coreadm -d process
```

With global core files enabled, and the core file location set, all core files from any process will end up in the central directory with the process name and the time as part of the core file name.

Looking at Core Files

You can generate a stack trace with this command:

```
pstack core-file
```

Using gdb to Generate a Core File

On an HP/UX machine, you can use the HP/UX debugger to attach to a running (or hung) TM1 process and generate a dump file. Run the following commands, in order, to start the debugger, attach to the TM1 server process, and generate the dump file:

```
/opt/langtools/bin/gdb

(gdb) attach [pid]
(gdb) dumpcore
(gdb) quit
```

Sending Core Files to Applix

When you send a UNIX core file to Applix, use the compress command to compress the file. Do not use ‘zip’ to compress it.

ADPlus 6.02.018 Usage Information

Command line switches for ADPlus.

Switch	Description
-Crash	Runs ADPlus in Crash mode
-Hang	Runs ADPlus in Hang mode
-p <PID>	Defines a Process ID to be monitored
-pn <ProcessName>	Defines a process name to be monitored
-sc <spawning command>	Defines the application and parameters to be started in the debugger
-iis	All IIS - related processes will be monitored (inetinfo, dllhost, mtx, etc.)
-o <output directory>	Defines the directory where logs and dumps are to be placed.
-quiet	No dialog boxes will be displayed
-notify <destination>	Will send a message to the destination
-c <config file name>	Defines a configuration file to be used
-ce <custom exception code>	Defines a custom exception to be monitored: -ce 0x80501001

Switch	Description
-bp <breakpoint parameters>	Sets a breakpoint Syntax: -bp address;optional_additional_parameters -bp MyModule!MyClass::MyMethod -bp MyModule!MyClass::MyMethod;MiniDump
-y <symbol path>	Defines the symbol path to be used
-yp <symbol path to add>	Defines an additional symbol path
-FullOnFirst	Sets ADPlus to create full dumps on first chance exceptions
-FullOnFirstOver	Sets ADPlus to create full dumps on first chance exceptions, overwriting the previous dump
-MiniOnSecond	Sets ADPlus to create mini dumps on second chance exceptions
-NoDumpOnFirst Sets	ADPlus to not create any dumps on first chance exceptions
-NoDumpOnSecond	Sets ADPlus to not create any dumps on second chance exceptions
-CTCF	Creates a full dump on CTL+C, and quits
-CTCFG	Creates a full dump on CTL+C, and resumes execution
-NoTlist	Will not use TList; only -p can be used (-pn and -iis will not work)

Switch	Description
-dbg <debugger>	Allows you to select the debugger to be used cdb, windbg or ntsd (default is cdb)

Required: ('-hang', or '-crash') AND ('-iis' or '-p' or '-pn')

If you are using a configuration file (-c switch), the required switches above can be provided in the config file or in the command line.

If you use the -sc switch, it must be the last one.

Examples

```
ADPlus -hang -iis
```

Produces memory dumps of IIS and all MTS/COM+ packages currently running.

```
ADPlus -crash -p 1896
```

Attaches the debugger to process with PID1896, and monitors it for 1st and 2nd chance access violations (crashes).

```
ADPlus -?
ADPlus -help
```

Displays ADPlus help.

HELP and Documentation

For more detailed information on how to use and config ADPlus please see the debugger's help file (debugger.chm) under the following hierarchy:

Using Debugging Tools for Windows

Crash dumps

User mode dump files

Creating a user mode dump file

ADPlus