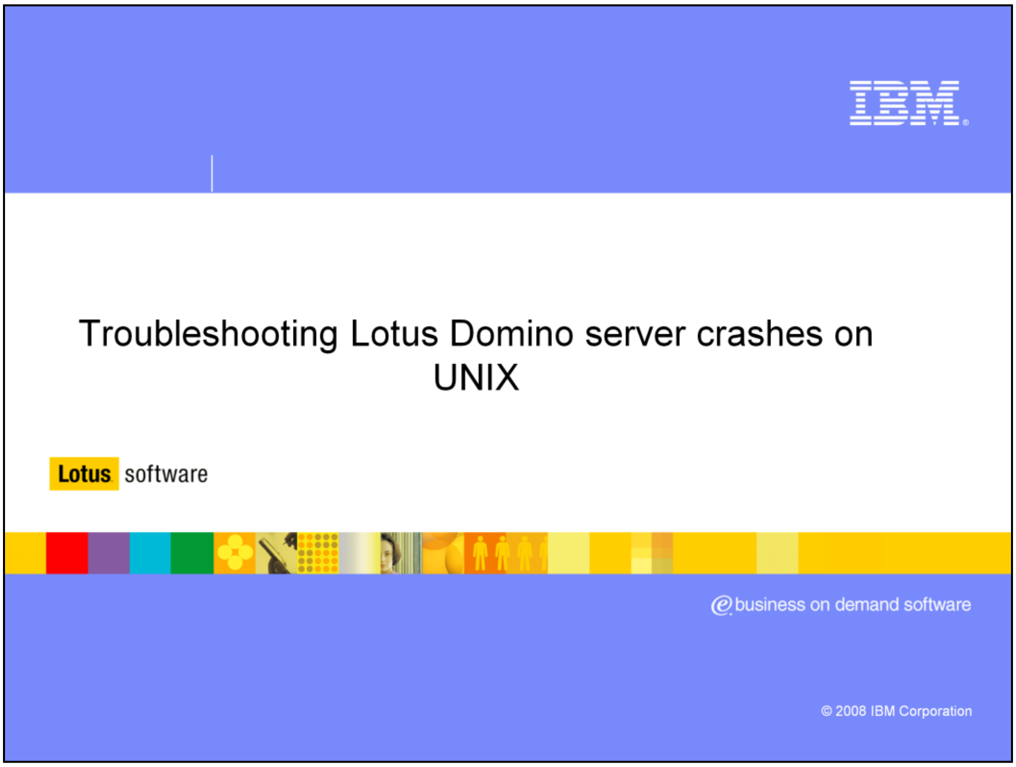
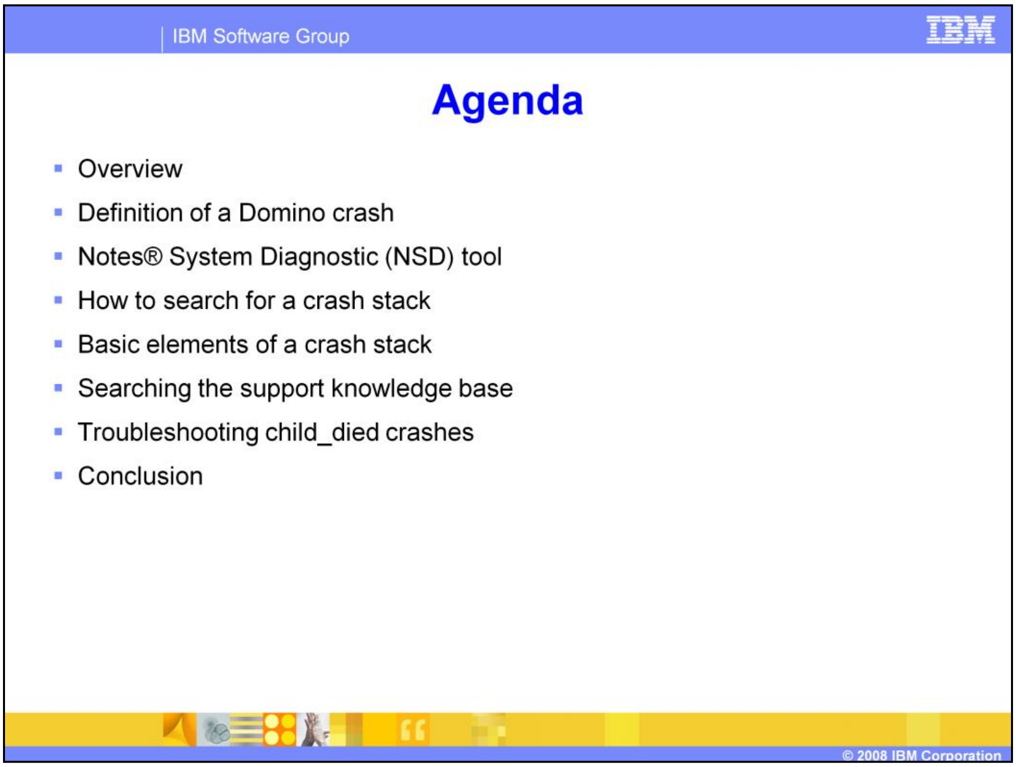


Troubleshooting Lotus Domino server crashes on UNIX

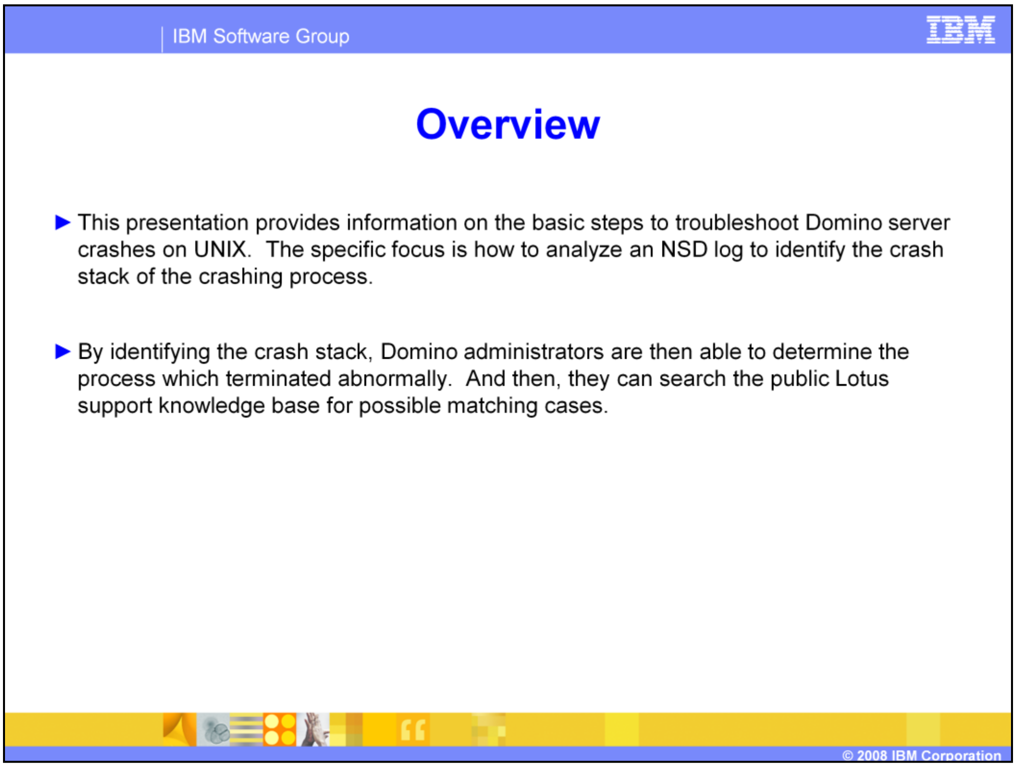
Lotus software

business on demand software



Agenda

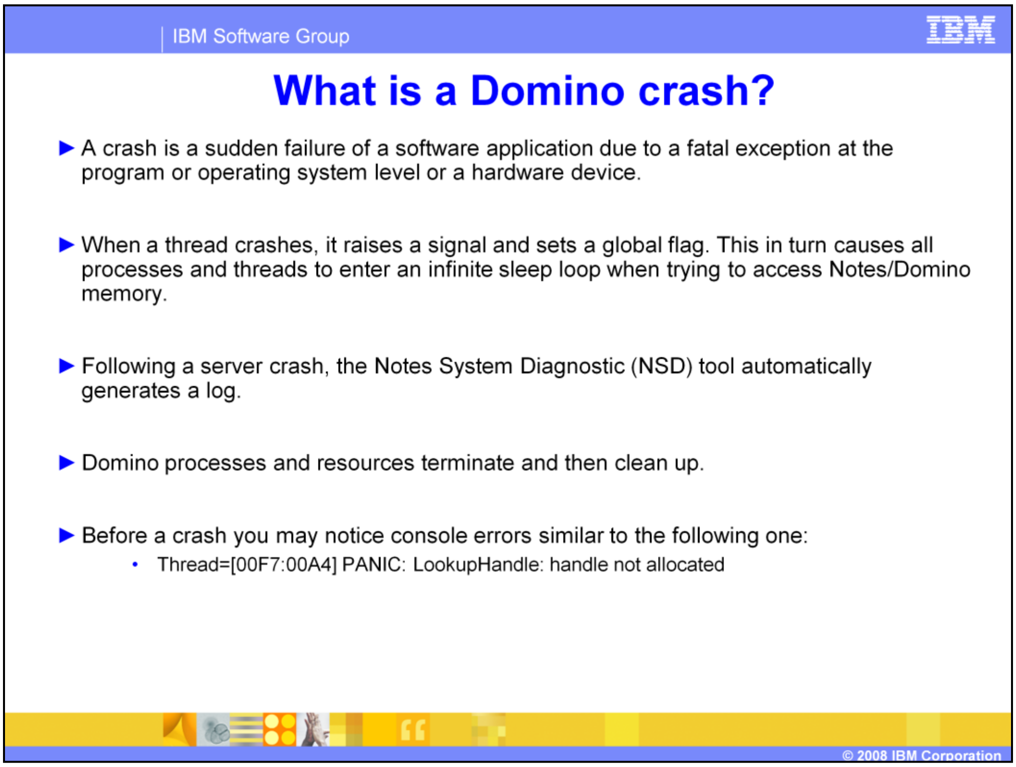
- Overview
- Definition of a Domino crash
- Notes® System Diagnostic (NSD) tool
- How to search for a crash stack
- Basic elements of a crash stack
- Searching the support knowledge base
- Troubleshooting child_died crashes
- Conclusion



Overview

- This presentation provides information on the basic steps to troubleshoot Domino server crashes on UNIX. The specific focus is how to analyze an NSD log to identify the crash stack of the crashing process.

- By identifying the crash stack, Domino administrators are then able to determine the process which terminated abnormally. And then, they can search the public Lotus support knowledge base for possible matching cases.



What is a Domino crash?

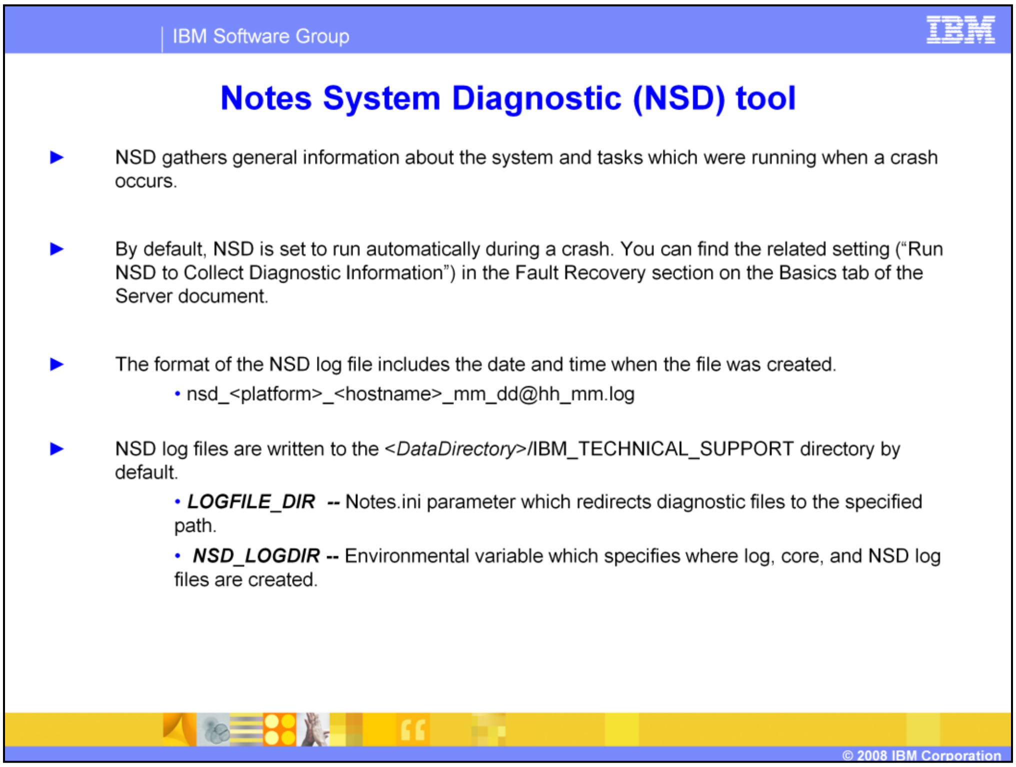
- A crash is a sudden failure of a software application due to a fatal exception at the program or operating system level or a hardware device.

- When a thread crashes, it raises a signal and sets a global flag. This in turn causes all processes and threads to enter an infinite sleep loop when trying to access Notes/Domino memory.

- Following a server crash, the Notes System Diagnostic (NSD) tool automatically generates a log.

- Domino processes and resources terminate and then clean up.

- Before a crash you may notice console errors similar to the following one:
  - Thread=[00F7:00A4] PANIC: LookupHandle: handle not allocated



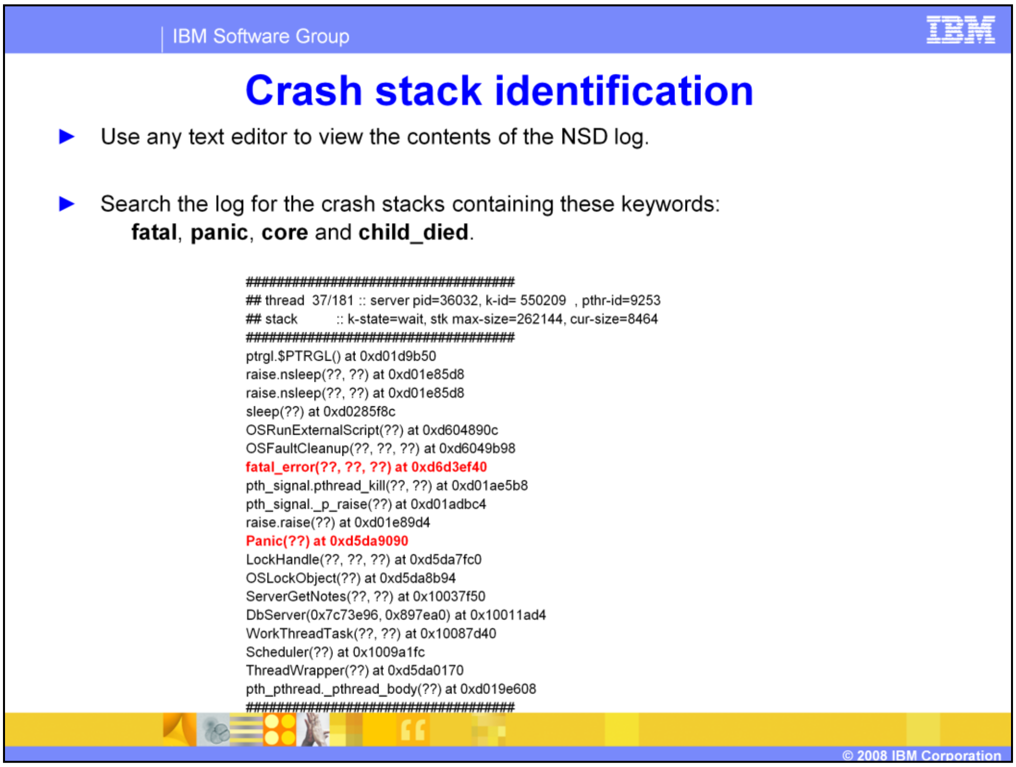
Notes System Diagnostic (NSD) tool

- NSD gathers general information about the system and tasks which were running when a crash occurs.

- By default, NSD is set to run automatically during a crash. You can find the related setting (“Run NSD to Collect Diagnostic Information”) in the Fault Recovery section on the Basics tab of the Server document.

- The format of the NSD log file includes the date and time when the file was created.
  - nsd_<platform>_<hostname>_mm_dd@hh_mm.log

- NSD log files are written to the <*DataDirectory*>/IBM_TECHNICAL_SUPPORT directory by default.
  - ***LOGFILE_DIR*** **--** Notes.ini parameter which redirects diagnostic files to the specified path.
  - ***NSD_LOGDIR*** **--** Environmental variable which specifies where log, core, and NSD log files are created.



Crash stack identification

- Use any text editor to view the contents of the NSD log.

- Search the log for the crash stacks containing these keywords:
  fatal, **panic**, **core** and **child_died**.

```
#####################################
## thread  37/181 :: server pid=36032, k-id= 550209  , pthr-id=9253
## stack          :: k-state=wait, stk max-size=262144, cur-size=8464
#####################################
ptrgl.$PTRGL() at 0xd01d9b50
raise.nsleep(??, ??) at 0xd01e85d8
raise.nsleep(??, ??) at 0xd01e85d8
sleep(??) at 0xd0285f8c
OSRunExternalScript(??) at 0xd604890c
OSFaultCleanup(??, ??, ??) at 0xd6049b98
fatal_error(??, ??, ??) at 0xd6d3ef40
pth_signal.pthread_kill(??, ??) at 0xd01ae5b8
pth_signal._p_raise(??) at 0xd01adbc4
raise.raise(??) at 0xd01e89d4
Panic(??) at 0xd5da9090
LockHandle(??, ??, ??) at 0xd5da7fc0
OSLockObject(??) at 0xd5da8b94
ServerGetNotes(??, ??) at 0x10037f50
DbServer(0x7c73e96, 0x897ea0) at 0x10011ad4
WorkThreadTask(??, ??) at 0x10087d40
Scheduler(??) at 0x1009a1fc
ThreadWrapper(??) at 0xd5da0170
pth_pthread._pthread_body(??) at 0xd019e608
#####################################
```

Call stack formats

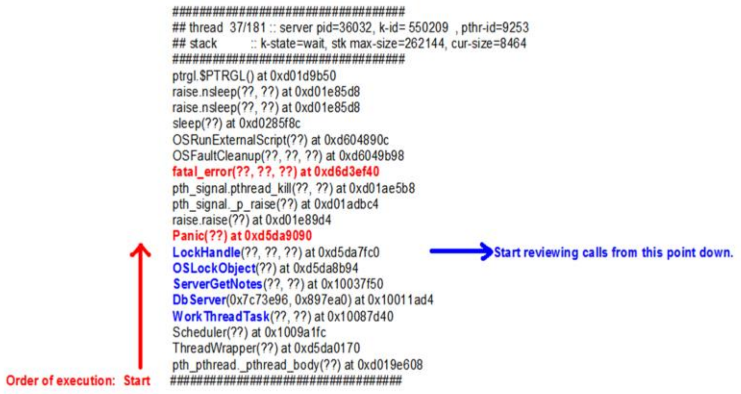
- Call stack formats differ based on the operating system. For example, AIX and Solaris call stacks contain a header section which identifies the process ID and name of the related task.

- Using the example below, note that the crash occurred on the server process, the server's process ID is 36032, and the physical thread ID of the crashing thread is 9253.

```
#####################################
 ## thread  37/181 :: server pid=36032, k-id= 550209  , pthr-id=9253
 ## stack         :: k-state=wait, stk max-size=262144, cur-size=8464
 ##################################
 ptrgl.$PTRGL() at 0xd01d9b50
 raise.nsleep(??, ??) at 0xd01e85d8
 raise.nsleep(??, ??) at 0xd01e85d8
 sleep(??) at 0xd0285f8c
 OSRunExternalScript(??) at 0xd604890c
 OSFaultCleanup(??, ??, ??) at 0xd6049b98
 fatal_error(??, ??, ??) at 0xd6d3ef40
 pth_signal.pthread_kill(??, ??) at 0xd01ae5b8
 pth_signal._p_raise(??) at 0xd01adbc4
 raise.raise(??) at 0xd01e89d4
 Panic(??) at 0xd5da9090
 LockHandle(??, ??, ??) at 0xd5da7fc0
 OSLockObject(??) at 0xd5da8b94
 ServerGetNotes(??, ??) at 0x10037f50
 DbServer(0x7c73e96, 0x897ea0) at 0x10011ad4
 WorkThreadTask(??, ??) at 0x10087d40
 Scheduler(??) at 0x1009a1fc
 ThreadWrapper(??) at 0xd5da0170
 pth_pthread._pthread_body(??) at 0xd019e608
```

Stack data structure

- Function calls list chronologically from the bottom up with the top calls being the last ones executed by the process.
- Review the function calls listed before the signal handler functions (fatal, panic, pth_signal). This is the area of the code that was executing prior to the outage.



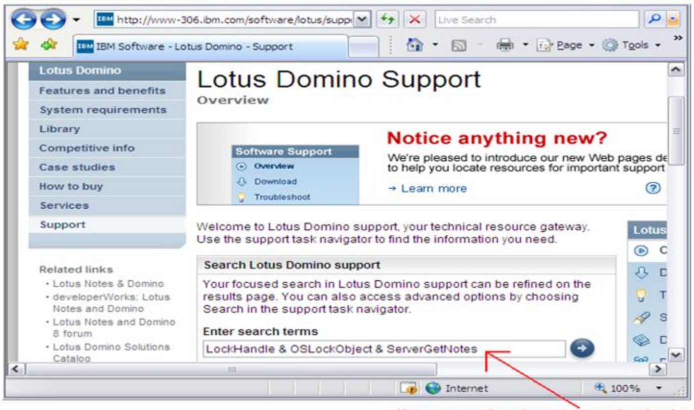
```
##################################
## thread  37/181 :: server pid=36032, k-id= 550209  , pthr-id=9253
## stack         :: k-state=wait, stk max-size=262144, cur-size=8464
##################################
ptrgl.$PTRGL() at 0xd01d9b50
raise.nsleep(??, ??) at 0xd01e85d8
raise.nsleep(??, ??) at 0xd01e85d8
sleep(??) at 0xd0285f8c
OSRunExternalScript(??) at 0xd604890c
OSFaultCleanup(??, ??, ??) at 0xd6049b98
fatal_error(??, ??, ??) at 0xd6d3ef40
pth_signal.pthread_kill(??, ??) at 0xd01ae5b8
pth_signal._p_raise(??) at 0xd01adbc4
raise.raise(??) at 0xd01e89d4
Panic(??) at 0xd5da9090
LockHandle(??, ??, ??) at 0xd5da7fc0
OSLockObject(??) at 0xd5da8b94
ServerGetNotes(??, ??) at 0x10037f50
DbServer(0x7c73e96, 0x897ea0) at 0x10011ad4
WorkThreadTask(??, ??) at 0x10087d40
Scheduler(??) at 0x1009a1fc
ThreadWrapper(??) at 0xd5da0170
pth_pthread._pthread_body(??) at 0xd019e608
##################################
```

Start reviewing calls from this point down.

Order of execution: Start

Lotus support knowledge base

- You can search the Lotus support knowledge base for technotes on function calls that you identify in the fatal stack.
- http://www.ibm.com/software/lotus/support/domino/support.html



Lotus Domino Support

Overview

Notice anything new?

We're pleased to introduce our new Web pages de to help you locate resources for important support

Welcome to Lotus Domino support, your technical resource gateway. Use the support task navigator to find the information you need.

Search Lotus Domino support

Your focused search in Lotus Domino support can be refined on the results page. You can also access advanced options by choosing Search in the support task navigator.

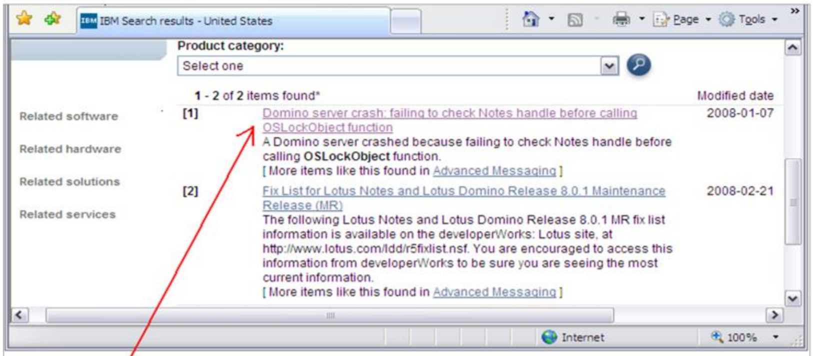
Enter search terms

LockHandle & OSLockObject & ServerGetNotes

You may need to alter your search string by including additional function calls from the fatal stack to narrow down your search.

Review search results

- Review the documents found and compare the crash stack, symptoms, and Domino versions to determine if the issues match.
- If there is no matching documentation or if you are uncertain whether the crash is a match, provide IBM® Lotus Support with the NSD and console log for further review.



Product category:

Select one

1 - 2 of 2 items found*   Modified date

[1] Domino server crash: failing to check Notes handle before calling OSLockObject function — 2008-01-07

A Domino server crashed because failing to check Notes handle before calling **OSLockObject** function.

[More items like this found in Advanced Messaging]

[2] Fix List for Lotus Notes and Lotus Domino Release 8.0.1 Maintenance Release (MR) — 2008-02-21

The following Lotus Notes and Lotus Domino Release 8.0.1 MR fix list information is available on the developerWorks: Lotus site, at http://www.lotus.com/ldd/r5fixlist.nsf. You are encouraged to access this information from developerWorks to be sure you are seeing the most current information.

[More items like this found in Advanced Messaging]

In this particular example the first document is a match to the outage.
The crash stack is an exact match and the server versions is at a release
where the issue has not been addressed.

Reference Doc. # 1258247

Shared OS or MM/OS Structure Information

- If you do not find a crash stack, you can also search the Shared OS or MM/OS Structure Information fields within the memcheck portion of the NSD log file.
- The information contained in these fields are useful when there is crash on a third-party application or when there are multiple crash stacks in the NSD.
- The StaticHang field shows the crashing process thread.

```
        <@@ -- Notes Memory Analyzer (memcheck) -> MM/OS Structure Information (Time 14:46:17) --@@>
Start Time = 10/20/2007 17:43:22
Crash Time = 10/30/2007 14:42:50
Error Message = PANIC: LookupHandle: null handle
SharedDPoolSize = 8388608
FaultRecovery = 0x00010013
Cleanup Script Timeout= 300
Crash Limits = 3 crashes in 5 minutes
Core Dump Path =
StaticHang = [  server:36032:2540]/ [  server:36032:9253] (0x8cc0/0x9ec/0x2425)
ConfigFileSem =  (  SEM:#0:0x010d) n=0, wcnt=-1, Users=-1,  Owner=[        :   0]
FDSem       =  ( RWSEM:#11:0x410f) rdcnt=-1, refcnt=0 Writer=[        :   0], n=11, wcnt=-1, Users=0,  swCnt=0, ssCnt=0 nat:
semid=4980766, lpid=18754, val=0  Owner=[        :   0]
```

Format: [proc name:PID:VTID] /[proc name:PID:PTID]
Process Thread that set the StaticHang global variable.

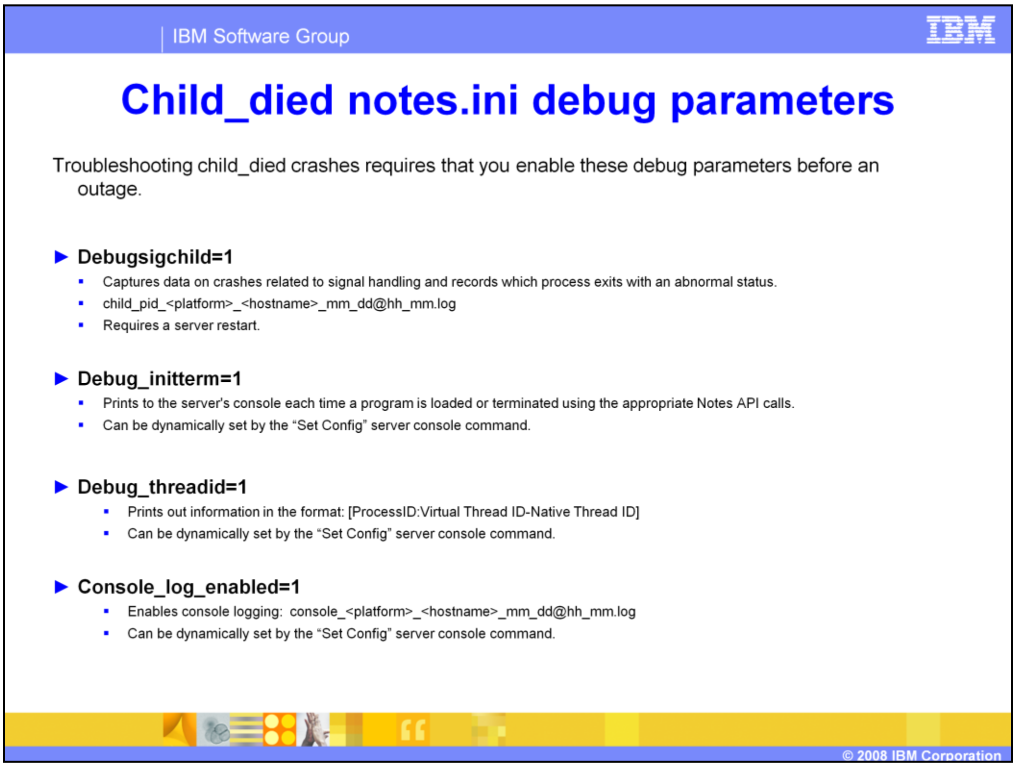
Note: The “Shared OS” field appears in Domino 6.5.4 and earlier versions. The equivalent field in Domino 6.5.5 and later is “MM/OS Structure Information.”

Child_died crashes

- Indicates that the thread was the parent process of another process which terminated without formally going through Domino's OS structure.

- The stack for this type of outage references the parent process rather than the actual child process which initiated the fault.

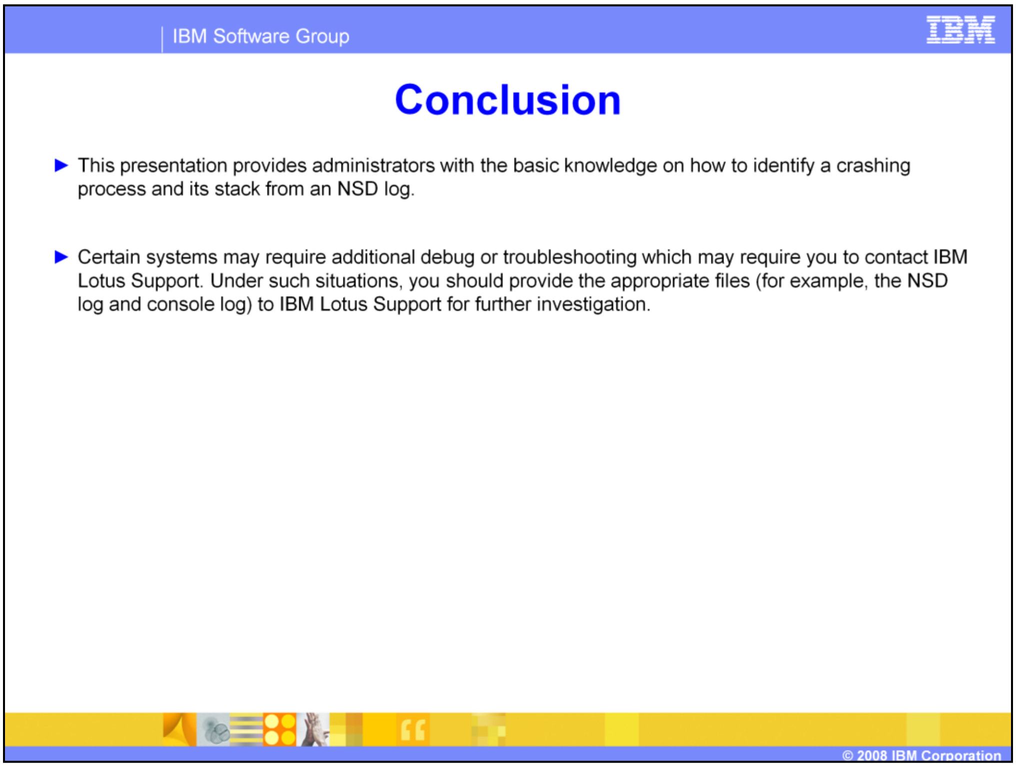
```
########################################
###### thread 1/144 :: server, pid=22914, lwp=1, tid=1 ######
########################################
[1]  ff2c0cc4 nanosleep (ffbed198, ffbed190)
[2]  fcec5b44 OSRunExternalScript (11e, 40, ffbed988, feca3278, 1, fef871ec) + 524
[3]  fcec1424 OSFaultCleanupExt (fb800000, 1e4c00, 0, 1, 5982, 1e4c00) + 924
[4]  fcec0ab4 OSFaultCleanup (0, 0, 0, 31000, 30478, 30400) + 14
[5]  fce6e8c8 ChildCleanup (30000, 9, 0, 22, 0, feca3278) + 588
[6]  fce6e2e4 child_died (ffbeeb08, ffbeef30, 40, 2f800, 2c800, feca3278) + 264
[7]  ff2c0a14 __sighndlr (12, ffbeef30, ffbeec78, fce6e080, 0, 2) + c
[8]  ff2b5a30 call_user_handler (12, 10000, 2000c, 0, fc8f2000, ffbeec78) + 3b8
[9]  ff2c16b8 __pollsys (0, 0, ffbef070, 0, 0, 1388) + 8
[10]  ff25d170 poll     (0, 0, 1388, 10624c00, 0, 0) + 7c
[11]  fcf035e4 unix_usleep (0, fb689104, feca3278, 2d11fc, 4e200, 2710) + 44
[12]  0002fb30 ServerPoller (0, 80, fb689104, 0, 1313e8, 1000) + 50
[13]  0002e9bc ServerMain (0, 129c94, 30, 0, 0, 0) + 51c
[14]  0002de7c main     (1, ffbff78c, 0, ffffc000, 129c94, fc8b6140) + 13c
[15]  0002da68 _start   (0, 0, 0, 0, 0, 0) + 108
########################################
```



Child_died notes.ini debug parameters

Troubleshooting child_died crashes requires that you enable these debug parameters before an outage.

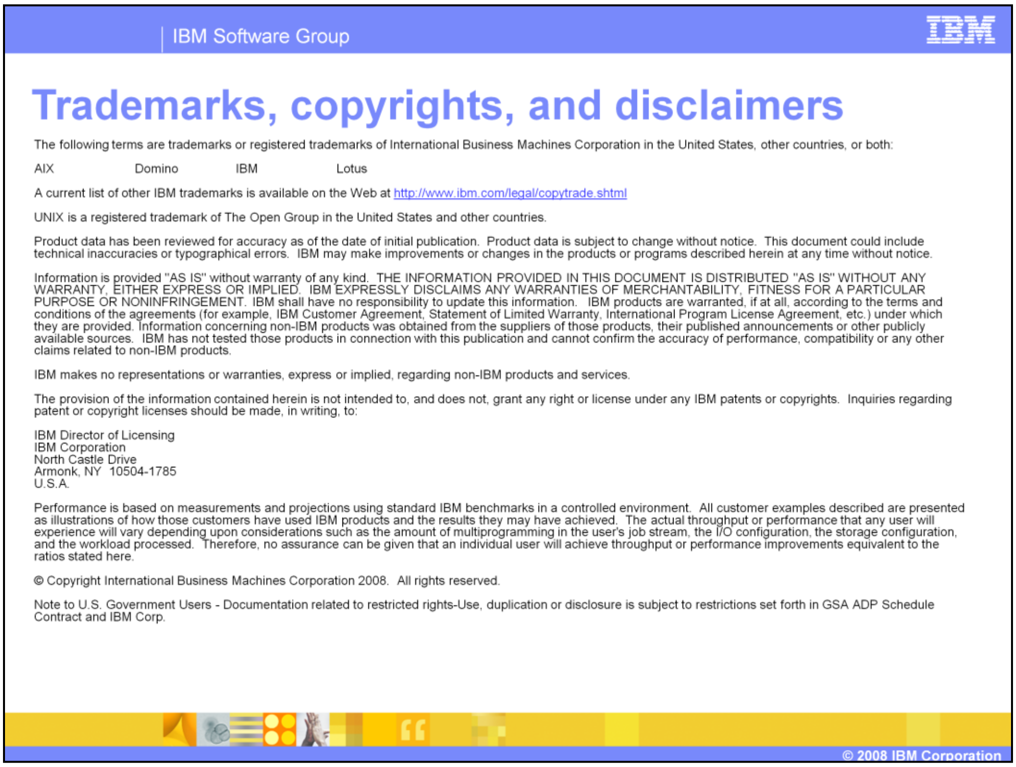
- **Debugsigchild=1**
  - Captures data on crashes related to signal handling and records which process exits with an abnormal status.
  - child_pid_<platform>_<hostname>_mm_dd@hh_mm.log
  - Requires a server restart.
- **Debug_initterm=1**
  - Prints to the server's console each time a program is loaded or terminated using the appropriate Notes API calls.
  - Can be dynamically set by the "Set Config" server console command.
- **Debug_threadid=1**
  - Prints out information in the format: [ProcessID:Virtual Thread ID-Native Thread ID]
  - Can be dynamically set by the "Set Config" server console command.
- **Console_log_enabled=1**
  - Enables console logging: console_<platform>_<hostname>_mm_dd@hh_mm.log
  - Can be dynamically set by the "Set Config" server console command.



Conclusion

- This presentation provides administrators with the basic knowledge on how to identify a crashing process and its stack from an NSD log.

- Certain systems may require additional debug or troubleshooting which may require you to contact IBM Lotus Support. Under such situations, you should provide the appropriate files (for example, the NSD log and console log) to IBM Lotus Support for further investigation.



Trademarks, copyrights, and disclaimers

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.