

Harvesting Error Reports from Windows Systems

Vince Orgovan
Microsoft Corp
Redmond WA USA
Vince.Orgovan@Microsoft.com

Abstract

Microsoft provides infrastructure to harvest and analyze error reports from Windows XP systems. This infrastructure makes it possible for IT professionals to harvest and analyze their organization's error reports with relatively modest investments, even on networks not connected to the Internet. This paper introduces this infrastructure and describes how it can be harnessed.

1. Error Reporting on Windows Systems

Microsoft incorporates broad error reporting support into the Windows XP and Windows Server 2003 operating systems. Microsoft uses this infrastructure to collect and analyze error reports to improve the Windows ecosystem. [1]

Windows systems can be easily configured to send application crash, application hang and system crash reports to a network server instead of to Microsoft. Microsoft calls this scenario "Corporate Error Reporting". The Windows debugger can then be used to analyze these reports.

Harvesting and analyzing error reports requires a few simple steps:

1. Configure systems for error collection.
2. Test the error reporting collection.
3. Analyze error reports.

The sections below guide a Windows IT professional through these steps for system crash reports. Other error reports are similarly collected and analyzed, but are not further discussed here.

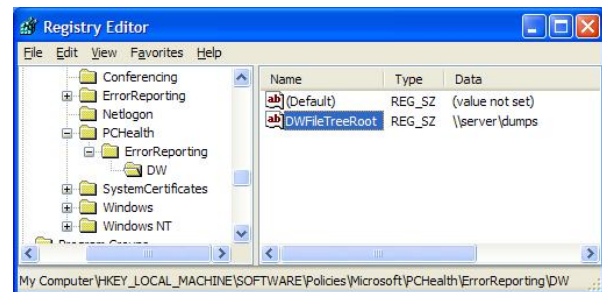
Note that adequate steps should be taken to safeguard the security and privacy of error reports, especially system crash dumps which can contain sensitive information.

2. Configuring Error Reporting

The first configuration step is to establish a repository for error reports. This is nothing more than designating one or more Windows systems with file shares that have enough free space for the expected error reports. The examples that follow assume the file share [\\server\dumps](#) is created for this purpose.

The next step is to configure error reporting client systems. Note the term client here refers to any system sending error reports, whether it's Windows XP or Windows Server 2003. Client system error reporting is controlled by a set of registry keys.

The most important registry key is `HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\PCHealth\ErrorReporting\DW`. It should contain a string value `DWFileTreeRoot` that points to the file share, e.g. [\\server\dumps](#). For example:



If silent (unprompted) error collection is desired, these values should also be set.

```
[HKLM\SOFTWARE\Policies\Microsoft\PCHealth\ErrorReporting]
"DoReport"=dword:00000001
"ShowUI"=dword:00000000
[HKLM\Policies\Microsoft\PCHealth\ErrorReporting\DW]
"DWAllowHeadless"=dword:00000001
```

Group policy tools make it easy to set these values on every system from which errors report collection is desired. [2]

3. Testing Error Reporting

Once Windows error reporting is configured on the server and clients, it is easy to test. One method is to force a Windows system crash (aka “bluescreen”).

The Sysinternals website contains a wealth of Windows system tools. One of these tools is NotMyFault, distributed with the Windows Internals book, a program and driver that can crash a Windows XP system in several different ways. [3]

Download and install NotMyFault onto a client system configured for corporate error reporting. Run NotMyFault.exe, select a crash type and click on “Do Bug”. This forces the specified crash type. When the client system reboots, logon to an administrative account. You will receive the message “Your system has recovered from a serious error” and be prompted to report the error. Click “Send” to upload the crash report to your server.

Finally, examine [\\server\dumps](#):

```
D:\>dir /s /b \\server\dumps
\\server\dumps\cabs
\\server\dumps\counts
\\server\dumps\cabs\blue
\\server\dumps\cabs\blue\00017673.cab
\\server\dumps\counts\blue
\\server\dumps\counts\blue\count.txt
D:\>type \\server\dumps\counts\blue\*.txt
Cabs Gathered=1
Total Hits=1
```

Each system crash is saved in a .cab file consisting of three files: a version.txt file that specifies the Windows version information, a sysdata.xml file that specifies the configuration of the crashed system, and a crash minidump file with the details of the crash.

4. Analyzing Error Reporting

The dump files from Windows error reports are analyzed using the freely available Windows debugger. [4] Download the debugger and install it on a system that will analyze error reports. Start with the debugger’s !analyze command to examine the test crash report.

```
D:\Debuggers>kd -y
SRV*http://msdl.microsoft.com/download/symbols
-z \\server\dumps\cabs\blue\00017673.cab

...Windows Debugger Version 6.6.0007.5
Extracted D:\Temp\Mini022507_02.dmp from
  \\server\dumps\cabs\blue\00017673.cab...

0: kd> !analyze -v
DRIVER_IRQL_NOT_LESS_OR_EQUAL (d1)
Arguments:
Arg1: e235b800, memory referenced
Arg2: 0000001c, IRQL
Arg3: 00000000, value 0 = read operation
Arg4: f8c873dd, adr which referenced memory
```

```
CURRENT_IRQL: 1c
PROCESS_NAME: NotMyFault.exe

STACK_TEXT:
f8785c64 myfault+0x3dd
f8785d00 nt!IopXxxControlFile+0x5ef
f8785d34 nt!NtDeviceIoControlFile+0x2a
f8785d34 nt!KiFastCallEntry+0xfc
0012fa54 0x7c90eb94

BUCKET_ID: 0xD1_myfault+3dd
```

This debugger command assigns each error report to a “bucket”, a collection of dumps assumed to have a single root cause. Buckets for system crashes consists of a crash stop code and a blamed code module. The debugger has numerous heuristics that attempt to make analyze accurate.

There are several sources of information about analyzing Windows error reports. The Windows debugger documentation is a good starting point. [4] The latest edition of the Windows Internals book includes a new chapter on system crash debugging. [5] Also, there is a TechNet webcast on system crash debugging techniques. [6]

The Windows infrastructure makes it straightforward for organizations to harvest and analyze their error reports with modest investments. Some universities are already successfully analyzing error reports in their environments. [7] Users and product developers can both benefit from an improved understanding of these reports.

5. References

- [1] [How WER Collects and Classifies Error Reports](#)
- [2] [Corporate Error Reporting Architecture](#), 2003
- [3] [Sysinternals NotMyFault](#), Feb 2007
- [4] [Overview of Debugging Tools for Windows](#)
- [5] [Microsoft Windows Internals](#), Mark Russinovich, David Solomon, Dec 2004
- [6] [Windows Hang and Crash Dump Analysis](#), Mark Russinovich, June 2006
- [7] Archana Ganapathi, [Windows XP Kernel Crash Analysis](#), *Proceedings of the 20th Large Installation System Administration Conference*, December 2006