Getting Started With WinQual For Windows Error Reporting (WER)

Contents

1. Introduction

If you develop software for Windows you need to learn how to access Windows Error Reporting (WER) data from WinQual. WinQual provides you with:

- **Intelligence**: How often does your software crash or hang? On which operating systems? Which languages?
- **Debugging**: Access to crash dumps that you can debug to understand why your software is crashing or hanging.
- **Responses**: Provide patches, new releases or troubleshooting to users via Windows *Problem Reports and Solutions*.

This guide contains the essential steps that you need to follow to start taking advantage of WinQual data.

Chapters 2, 3 and 4 cover creating a symbol server for your company, indexing symbols to provide debugger access to source code and mapping product deliverables for WinQual. In order to successfully debug crash dumps your build process should automate all three of these steps. If you already have a symbol server with source code indexing skip ahead to Chapter 4 to learn how to create WinQual product mapping files.

Chapter 5 describes how to configure the Debugging Tools for Windows and Visual Studio for crash dump debugging.

Chapter 6 guides you through the process of signing up for a WinQual account.

Chapter 7 provides documentation for using the Windows Error Reporting related sections of the WinQual site.

Chapter 8 covers special considerations for debugging managed crash dumps.

Chapter 9 lists books, blogs and other resources for post-mortem crash debugging.

2. Symbol Server Setup

Symbol files[1] contain essential information for debugging including variable names, source code line numbers and frame pointer omission (FPO) records. Debugging crash dumps is practically impossible without the correct symbols.

Storing symbols is an easy step to add to your build. You should do this for every build that you might need to debug later and certainly for any build that ships to QA or customers.

To get started install the latest Debugging Tools for Windows[2]. These tools are available as part of the Microsoft Windows SDK for X86, X64 and Itanium systems. Use the web installer and look for the tools under *Common Utilities -> Debugging Tools* (to install for the current platform) or *Redistributable Packages -> Debugging Tools* (to install for multiple platforms). Alternatively download the ISO version of the SDK and extract the installers directly.

Use SymStore.exe (in `C:\Program Files\Debugging Tools for Windows` by default) to archive modules and symbols as follows:

```
SymStore.exe add /r /f r:\mybuild\bin /s r:\symbols /t "Product Name" /v
"1.2.3.4" /c "Comment (e.g. Daily Build)"
```

This processes `r:\mybuild\bin` and any subdirectories and adds all modules (i.e. EXE and DLL files) and symbols (i.e. PDB files) to `r:\symbols`. /t sets the product name, /v the version and /c an optional comment. This information is helpful if you're running out of space and need to delete older symbols from the store.

If you inspect the contents of the symbol store you'll notice that files are stored uniquely, for example the following PDB is indexed by a build-specific GUID:

```
R\Symbols\MapTest.pdb\1EA3C81991814289A5E95FF06E891E2420\MapTest.pdb
```

If you are likely to store a very large number of symbols add a file called `index2.txt` to the root of the symbol store (i.e. `R:\Symbols\index2.txt`). The contents of the file do not matter. This will create an extra layer of directories based on the first two characters of the filename:

```
R\Symbols\MA\MapTest.pdb\1EA3C81991814289A5E95FF06E891E2420\MapTest.pdb
```

At this point you can create a share for the symbol folder (i.e. `\\Server\Symbols`) and use this within your company for debugging. If a shared folder does not meet your needs then you can create a HTTP Symbol Server. Some companies, including Microsoft, make private symbols available through a public HTTP Symbol Server. You can use the same technology to privately share your public symbols.

---

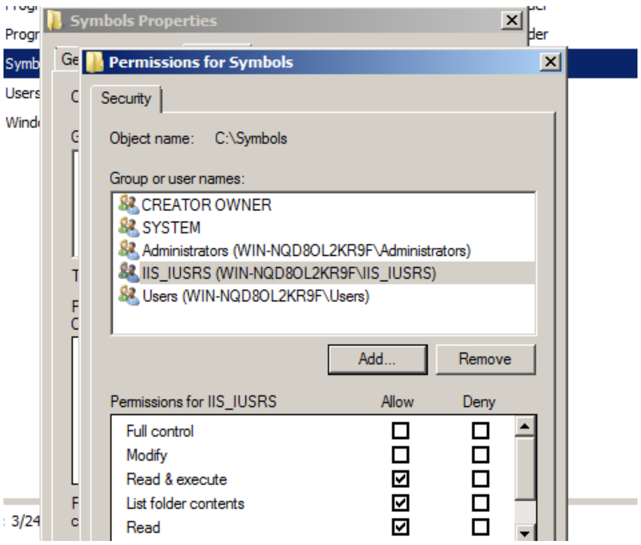
[1] http://msdn.microsoft.com/en-us/library/ff558825(v=VS.85).aspx (http://goo.gl/kIsKp), http://www.wintellect.com/CS/blogs/jrobbins/archive/2009/05/11/pdb-files-what-every-developer-must-know.aspx (http://goo.gl/tHrI)

[2] http://msdn.microsoft.com/en-us/windows/hardware/gg463016 (http://goo.gl/1FtjF)

HTTP Symbol Server

Creating a HTTP Symbol Server is as simple as sharing the symbol store folder as a virtual directory in Internet Information Services (IIS). The instructions below assume that you're using Windows Server 2008 R2.

1. If necessary install IIS. From *Server Manager* go to *Roles* and click *Add Roles*. Use the *Add Roles Wizard* to install the *Web Server (IIS)* Role. If you also plan to enable symbol proxy functionality (see below) make sure you select the *ISAPI Filter* and *ISAPI Extension* options during installation.
2. Right-click the symbol store folder and choose *Properties*. Go to the *Security* Tab and click *Edit*. Click *Add* and enter `IIS_IUSRS`. Check that the *Read & execute*, *List folder contents* and *Read permissions* are set to *Allow*.

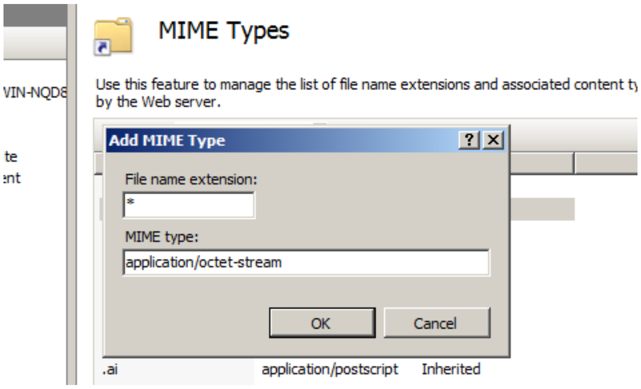


Object name: C:\Symbols

Group or user names:
- CREATOR OWNER
- SYSTEM
- Administrators (WIN-NQD8OL2KR9F\Administrators)
- IIS_IUSRS (WIN-NQD8OL2KR9F\IIS_IUSRS)
- Users (WIN-NQD8OL2KR9F\Users)

Permissions for IIS_IUSRS	Allow	Deny
Full control	☐	☐
Modify	☐	☐
Read & execute	☑	☐
List folder contents	☑	☐
Read	☑	☐

3. Run *Internet Information Services (IIS) Manager*. Expand the tree to the *Default Web Site*, right click it and choose *Add Virtual Directory*. Enter an alias (i.e. `Symbols`) and the path to the symbol store (i.e. `C:\Symbols`).
4. In the MIME Types settings for the virtual directory add a new extension `*` with the MIME Type `application/octet-stream`.



Use this feature to manage the list of file name extensions and associated content t by the Web server.

File name extension: *

MIME type: application/octet-stream

.ai application/postscript Inherited

5. Restart *Default Web Site*.

You should now have a symbol server running at `http://myserver/Symbols`. Test that you can successfully retrieve symbols from the server. If you plan to make the symbol server accessible outside your intranet, work with your network administrator to ensure that the server is secure and that an appropriate authentication mechanism is enabled.

Further documentation for setting up a HTTP Symbol Server can be found at:

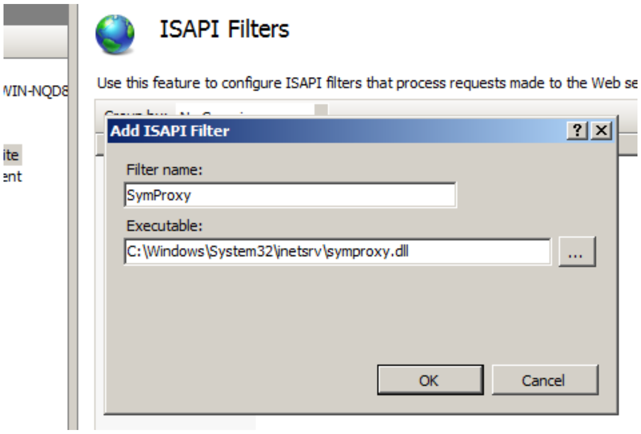
`C:\Program Files\Debugging Tools for Windows\symproxy\symhttp.doc`

Extra Credit 1: Symbol Proxy Server

Once your company has implemented a HTTP symbol server you can extend it to act as a proxy for other symbol servers, including Microsoft's public symbol server. This simplifies configuration as you only need to point debuggers at one location for symbols. It also speeds up debugging as the local symbol server cache is populated and can be very helpful when debugging on systems that cannot access the Internet.

The instructions below assume that you have configured a HTTP symbol server as described above and that you are using Windows Server 2008 R2.

1. Copy `symproxy.dll` and `symsrv.dll` from `C:\Program Files\Debugging Tools for Windows\symproxy\` to `C:\Windows\System32\inetsrv\`. Use files from the 32 or 64-bit install of the Debugging Tools for Windows as appropriate for your server.
2. Run `C:\Program Files\Debugging Tools for Windows\symproxy\symproxy.reg` to add default symbol proxy settings to your registry.
3. Run Internet Information Services (IIS) Manager and expand the tree to *Default Web Site*. Add `symproxy.dll` as an ISAPI filter called SymProxy.



Use this feature to configure ISAPI filters that process requests made to the Web se

Filter name: SymProxy

Executable: C:\Windows\System32\inetsrv\symproxy.dll

4. Restart *Default Web Site*.

5. If your server connects to the Internet without using a proxy server run `regedit.exe`, browse to `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Symbol Server Proxy` and set the `NoInternetProxy` value to 1.
6. Run `regedit.exe` and browse to `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Symbol Server Proxy\Web Directories\symbols`. The `SymbolPath` value is a semicolon separated lists of symbol server that the proxy should query for requested symbols that are not in the local symbol store. The default value is the Microsoft public symbol server (`http://msdl.microsoft.com/download/symbols`).
7. Using `regedit.exe` again browse to `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\eventlog\Application` and create a new subkey called `Symproxy`. Right-click the `Symproxy` key and add a new *Expandable String Value* called `EventMessageFile` and set this to `%SystemRoot%\System32\inetsrv\symproxy.dll`.

Your symbol server at `http://myserver/Symbols` should now serve symbols from the local symbol store, the Microsoft public symbol server and any additional symbol sources you configured in step 6.

Further documentation for setting up a HTTP Symbol Server Proxy can be found at:

```
C:\Program Files\Debugging Tools for Windows\symproxy\symhttp.doc
```

Extra Credit 2: Pre-populate Symbols

SymChk.exe can be used to pre-populate your local symbol cache. This tool is installed with the Debugging Tools for Windows. Run SymChk with no parameters for a full list of options. To load symbols for a crash dump use the /id switch:

```
SymChk.exe /id r:\dump.mdmp /s SRV*c:\symbols*http://symbols.mycompany.com/
```

This command checks all the loaded modules for `dump.mdmp` against the contents of `c:\symbols` and attempts to add any missing symbols from the symbol server at `http://symbols.mycompany.com/`. If your `_NT_SYMBOL_PATH` environment variable is set correctly you can omit the symbol server specification from the command line.

As well as a dump file SymChk can be used to populate symbols from a running process or from any EXE or DLL (including recursively finding symbols from a directory).

For operating system symbols you can laboriously install each OS and service pack that you support and then use SymChk to populate your local store. Alternatively, Microsoft maintains a list of symbol packages[3] that you can download and extract.

---

[3] http://msdn.microsoft.com/en-us/windows/hardware/gg463028 (http://goo.gl/kTN5m)

3. Source Server Setup

Source Server loads the correct revision of your source code when debugging a crash dump. To enable this functionality you need to modify your build to index source code at the same time as you add symbols to your symbol server.

The indexing process annotates your symbol files with the revision number of each related source code file together with the version control command used to pull the correct version of the file during debugging.

If you use CVS, Perforce, SourceSafe, Subversion or Team Foundation Server then you're in luck. Modules that support your source control system are included with the Debugging Tools for Windows. For TFS you can index source and symbols directly from the build configuration[4]. If your system is not supported you can implement your own source control provider module by following the instructions in the source server documentation:

```
C:\Program Files\Debugging Tools for Windows\srcsrv\srcsrv.doc
```

To index source code (assuming Perforce):

1. Install Perl[5] 5.6 or later.
2. Add `C:\Program Files\Debugging Tools for Windows\srcsrv\` to your path.
3. Make a copy of `C:\Program Files\Debugging Tools for Windows\srcsrv\srcsrv.ini` and edit it to identify your source control server.
4. Run "`ssindex.cmd /system=P4 /?`" to list source control specific options. For Perforce you can use the `/label=` switch to use a label instead of the current revision number for indexing.
5. Run `ssindex.cmd` to index source:

   ```
   ssindex.cmd /ini=R:\srvsrv.ini /source=R:\Build /symbols=R:\Build
   /system=P4 /Debug
   ```

6. Verify that indexing succeeded by running "`SrcTool MyProduct.pdb`". This command will dump the command to use to extract each indexed source code file.

Source code indexing varies by source control provider and so you will need to review the documentation and the command line help from `ssindex.cmd` to tune the process for your environment. The `/Debug` switch should always be used as ssindex.cmd can otherwise fail to index PDBs without providing any error or warning messages (for example if your Perforce workspace includes forward slashes indexing will fail, with debug output enabled you'll see a "*zero source files found*" message).

---

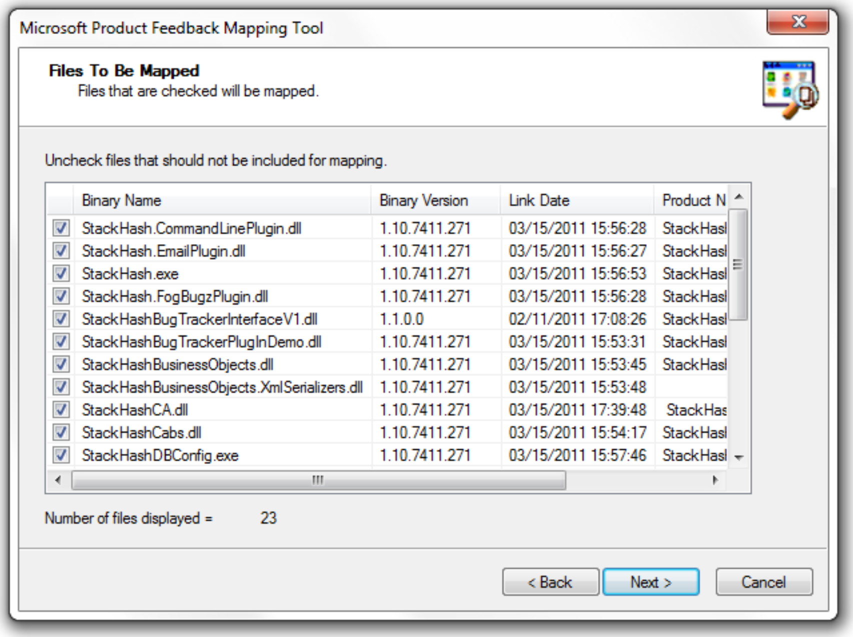
[4] http://www.edsquared.com/2011/02/12/Source+Server+And+Symbol+Server+Support+In+TFS+2010.aspx (http://goo.gl/0keX0)

[5] http://www.perl.org/

4. Product Mapping

Before WinQual can start providing you with data about your application you need to provide a product mapping file containing details of each EXE and DLL that makes up the product. When a crash or hang occurs, Windows Error Reporting on the end-user machine sends the faulting application and module names together with their versions to the WinQual site. If this information matches a mapped file then each such request will generate a 'hit' in your WinQual account. Without the application files having been mapped, the crash data would not be recorded.

Mapping files are generated using the Microsoft Product Feedback Mapping Tool[6]. This tool can be run as a wizard, or more usefully from the command line (`appmap.exe`).



Files To Be Mapped
Files that are checked will be mapped.

Uncheck files that should not be included for mapping.

	Binary Name	Binary Version	Link Date	Product N
☑	StackHash.CommandLinePlugin.dll	1.10.7411.271	03/15/2011 15:56:28	StackHasl
☑	StackHash.EmailPlugin.dll	1.10.7411.271	03/15/2011 15:56:27	StackHasl
☑	StackHash.exe	1.10.7411.271	03/15/2011 15:56:53	StackHasl
☑	StackHash.FogBugzPlugin.dll	1.10.7411.271	03/15/2011 15:56:28	StackHasl
☑	StackHashBugTrackerInterfaceV1.dll	1.1.0.0	02/11/2011 17:08:26	StackHasl
☑	StackHashBugTrackerPlugInDemo.dll	1.10.7411.271	03/15/2011 15:53:31	StackHasl
☑	StackHashBusinessObjects.dll	1.10.7411.271	03/15/2011 15:53:45	StackHasl
☑	StackHashBusinessObjects.XmlSerializers.dll	1.10.7411.271	03/15/2011 15:53:48	
☑	StackHashCA.dll	1.10.7411.271	03/15/2011 17:39:48	StackHas
☑	StackHashCabs.dll	1.10.7411.271	03/15/2011 15:54:17	StackHasl
☑	StackHashDBConfig.exe	1.10.7411.271	03/15/2011 15:57:46	StackHasl

Number of files displayed = 23

It is a good idea to make sure that your application components are uniquely named and versioned. If you have a DLL called Utils.dll with version 1.0.0.0 then you're liable to receive a large number of crashes that are unrelated to your DLL. You should also ensure that you don't map third party components.

The command line can easily be integrated with your build:

```
appmap.exe -s R:\Build -d R:\Mapping\1.2.3.4.xml -n MyProduct -v 1.2.3.4
```

Run "`appmap.exe /?`" for a full list of command line options.

---

[6] http://www.microsoft.com/downloads/en/details.aspx?FamilyId=4333E2A2-5EA6-4878-BBE5-60C3DBABC170&displaylang=en (http://goo.gl/TXzq5)

5. Debugger Configuration

After setting up a symbol server and source code indexing you need to configure your debuggers to take advantage of this information.

Debugging Tools for Windows

The Debugging Tools for Windows include WinDbg, KD, CDB and NTSD. For all of these tools you can set your symbol, source and image file search path via environment variables, command line options or commands within the debugger. WinDbg additionally allows these paths to be set through the `File...` menu.

Using the environment variables has the benefit that Visual Studio will pick up the symbol and image search paths automatically (from 2008).

Search Path	**Environment Variable**	**Command Line**	**Command**
Symbols	_NT_SYMBOL_PATH	-y	.sympath
Image	_NT_EXECUTABLE_IMAGE_PATH	-i	.exepath
Source	_NT_SOURCE_PATH	-srcpath	.srcpath

All three search paths consist of a semicolon separated list of directories. To access a symbol server use the format:

```
SRV*C:\Symbols*http://msdl.microsoft.com/download/symbols
```

SRV indicates that a symbol server is in use. C:\Symbols is a local cache folder. The debugger checks the cache first and then accesses the symbol server if the required file is not contained in the cache. A typical symbol or image search path might be:

```
C:\SymbolFolder;\\Server\Symbols;SRV*C:\Symbols*http://symbols.mycorp.com/sym
bols;SRV*C:\Symbols*http://msdl.microsoft.com/download/symbols
```

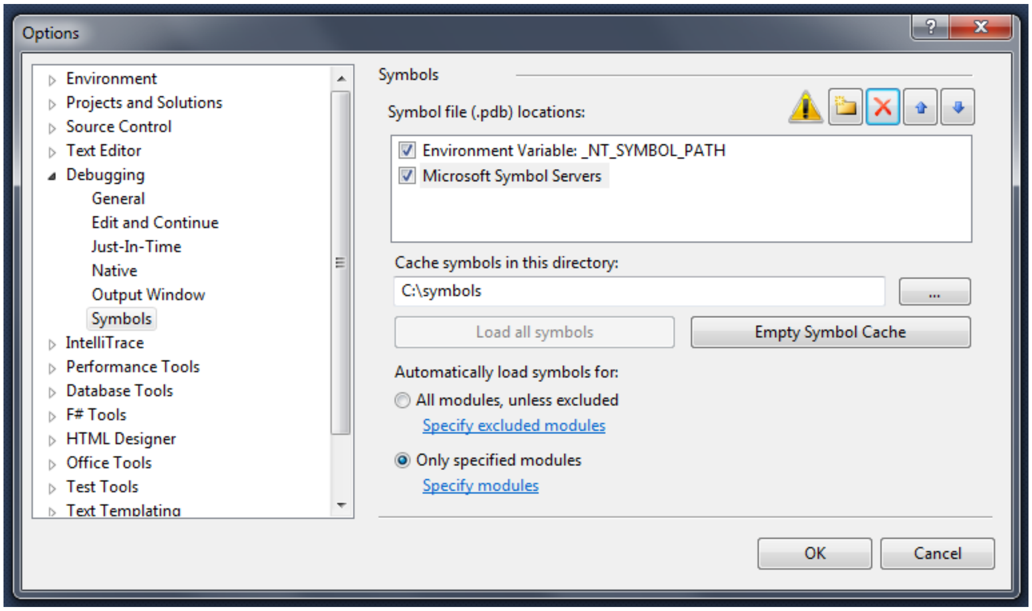
In this case the debugger will search the local folder, then the UNC folder, then the company symbol server and finally the Microsoft public symbol server.

Source server uses a slightly different format where only the cache needs to be specified:

```
SRV*C:\Symbols
```

Visual Studio

In Visual Studio 2010 set the symbol search path in *Tools -> Options -> Debugging -> Symbols*:



Symbol file (.pdb) locations:

- ☑ Environment Variable: _NT_SYMBOL_PATH
- ☑ Microsoft Symbol Servers

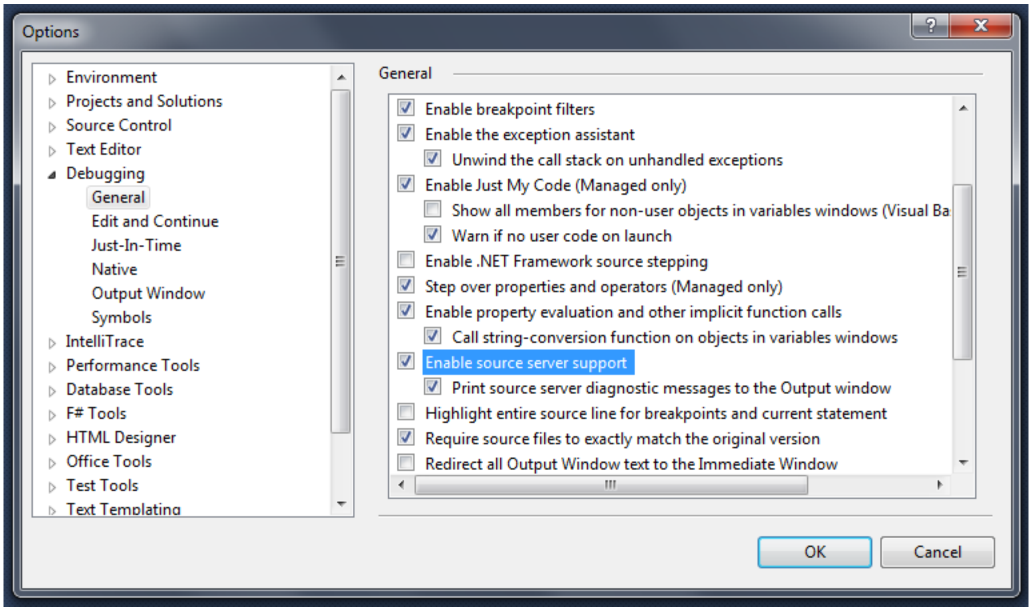
Cache symbols in this directory: C:\symbols

Automatically load symbols for:

- ○ All modules, unless excluded
- ● Only specified modules

If set the `_NT_SYMBOL_PATH` environment variable is pulled in automatically. You can add additional directories and symbol servers to search. Note that in Visual Studio you do not need to use the SRV format, just enter the web address of the symbol server.

Source server can be enabled from Tools -> Options -> Debugging -> General:



- ☑ Enable breakpoint filters
- ☑ Enable the exception assistant
  - ☑ Unwind the call stack on unhandled exceptions
- ☑ Enable Just My Code (Managed only)
  - ☐ Show all members for non-user objects in variables windows (Visual Ba
  - ☑ Warn if no user code on launch
- ☐ Enable .NET Framework source stepping
- ☑ Step over properties and operators (Managed only)
- ☑ Enable property evaluation and other implicit function calls
  - ☑ Call string-conversion function on objects in variables windows
- ☑ Enable source server support
  - ☑ Print source server diagnostic messages to the Output window
- ☐ Highlight entire source line for breakpoints and current statement
- ☑ Require source files to exactly match the original version
- ☐ Redirect all Output Window text to the Immediate Window

6. WinQual Account Setup

WinQual (Windows Quality Online Services) is the Microsoft portal for Windows Error Reporting (WER) data, Driver Distribution Center (DDC) and Windows Logo.

To create a WinQual account visit https://winqual.microsoft.com/Signup/ in Internet Explorer. Your WinQual account is associated with a Windows Live Id. If you don't already have a Live Id you'll need to create one at http://www.live.com/ before attempting to access WinQual.

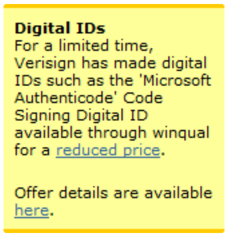
If your company has already started using WinQual you can pick your company from the list and request a user account. This will send an email to your company's WinQual administrator to request access. If your company has not registered for WinQual yet then you'll need to create a company account first. This requires a VeriSign code signing certificate, two legal agreements and some contact information.

Code Signing

The first stage of setting up a company account is to download a file called `Winqual.exe` and sign it with a VeriSign code signing or organizational certificate. The reason this is required is so that WinQual can trust that the person creating the account is a bona fide representative of the company in question.

Note that using WinQual does not require that your company uses VeriSign for code signing. You can use a different vendor, or not sign at all. The certificate is just required to authenticate your company to WinQual.

You can purchase the required certificate for $99 by following the link on the WinQual home page:



Digital IDs
For a limited time, Verisign has made digital IDs such as the 'Microsoft Authenticode' Code Signing Digital ID available through winqual for a reduced price.

Offer details are available here.

Once you have obtained the certificate, sign `Winqual.exe` using `signtool.exe` (a download link for `signtool.exe` is provided during the sign-up process).

If the certificate is in your certificate store:

```
signtool.exe sign /a /t http://timestamp.verisign.com/scripts/timstamp.dll
Winqual.exe
```

If the certificate has been exported to a PFX file:

```
signtool.exe sign /f certificate.pfx /p password /t
http://timestamp.verisign.com/scripts/timstamp.dll Winqual.exe
```

Legal Agreements

To access Windows Error Reporting data your company will need to electronically sign two agreements with Microsoft. The current versions are *Windows Error Reporting Agreement v1.3* and *Windows Error Reporting Terms Of Use V1.2*.

These agreements can be reviewed[7] before the company account is created.

Contact Information

During sign-up you will be required to provide contact and billing information for your company. Some services available through WinQual (i.e. Windows Logo certification) involve paying a fee to Microsoft. Windows Error Reporting data is available free of charge and so while you need to complete this information you will not be billed by Microsoft for accessing crash reports.

---

[7] https://winqual.microsoft.com/member/LAC/Default.aspx (http://goo.gl/FcJN0) Link must be opened in Internet Explorer

7. WinQual Usage

This chapter summarizes how to use WinQual to access Windows Error Reporting data. For more detailed documentation click the *Help* link at the top right of the WinQual site or download[8] the PDF version.

Note that WinQual takes some time to process data so crash reports will only be available from one to two weeks ago. When in the Software section of the site the page title shows you when the site data was last updated. Also note that WinQual only stores the past three months of data.

Terminology

Crash and hang reports in WinQual are organized by products, events, hits and cabs:

Product	A set of mapped files grouped under a product name and version.
Event	A mechanism to group crashes based on several parameters[9] (application name/version, module name/version, offset and exception code). Also referred to as a bucket. Note that an event is uniquely identified by the combination of the Event Id and the Event Type. Event Ids may be duplicated and may be negative. For example "-2124522262, Crash 32-bit" is a unique event.
Hit	An instance of an event, identified by date, operating system and locale information for computers experiencing the crash or hang. Also referred to as an "Event Info".
Cab	A compressed file containing more information about a crash or hang. A Cab typically contains a minidump (.mdmp) and metadata related to the system environment. If additional information has been requested, the Cab may contain a larger crash dump and files from the end user's system.

Adding Users

Once you have created a company account you will probably want to provide access to other users within your company. There is no way to invite a user from within WinQual. Direct users to the sign-up page (https://winqual.microsoft.com/Signup/, Internet Explorer only) where they can select your company and request an account. Click the *Member Services* link on WinQual to approve requests and assign site permissions to users.

Uploading and Managing Mapping Files

To receive crash reports for a product you need to upload an XML mapping file. For instructions on creating the file see Chapter 4 above. Once you have created the mapping file go to *Windows Error Reports -> Software -> Mappings -> Upload Mapping File* to upload it. You will start to see crash reports for the product a week or two after uploading the mapping file.

---

[8] http://winqual.microsoft.com/help/winqual_help.pdf (http://goo.gl/auVib)
[9] http://www.sigops.org/sosp/sosp09/papers/glerum-sosp09.pdf (http://goo.gl/Rqhg5)

You can view and delete existing product mappings from *Windows Error Reports -> Software -> Mappings -> Manage Product Mappings*. Click the icon under *File Mappings* to view and delete files mapped to each product. This is especially useful if you have inadvertently mapped a third party component and wish to stop receiving crash report data for it.

Software Home

Windows Error Reports -> Software -> Software Home

Software Home shows the top twenty events across all products for your company (Company Hotlist). This page also displays alerts if you have a failed file mapping, a denied response or a buffer overrun event that does not yet have a response.

Products

Windows Error Reports -> Software -> Product Rollups

The *Product Rollup* page lists each mapped product that currently has events in WinQual. Click the icon under *Eventlist* to list all events associated with a product. Click the icon under *Hotlist* to show the top twenty events for a product.

Event List

The *Event List* shows all events associated with a product. Company and product hotlists have similar functionality but show only the top twenty events.

Click the hyperlinked Event Id to open the *Event Details* page.

The Cabs column displays one of three icons depending on the current status:

- Cabs are currently not being collected for the event. Click the icon to toggle to collection mode.
- Event is in collection mode but no cabs have been collected yet.
- Cabs are available for the event. Click the icon to download cabs.

Further options for data collection are available on the *Event Details* page.

Responses will show a green tick icon if a response has been published for the event or affected module. Click the icon to preview the response. Responses can be created on the *Event Details* page.

Average Hits is the total number of hits for event divided by the number of days that have hit data. If this value is 1.00 then the event is being reported to Windows Error Reporting once a day.

Growth Percent compares the past two days of hits to hits from 2-3 weeks ago, scaled by the average number of hits for your company.

The remaining columns show the bucketing parameters used to identify the event.

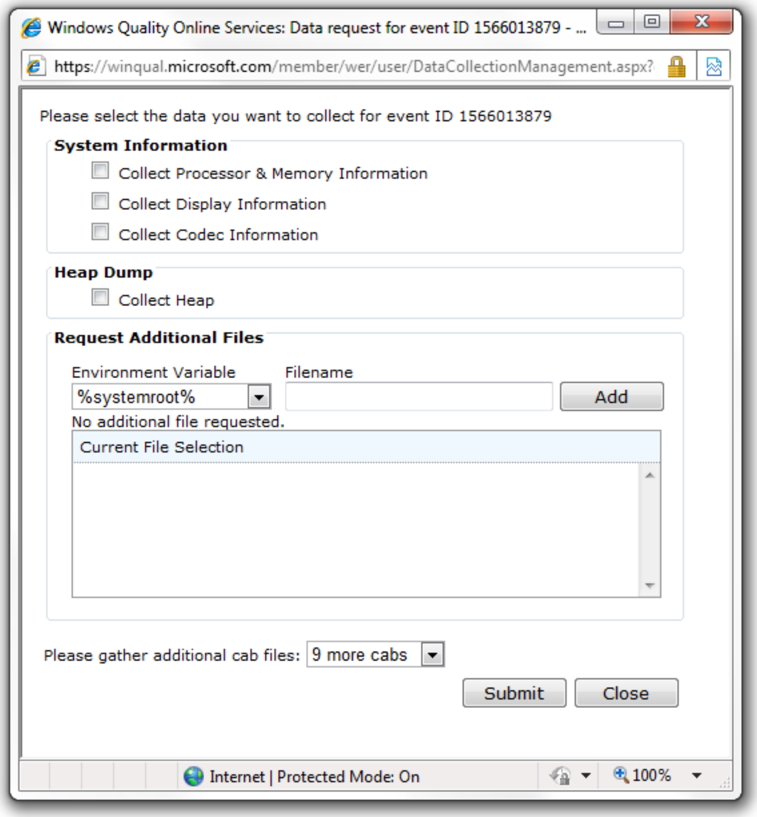
Event Details

The *Event Details* page allows you to create a response, manage data collection and view hit information for an event.

If the event has not yet been responded to you'll see a warning at the top of the page: *No Response Registered!*

Responses can be associated with a single event or with a module or application. See Responses below for more information.

To manage data collection click the icon next to *Data Request* ( ):



Windows Quality Online Services: Data request for event ID 1566013879 - ...

https://winqual.microsoft.com/member/wer/user/DataCollectionManagement.aspx?

Please select the data you want to collect for event ID 1566013879

System Information

☐ Collect Processor & Memory Information
☐ Collect Display Information
☐ Collect Codec Information

Heap Dump

☐ Collect Heap

Request Additional Files

Environment Variable: %systemroot%
Filename:

No additional file requested.

Current File Selection

Please gather additional cab files: 9 more cabs

Requesting additional files through WinQual can be difficult. A better approach is to use the `WerRegisterFile` API in your application to flag files for collection in the event that a crash report is submitted. In addition to providing greater precision this approach means that the requested files are added to all cabs rather than requiring you to make specific per-event requests in WinQual.

Finally on the Event Details page there is a chart showing hits over time and a breakout of hits by operating system and end-user language (locale).

Responses

Responses can be filed for a specific event or for an application or module from the *Event Details* page. Once a response is approved it will be displayed to end-users when a crash or hang is reported for the linked event, application, or module. On Windows Visa and later, past crashes are periodically checked to see if a response is available and if so the end-user is notified.

A response can be used to provide a specific fix, or to collect more information from end-users. Select an appropriate template and then fill in the required fields. Each response is manually reviewed[10] before being approved.

You can view the status of requested and published responses from *Windows Error Reports -> Software -> Responses -> Manage Responses*.

---

[10] http://blogs.msdn.com/b/wer/archive/2010/11/09/a-checklist-to-help-you-create-clear-and-accurate-responses.aspx (http://goo.gl/y32Xw)

8. Managed Debugging

Visual Studio 2010 supports debugging crash dumps for .NET 4.0. If your product uses an older version of the framework then you'll need to use the SOS (Son of Strike) debugger extension. It's possible to load this in the Visual Studio Immediate Window or from the Debugging Tools for Windows (i.e. WinDbg). Debugging crash dumps with SOS can be especially challenging because it depends on the exact version of .NET that was in use on the computer where the crash dump was created. General use of SOS is beyond the scope of this guide (see Resources below) but it is important to understand how to get the extension working with crash dumps.

When a managed dump file is loaded in WinDbg the SOS extension can be loaded as follows:

```
.loadby sos mscorwks
```

Once SOS is loaded the following command lists managed threads:

```
!threads
```

SOS depends on a file called `mscordacwks.dll` (data access DLLL) which is tightly bound to each build of the CLR. For example if your dump file used 2.0.50727.4206 but the machine you are debugging on has 2.0.50727.4927 installed then SOS commands will fail. The error message when the correct `mscordacwks.dll` cannot be loaded looks like this:

```
Failed to load data access DLL, 0x80004005
Verify that 1) you have a recent build of the debugger (6.2.14 or newer)
            2) the file mscordacwks.dll that matches your version of mscorwks.dll is
                in the version directory
            3) or, if you are debugging a dump file, verify that the file
                mscordacwks_<arch>_<arch>_<version>.dll is on your symbol path.
            4) you are debugging on the same architecture as the dump file.
                For example, an IA64 dump file must be debugged on an IA64
                machine.

You can also run the debugger command .cordll to control the debugger's
load of mscordacwks.dll.  .cordll -ve -u -l will do a verbose reload.
If that succeeds, the SOS command should work on retry.
```

Microsoft publish some versions of `mscordacwks.dll` on their public symbol server but not all of them and unfortunately there is no way to determine if a particular version is available.

You can determine the required framework version in WinDbg by running "`lmv m mscorwks`". If the process in the dump has loaded the CLR the version will be reported (i.e. `File version: 2.0.50727.4206`).

To successfully debug managed crash dumps you will need to collect .NET installs. Start with 2.0.50727.42 (.NET 2.0 RTM) and archive the `C:\Windows\Microsoft.NET\Framework\v2.0.50727` folder (`C:\Windows\Microsoft.NET\Framework64\v2.0.50727` for 64-bit). Install the next version and repeat the process until you have a library of .NET releases. You can either add the library to your image search path or load SOS directly from the framework folder that matches the crash dump.

The most comprehensive history of .NET 2.0 versions is at http://blogs.msdn.com/b/dougste/archive/2007/09/06/version-history-of-the-clr-2-0.aspx (http://goo.gl/0yZBs). This includes links to the hotfixes and security updates associated with most releases. You can use this list to patch a system up to a specific .NET release in order to archive that version of the framework directory.

9. Resources

Recommended blogs:

- WER Services: http://blogs.msdn.com/b/wer/
- John Robbins: http://www.wintellect.com/CS/blogs/jrobbins/default.aspx (http://goo.gl/gbINP)
- Tess Ferrandez: http://blogs.msdn.com/b/tess/
- Maoni Stephens: http://blogs.msdn.com/b/maoni/
- Debugging Toolbox: http://blogs.msdn.com/b/debuggingtoolbox/
- Analyze -v: http://analyze-v.com/

Debugging list on twitter: http://twitter.com/StackHash/debugging

Native Debugging

The following books are essential references for debugging native code:

- Advanced Windows Debugging (*Mario Hewardt, Daniel Pravat : Addison Wesley*)
- Windows Internals: Including Windows Server 2008 and Windows Vista, Fifth Edition (*Mark Russinovich, David A. Solomon and Alex Ionescu : Microsoft Press*)

Managed Debugging

The following books are essential references for debugging managed code:

- Debugging Microsoft .NET 2.0 Applications (*John Robbins : Microsoft Press*)
- Advanced .NET Debugging (*Mario Hewardt : Addison Wesley*)

Documentation for the SOS debugger extension is available on MSDN at http://msdn.microsoft.com/en-us/library/bb190764.aspx (http://goo.gl/wUvkE).

Tess Ferrandez at Microsoft has a set of managed debugging tutorials at http://blogs.msdn.com/b/tess/archive/2008/02/04/net-debugging-demos-information-and-setup-instructions.aspx (http://goo.gl/sA4lK).

About StackHash

StackHash is a tool that works with the WinQual API to help developers make the most of Windows Error Reporting data:

- New product mappings, events, hits and cabs are downloaded locally overnight. This means you can start debugging in seconds rather than spending time finding and downloading crash data from WinQual. It also means that data is stored longer than the 90 day limit on WinQual allowing you to meaningfully compare product quality between releases.
- WinDbg scripts are run automatically on downloaded dump files. You can view debug results in StackHash and decide which files merit further investigation.
- A plugin architecture supports adding WinQual data to your defect tracking system. Events can be added manually or automatically during synchronization.
- Advanced search and filtering allow you to quickly zero in on the crashes that matter most at the company, product team and individual developer level.
- Full client/server support means you can configure a company StackHash Server connected to your WinQual account and then provide each developer with convenient access through the StackHash Client.

Learn more at http://stackhash.codeplex.com/.