Post-Mortem Debugging Tips for GME

This is a short overview on how you can help finding bugs in GME. Some of these suggestions (see Quickstart) need very few changes to your work environment.

# Quickstart

The easiest step to understand what went wrong with GME after a crash is to make the source symbols available on your machine. If you have Visual Studio 2008 installed on your machines (preferably, with SP1 and Feature Pack installed), all you have to do is to set up the following environment variable:

**\_NT\_SYMBOL\_PATH=SRV\*C:\symcache\*\\atlantis.isis.vanderbilt.edu\Project\GME\symbols\*http://msdl.microsoft.com/download/symbols**

Note, that C:\symcache will be used to store symbol files locally. Modify this part of the environment string to match your taste.

If this environment variable is set, then selecting Visual Studio as a post-mortem debugging option after a GME crash will show you the stack trace using proper symbols (both from the Microsoft and GME binaries).

The debugging symbols for Udm and GReAT are available at \\129.59.129.214\symbols. Append **\*\\129.59.129.214\symbols** to \_NT\_SYMBOL\_PATH to use these symbols.

# Minidumps

The previous method can be used immediately after a GME crash on the same computer where the crash occurred. GME also can save minidumps of these crashes (and these minidumps can be sent right after the crash) – thanks Kevin for his work on cleaning this up.

Minidump debugging is similar to the previous (immediate) method. You should open the minidump file with Visual Studio but before starting the debug session (F5), open the project properties dialog box and set the **Command Arguments** (Configuration Properties / Debugging) as follows:

**MODPATH=symsrv\*symsrv.dll\*C:\symcache\*\\atlantis.isis.vanderbilt.edu\Project\GME\symbols\*http://msdl.microsoft.com/download/symbols**

See my previous comment on the C:\symcache part. This setting will ensure that you will get the proper binaries (not just the symbols files), no matter what is the current host environment and/or GME version is installed.

# GME source code

You can access the source code of GME at this location:

**https://svn.isis.vanderbilt.edu/GMESRC/**

After each release (external or internal) I create a snapshot of the sources in the “tags” folder. The SVN head is in the “trunk” folder. You can use your ISIS credentials to access the repo. Let me know, if it does not work.

# Tools needed to compile GME

These are the tools you need to have installed to compile GME from the sources:

* **Visual Studio 2008 Professional**: available from our MSDNAA web site
* **Visual Studio 2008 Service Pack 1**: available from <http://www.microsoft.com/downloads/details.aspx?FamilyId=FBEE1648-7106-44A7-9649-6D9F6D58056E&displaylang=en>

If you want to build the entire GME distribution (.msi installer), you need the additional tools:

* **ActivePython 2.6:** available from <http://www.activestate.com/activepython>
* **WiX 3.0.5419.0** or later: available from <http://wix.sourceforge.net/>

# GME build process

First, you need to set the **GME\_ROOT** environment variable on your system, pointing to the local directory where the root of your GME source tree was checked out (i.e.: the local directory which corresponds either to the ‘trunk’ repository folder or one of the release tag folders in the repository)

To compile the core GME application, you need to open the GMESRC/trunk/GME/GME.sln solution file and build the solution. The Gme project within this solution should be set as the “Startup Project” to run/debug GME.

To build the entire GME distribution, open a command window as Administrator and execute the build.bat batch file in GMESRC/trunk.

# Error reporting

If you find a bug or have a feature request, the preferred way is to submit a JIRA ticket. Please, use this channel if you can explain/show the issue without providing sensitive data (**our JIRA system is not ITAR restricted/compliant!**):

<http://escher.isis.vanderbilt.edu/JIRA>