CodeDynamics 2017.1 Release Notes

Updated: April 2017

These release notes contain a summary of new features and enhancements, late-breaking product issues, migration from earlier releases, and bug fixes.

PLEASE NOTE: The version of this document in the product distribution is a snapshot at the time the product distribution was created. Additional information may be added after that time because of issues found during distribution testing or after the product is released. To be sure you have the most up-to-date information, see the version of this document on the Rogue Wave web site:

http://www.roguewave.com/support/product-documentation/codedynamics.aspx

Additions and Updates

Replay Bookmarks

For ReplayEngine supported platforms, Replay Bookmarks allow you to easily mark a point during your programs execution history and easily jump back to that point time! Read more about how to utilize Replay Bookmarks in the Chapter 8 of the User Guide.

Early Access to Mixed Language Debugging with Python and C/C++

This release introduces mixed debugging of Python and C/C++ applications, enabling you to easily see a fully integrated call stack across the language barriers and to examine data passed between the layers. Read more about the Python and C/C++ debugging capabilities in Chapter 7 of the User Guide for more information.

Support added to modify program arguments after debug session is started

Support has been added to modify program arguments after a debug session has been launched. The new arguments and settings will be used the next time the debug session is restarted.

New License Model for ARM64 and PowerLE Platforms

The ARM64 and PowerLE platforms have a new file based FlexNet Embedded license model. Customers who have purchased a license for these platforms will receive a new license file. Instructions for installing the license are in the Installation Guide.

GNU DebugFission Split DWARF on Linux

Split DWARF is a new approach that dramatically shrinks the space required to store DWARF debug information, speeds-up application link times, reduces system memory and IO requirements, and speeds-up debugger startup. Split DWARF places the majority of the debug information for an "object file" (.o) in a separate "DWARF object file" (.dwo) that is *not* processed by the linker.
TotalView now supports the GNU DebugFission variant of Split DWARF, which is described in more detail at https://gcc.gnu.org/wiki/DebugFission. The DWARF 5 variant of Split DWARF is not yet supported. For more information, check out the Saving Time and Space with Split DWARF white paper.

Building your application for GNU DebugFission Split DWARF:

- **Compiling:**
  - Use the "**-gsplit-dwarf**" compiler option to generate ".dwo" (DWARF object) files containing the full DWARF debug information, and a ".o" (object) file containing the code, data, and skeleton DWARF debug information. Note: the DWARF debug information in the ".o" file points to the ".dwo" file, therefore the ".dwo" file must not be deleted to debug that module.
- **Linking:**
  - Use the "**-fuse-ld=gold**" compiler option to use the gold linker (**ld.gold**).
  - Use the "**-Wl,--gdb-index**" compiler option to pass the "**--gdb-index**" option to the gold linker. Note: The resulting executable or shared library image file will contain a "**.gdb_index**" section that the debugger can use for faster startup.

Distribution Tar Bundle Name Change for macOS

The distributed tar bundles for macOS now has "-64" appended to their names to more clearly identify the architecture.

Bug fixes and Improvements

There have been a significant number of bug fixes and improvements added to the 2017.1 release.

Deprecation Notices

None

Known Issues

Linux

Split-DWARF and .gdb_index Support, and Related Options

State variable **TV::dwarf_global_index** is a boolean flag that controls whether or not TotalView consider using the DWARF global index sections (.debug_pubname, .debug_pubtypes, .debug_typenames, etc.) in executable and shared library image files. It defaults to **true**. It may be useful to set this flag to **false** if you have an image file that has incomplete global index sections, and you want to force TotalView to skim the DWARF instead, which may cause TotalView to slow down when indexing symbol tables. Command option **-dwarf_global_index** sets the flag to **true**,and **-no_dwarf_global_index** set the flag to **false**.

State variable **TV::gdb_index** -is a boolean flag that controls whether or not TotalView consider using the .gdb_index section in executable and shared library image files. It defaults to **true**. It may be useful to set this to **false** if you have an image file that has an incomplete .gdb_index section and you want to force TotalView to skim the DWARF instead. Command option **-gdb_index** sets the flag to **true**,and **-no_gdb_index** set the flag to **false**.

ReplayEngine On-Demand Records Can Show Invalid Stack Trace

In some circumstances in which a ReplayEngine debugging session is driven to the beginning of recorded history, the debugger will display an invalid stack trace and stack frame. We have observed this when debugging a ReplayEngine recording file that was created during a live debugging session in which ReplayEngine was enabled on-demand.

To recover a valid stack, simply step or continue the session - that is, move forward in history. If the beginning of history is specifically of interest, it can be reached directly by opening a CLI window and issuing the command "dhistory -go_time 1".

We expect this issue to be fixed in the next release.

CUDA

CUDA 8.0 Debugger API Internal Error

Certain NVIDIA driver versions associated with CUDA 8.0 may provoke "CUDA Debugger API internal error" messages from TotalView or CUDA-GDB. Known driver versions that have this problem are r361, r367, and r375, however the problem may exist in other driver versions. The internal error typically involves debugging a multi-thread application, when multiple host threads in the process are launching CUDA kernels. NVIDIA has confirmed the driver error, and plans to release a fixed driver version. A release date is not yet available. If the problem occurs, TotalView may print a message and the program may appear to hang.

NVIDIA Pascal Unified Memory Debugger Internal Error

CUDA applications running on Pascal under the debugger may cause a debugger internal error (for example, a SEGV) when the application process exits. Known driver versions that have this problem are r361 and r375, however the problem may exist in other driver versions. The debugger internal error typically involves debugging a CUDA application that exits after using unified memory. NVIDIA has confirmed the driver error, and plans to release a fixed driver version. A release date is not yet available. If the problem occurs, TotalView will exit with an internal error.