CodeDynamics 2017.0 Release Notes

Updated: January 2017

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

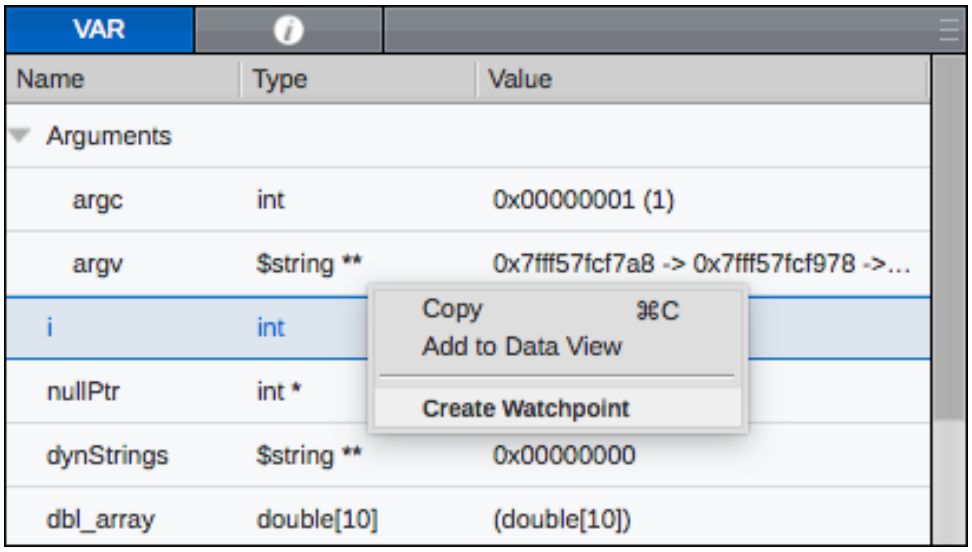
PLEASE NOTE: The version of this document in the product distribution is a snapshot at the time the product distribution was created. Additional information may be added after that time because of issues found during distribution testing or after the product is released. To be sure you have the most 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

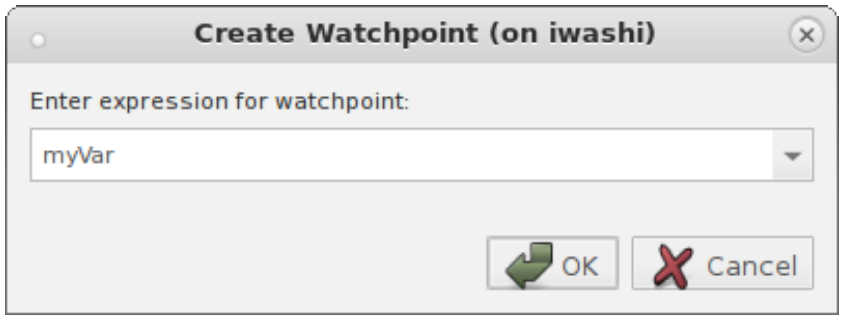
Additions and Updates

Create Watchpoints through the UI

[CodeDynamics|NextGen TotalView for HPC] now provides the ability to create watchpoints directly from the user interface. Watchpoints instruct the debugger to "watch" a segment of memory and stop execution of the program if the memory is changed. Watchpoints are easily created by simply right clicking on a variable in the VAR panel or in the Data View:



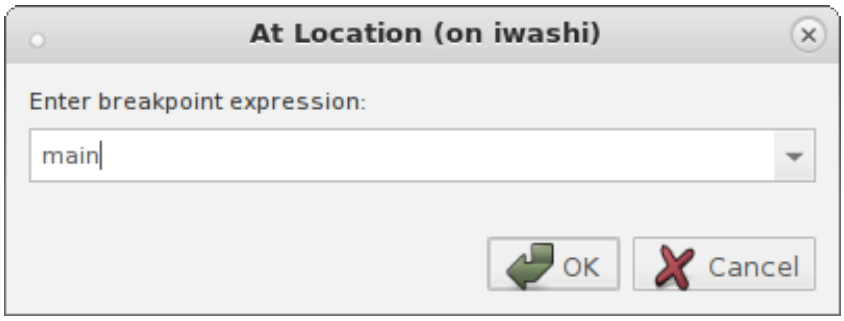
A free-form expression can also be entered through the Create Watchpoint dialog, which is accessed from the Action Points top-level menu:



Once a watchpoint has been created, it can be managed just like any other action point from the Action Point View.

Easily Create Breakpoints Throughout Your Program

Breakpoints can now be easily created throughout your program using the new At Location dialog. The dialog is accessible through the Action Points top-level menu and provides a convenient way to enter a valid breakpoint expression. Typical breakpoint expressions include a line number, a file and line number location (myFile.cxx#35), or a function signature (main). For more information about breakpoint expressions, see the dbreak command in the CodeDynamics Reference Guide.



ReplayEngine Performance Improvements

Performance for the ReplayEngine GoBack operation has been substantially improved. The magnitude of the improvement depends on the nature of the program being debugged, but it has

been measured at least a factor of 10 improvement when running a process backward until it hits some action point or the beginning of recorded Replay history.

NVIDIA® Tesla® P100 GPU with NVIDIA Pascal™ GPU architecture

The CodeDynamics 2017.0 release enables debugging support on NVIDIA's new Tesla P100 GPU which utilizes their new Pascal GPU architecture.

Bug fixes and improvements

There have been a significant number of bug fixes and improvements added to the 2017.0 release.

Platform Updates

CodeDynamics 2017.0 introduces support for the following platforms, compilers and parallel environments.

Platforms:

- Fedora 25

Compilers:

- Oracle Studio 12u4

Deprecation Notices

CLI *replay* command's –get_time and –go_time deprecated

The following two CLI arguments for the *replay* (which is an alias for *dhistory*) command have been deprecated

-get_time - display the current time

-go_time - put the process back to the specified virtual time

The arguments will still be available in this release but will be removed in a subsequent release. A warning will be displayed when the arguments are used. In place of these arguments, we recommend using the new CLI Replay bookmark facility instead, which provides much better accuracy in returning you to the exact point in Replay history.

Through the Replay bookmark facility, it is now possible to create a bookmark at any point in recorded history and then go back to that exact point at any time. The Replay bookmarking facility is accessed through the CLI's *replay* command through the following new arguments:

- -create_bookmark *[comment]*
  Create a replay bookmark at the current location. You can specify an optional 'comment' to this command and it will be stored with the bookmark for display purposes. A bookmark will be created with a unique numeric 'ID'.
- -delete_bookmark *ID*
  Delete the bookmark with the specified ID.
- -clear_bookmarks
  Delete all Replay bookmarks.
- -goto_bookmark *ID*
  Go to the bookmark with the specified ID. This will bring the focus process back in time to the place where the bookmark was first created.
- -show_bookmarks
  Display all Replay bookmarks. This command shows the bookmark ID along with information about what line number, pc and function the bookmark is on.

  Example:

```
dhistory -show_bookmarks
bookmark: 1:  pc: 0x<Address>, function: main, line: 59, comment:
Starting bookmark
bookmark: 2:  pc: 0x<Address>, function: main, line: 60, comment:
bookmark: 3:  pc: 0x<Address>, function: main, line: 69, comment:
```

Known Issues

The new user interface currently lacks the ability to change the target program's arguments from the user interface. Arguments that were specified on the command line are used for each subsequent restart during the debugging session. The ability to change program arguments through the UI will be added soon but in the meantime, the following workaround can be used:

- Run the program from the CLI using `drun`
  The program can be run from the command line through the Command Line View with the drun command, specifying new arguments after the drun command, e.g. `drun arg1 arg2`. For example:

```
drun 0 4 –silent
```

Licensing

Linux ARM64 uses an Expiring Licensing Model

The Linux ARM64 platform uses an expiring license model since the platform is not supported by the FlexNet license manager. CodeDynamics will cease to run six months from the release date. Subsequent releases will extend the expiration date or provide a FlexNet license managed solution.

Linux

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.