RogueWave SOFTWARE

Innovate with Confidence

CODEDYNAMICS CHANGE LOG

Versions 2015.09 through 2018.3

ACKNOWLEDGMENTS

Use of the Documentation and implementation of any of its processes or techniques are the sole responsibility of the client, and Rogue Wave Software, Inc., assumes no responsibility and will not be liable for any errors, omissions, damage, or loss that might result from any use or misuse of the Documentation

ROGUE WAVE SOFTWARE, INC., MAKES NO REPRESENTATION ABOUT THE SUITABILITY OF THE DOCUMENTATION. THE DOCUMENTATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. ROGUE WAVE SOFTWARE, INC., HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS WITH REGARD TO THE DOCUMENTATION, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT SHALL ROGUE WAVE SOFTWARE, INC., BE LIABLE, WHETHER IN CONTRACT, TORT, OR OTHERWISE, FOR ANY SPECIAL, CONSEQUENTIAL, INDIRECT, PUNITIVE, OR EXEMPLARY DAMAGES IN CONNECTION WITH THE USE OF THE DOCUMENTATION.

The Documentation is subject to change at any time without notice.

Rogue Wave Software, Inc.

Product Information:	(303) 473-9118 (800) 487-3217
Fax:	(303) 473-9137
Web:	http://www.roguewave.com

Contents

Chapter 1 C

odeDynamics Change Log, Versions 2015.09 to 2018.3

This document lists updates made to the CodeDynamics product for versions 2015.09 through 2018.3. For specifics about the current release, please see the documentation provided with your CodeDynamics installation or the CodeDynamics Online Help and Documentation on the Rogue Wave website.

CodeDynamics 2018.3

Barrier Point Support

The ability to create Barrier Points to synchronize threads and processes has been added.

Set Breakpoint

You can now use the Set Breakpoint menu item on a selected source line number to create a breakpoint.

CUDA Debugging Improvements

CodeDynamics' 2018.3 enhancements include a new GPU navigation bar that allows for easy navigation between the Logical or Physical coordinates of the GPU, performance improvements when displaying breakpoints for GPU code, and other stability improvements.

CUDA 9.2 and 10.0 Support

CodeDynamics 2018.3 provides official support of CUDA 9.2 and CUDA 10.0 as well as CUDA 8.0 and 9.0-9.2.

Multi-GPU Debugging Improvements

Through collaboration with customers using architectures with multi-GPU enabled compute nodes, CodeDynamics 2018.3 improves the capabilities, reliability, and performance of debugging on advanced multi-GPU capable applications.

Bug Fixes and Improvements

Numerous bug fixes and minor improvements have been made to CodeDynamics 2018.3. Including improving the performance of UI updates when stepping applications and displaying large source files with many breakpoints.

CodeDynamics 2018.2

Set PC

You can now set the PC to a new line location. To change the PC, select a valid line in the source code, then choose the **Thread | Set PC** menu item or simply hit the "**p**" key.

Multi-Process Debugging Improvements

Various aspects of the UI now use the share group of the process in focus for operations and displayed data. For example, only source files and action points related to the current share group in focus are shown. When changing focus to another process in a different share group, the Source view and Action Point view update to show the related files and action points.

Display std::string Value Without Diving

The UI now displays the resulting string value of a `std::string` without having to dive on the string and view it in the Data View. The string is displayed in tooltips, in the VAR pane and other locations where you would expect the string value to be displayed.

Data View

A number of enhancements have been added to better display changing data and data collections during program execution.

Action Points

Action points now accurately display their state as dynamic code from CUDA kernels or shared libraries are loaded.

CUDA Debugging Model and Unified Display Improvements

This release improves on the ability to easily set action points within CUDA applications and applications that dynamically load shared libraries with **dlopen**. In either case, until the CUDA or shared library code is loaded, the information required for setting a breakpoint is not available to the debugger, so now, CodeDynamics allows setting a breakpoint on any line in the Source view, whether or not it can identify executable code for that line. The breakpoint becomes either a pending breakpoint or a sliding breakpoint until the CUDA or shared library code is loaded at runtime.

Manage Single-Stepper Skip Rules

The ability to define single-stepper “skip” rules that modify the way source-level single stepping works has been added. These rules identify functions that require no debugging. Skip rules can be defined to skip over a function or through a function.

QString Type Transformation

Instances of type QString in Qt4 and Qt5 applications are now automatically transformed so users no longer need to locate and manipulate the underlying character data to a human-friendly format.

Bug Fixes and Improvements

Numerous bug fixes and minor improvements have been made.

CodeDynamics 2018.1

New exec and fork handling controls

New with 2018.1, CodeDynamics allows you to control how `fork()`, `vfork()`, `clone()` (when used without the CLONE_VM flag), and `execve()` system calls are handled by the debugger. Using the new command line options or CLI state variables, users can now define if the debugger should stop the process, continue the process, or ask whether or not to stop the process when the fork'd or exec'd process is acquired in the CodeDynamics debugging session. For example, to configure CodeDynamics such that when **bash** calls exec, the process automatically continues with no questions asked, but for other processes CodeDynamics does ask to continue, use the following dset CLI command or CodeDynamics command option:

```
dset TV::exec_handling {{{^bash$} go} {. ask}} totalview -exec_handling '{{^bash$} go} {. ask}'
```

Above, the `<regexp>` is wrapped in an extra set of curly braces to make sure that Tcl did not process the "$" as a variable reference. Setting up **fork_handling** rules work in a similar manner.

See the **TV::exec_handling** and **TV::fork_handling** command line option and state variable documentation in the TotalView for HPC Reference Guide for more information.

Multiple Data Views

The Data View is used to explore variables and debug data in your program. To better manage the display of your data, multiple Data Views can now be created by clicking the **Duplicate View** icon in the toolbar of the Data View. This creates a new Data View and clones any selected items in the original Data View into the new one.

Managing Action Points from a New Source View Context Menu

To find out more information about an expression in the Data View, a Drawer was added to the view with a new Information tab. The Information tab displays the expression, scope, thread, type, address, language, and other information for the selected item in the Data View. Adjust the size of the Drawer by clicking on the grab bar and moving it up or down. Double clicking on the grab bar snaps it open or closed.

Bug Fixes and Improvements

Numerous bug fixes and minor improvements have been made.

CodeDynamics 2018

Python Debugging Support for ctypes

Python debugging now supports filtering of ctypes "glue" frames that tie together function calls between Python and C/C++. This allows developers to see a clean stack trace between the two languages as they would expect, without the unnecessary noise of the layers required to shepherd data and make the calls between the languages.

Managing Action Points from a New Source View Context Menu

The Source view now supports the ability to enable, disable, delete and view properties of action points through a context menu accessed by right-clicking on the action point line number. This supports more streamlined debugging sessions.

Easier Tooltip Viewing

CodeDynamics has improved how it displays very long string within tooltips.

Bug Fixes and Improvements

Numerous bug fixes and minor improvements have been made.

CodeDynamics 2017.3

Improved Inline and Optimized Code Debugging

Significant improvements were made to the way CodeDynamics debugs inline functions. CodeDynamics is now able to step the debugging session through the functions and show the local variables for each inline function call.

Evaluation Points Performance

CodeDynamics now evaluates most evaluation point expressions within the CodeDynamics server, providing faster performance. Some expressions may still need to be evaluated on the front-end portion of the debugger, depending on the complexity of the expression or variable data being accessed.

Python Debugging

Removed the need for a debug build of the Python interpreter for Python debugging. You may use the standard Python interpreter that was distributed with your operating system.

Watchpoint Expressions

Added support for watchpoint expressions, which allow you to define a short expression to run when the watchpoint triggers. Use the expression to check the new value of a variable, display information, or execute some other type of custom logic.

Improvements to Lookup View Search Algorithm

Enhanced and refined the search algorithm used by the Lookup View in order to remove duplicates and better prioritize the found results.

Modifying Properties on Action Points

Added support for editing properties on each Action Point type, including breakpoints, watchpoints, and evaluation points. Use properties to make minor changes to an existing action point without having to delete it and recreate it.

Incremental Display of Data in the Data View

Performance and scalability improvements have been made to the Data View allowing for the incremental display of very large structures.

CUDA 9

CodeDynamics has been validated against the latest release of the CUDA SDK, CUDA 9.

Platform Updates

CodeDynamics 2017.3 introduces support for the Fedora 26 platform and GCC 7.1 compiler, as well as Mac OS X High Sierra.

Bug Fixes and Improvements

Numerous bug fixes and minor improvements have been made.

CodeDynamics 2017.2

Evaluation Points Support Added

CodeDynamics now allows you to create and modify evaluation points, snippets of code that run when a location is hit. Evaluation points are an excellent resource to add conditional logic for stopping execution of your program or trying out new execution logic without modifying your program.

Official Support for Mixed Language Debugging with Python and C/C++

Mixed language debugging of Python and C/C++ applications, enables you to easily see a fully integrated call stack across the language barriers and to examine data passed between the layers. Currently supported platforms:

- Python 2.7 on Linux x86 64-bit, Linux PowerLE, and Linux ARM64

Data View Improvements for Deep Structures

The Data View better handles structures with many pointers to other structures that create deep trees of information. The UI now clips the displayed tree to support better performance while still allowing full display via right-clicking on a data element and selecting Dive.

Replay Reverse Debugging Added to tvscript

tvscript enables unattended debugging functionality that is often useful in test, continuous integration, and batch environments where interactive debugging is not always feasible. With this release, users can now enable reverse debugging through **tvscript** and use an event-driven approach to generate ReplayEngine recording files for later analysis. A common use is to enable reverse debugging as part of an overnight test run and if a test failure occurs, to generate a ReplayEngine recording file and attach it to a bug report for later analysis.

.gdb_index Section Support

CodeDynamics now supports processing **.gdb_index** sections contained within executable and shared library files. Compiling with DebugFission and linking with the gold linker to produce a **.gdb_index** can greatly reduce link time, disk usage, and debugger start-up time for large applications.

dwz Optimized DWARF Support

CodeDynamics supports debugging ELF shared libraries and ELF executables that are processed with the dwz tool, a tool for optimizing and removing duplicate debug symbols. For more information about dwz, see the dwz (1) Linux man page.

Platform Updates

CodeDynamics 2017.2 introduces support for the ARM64 platform and Absoft 17 compiler.

Bug Fixes & Performance Improvements

A significant number of bug fixes and improvements have been added to the 2017.2 release.

CodeDynamics 2017.1

Replay Bookmarks

For ReplayEngine supported platforms, Replay Bookmarks allow you to easily mark a point during your program's execution history and then jump back to that point in time.

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.

Modifying Arguments in an Open Session

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.

Bug Fixes & Performance Improvements

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

Distribution Tar Bundle Name Change for Mac OS

The names of the Mac OS distributed tar bundles now append "-64" to more clearly identify the architecture.

Currently Supported Platforms

Linux x86 64-bit, Linux PowerLE, Linux ARM64 and Apple's macOS/Mac OS X

CodeDynamics 2017.0

Create Watchpoints Through the UI

This release supports creating watchpoints directly from the user interface. Watchpoints instruct the debugger to "watch" a segment of memory and to stop program execution if that memory changes.

Create Breakpoints Anywhere in Your Program

Easily create breakpoints throughout your program using the new **Create Breakpoint At Location** dialog accessible through the Action Points top-level menu. Enter a valid breakpoint expression such as a line number (35), a file and line number location (`myFile.cxx#35`), or a function signature (`main` or `myClass::myFunction`).

Process and Thread State Now Identified by Color-Coded Icons

The Process and Threads view displays grouped threads and processes using common properties so you can quickly see the state of all the processes and threads. This release adds color-coded icons for easy identification of any process or thread state.

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

CodeDynamics 2016.07

Early Access Support for Linux ARM64

This release adds Early Access support for Linux ARM64, with support for most of CodeDynamic's standard functionality.

CodeDynamics has been tested on the Cavium ARM64 processor running Ubuntu 14.04 and on NVIDIA Jetson TX1 running Ubuntu 16.04.

The following are known limitations and issues:

- Hardware data watchpoints are not supported on NVIDIA Jetson TX1. Setting a watchpoint crashes the Linux kernel.
- TotalView cannot unwind stack frames that have no debug information.
- Unsupported in this release: Debugging user level threads, asynchronous thread control, and reverse debugging with ReplayEngine.

Fortran Support

Fortran debugging is now supported, with several enhancements to handle Fortran data types in the Data View, including support for modules and common blocks. Please submit any feature requests for additional Fortran functionality to support@roguewave.com.

CUDA 8 Support

CodeDynamics has been validated against the official CUDA 8 release.

CodeDynamics 2016.06

Source Search Path Management

If CodeDynamics cannot find the source for your program, you can now adjust the source search path directly through the Search Path Preferences panel.

CUDA 8 Support

CodeDynamics has been tested against the latest CUDA 8 release candidate and works as expected for CUDA debugging. When the final version CUDA 8 is released, CodeDynamics CUDA 8 support will be revalidated, but no changes are anticipated.

CodeDynamics License Software Update

All versions prior to 11.13.1.2 of the Flexera Software FlexNet Publisher, the licensing software used by CodeDynamics, contain a buffer overflow vulnerability that may be leveraged to gain code execution.

CodeDynamics has been updated to use the latest version and so contains an updated vendor daemon. You must update your license server installation to at least 11.13.1.2 to use this version of CodeDynamics in order to eliminate the security vulnerability. See the *CodeDynamics Installation Guide* for more information about setting up the license server. The updated licensing software is included in the distribution.

CodeDynamics 2016.05

Support for Rogue Wave Replicator

CodeDynamics 2016.05 adds support for loading and debugging application runtime sessions recorded with Rogue Wave's new Replicator product. Replicator records a Linux program while it's executing to produce a single Replay recording file containing a detailed recording of that program's execution history. You can then provide this Replay file to CodeDynamics and use all the power of the CodeDynamics reverse debugging technology to re-run and move back and forth in the program's execution history to debug and understand how the program ran.

The key feature of Replicator is that it can record without involving CodeDynamics. This independent execution makes it easy to debug a customer's code, since the Replay recording file is an exact replica of the issues that created the crash or bug. It can also be an important tool for testing in-house applications in which the testing team provides the engineers a recording of the problem that can be played back.

CodeDynamics 2016.04

CodeDynamics named user license

The CodeDynamics license now requires the license administrator to define a list of named users. The named users are defined as part of setting up the FlexLM license manager. The list of users can be changed at any time.

Support for evaluation and barrier points

While barrier and evaluation points must be initially created from the command line, once created they are displayed in the new interface Source and Action Points views. They can then be enabled, disabled, deleted, and suppressed like breakpoints.

Calling functions when running ReplayEngine recorded execution history

CodeDynamics now supports the calling of functions through the CodeDynamics expression system when the ReplayEngine reverse debugging engine is activated. This allows for calls to functions such as `printf` or your own functions while evaluating expressions in the debugger during playback mode. See the Reverse Debugging with ReplayEngine document for more information on how memory-related side-effects are handled during the evaluation of expressions.

Improved support for CUDA debugging

The interface correctly displays CUDA source code and threads, and supports the setting of breakpoints in CUDA code.

Improved source search algorithms

The locating of source files through the Source view and the Find Files and Functions view uses improved algorithms. The interface also responds to changes to the -EXECUTABLE_SEARCH_PATH variable. See the TotalView for HPC Reference Guide for information on setting this variable through the command line.

CodeDynamics 2016.01

Source Code Search “Wrap Search” Toggle

You can now enable or disable search wrapping beyond the end or beginning of the source using a toggle.

Action Point View Highlights the Hit Breakpoint

The Action Point View now highlights the breakpoint hit for a clear reference of which breakpoint caused your program to halt.

CodeDynamics 2015.11

C++ 11 Support

CodeDynamics supports C++11 features for the GNU compiler, including support for lambdas, transformations for smart pointers, auto types, R-Value references, range-based loops, strongly-typed enums, initializer lists, user defined literals, and transformations for many of the containers such as array, forward_list, tuple and others.

Source Code Search

In addition to function and file searches, you can now search for text in your source files using a new, integrated search panel.

Save and Load Action Points

Save your action points and then either reload or suppress them when you start another debugging session.

Use of Scope for Data View Expressions

The Data View now evaluates expressions using the original scope in which they were created, allowing you to view values of the same name used in different scopes.

CodeDynamics 2015.09

Introducing CodeDynamics

CodeDynamics' first release introduces a new UI for TotalView, representing a complete redesign of the interface. The new UI provides a single window that allows you to control multiple individual threads and processes covering all aspects of program execution and data, including viewing the call stack, running and stepping through your code, and setting breakpoints.

The entire power of the TotalView debugging engine is behind the new UI, but the interface does not yet incorporate all functionality of the engine. If not yet supported in the UI, users can still access all of TotalView's features through the CLI, or command line interface.

Each additional release of CodeDynamics will add further functionality.

Limitations in the UI

The current UI does not yet support:

- Parallel environments such as MPI and OpenMP, and HPC environments
- Remote debugging
- Memory debugging with MemoryScape
- CUDA debugging
- Xeon Phi debugging

Users who need these features can launch TotalView for HPC by simply running the `totalview` executable:

```
totalview
```