CodeDynamics 2019.2 Release Notes

Updated: July 2019

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:

https://docs.roguewave.com/en/codedynamics/current/

Additions and Updates

Reverse Connect

The organization of modern HPC systems often makes it difficult to deploy tools such as CodeDynamics. For example, the compute nodes in a cluster may not have access to any X libraries or X forwarding, so launching a GUI on a compute node is not possible.

Using the new Reverse Connect feature, you can easily establish a connection where the CodeDynamics UI is running on a front-end node and debugging a job executing on compute nodes.

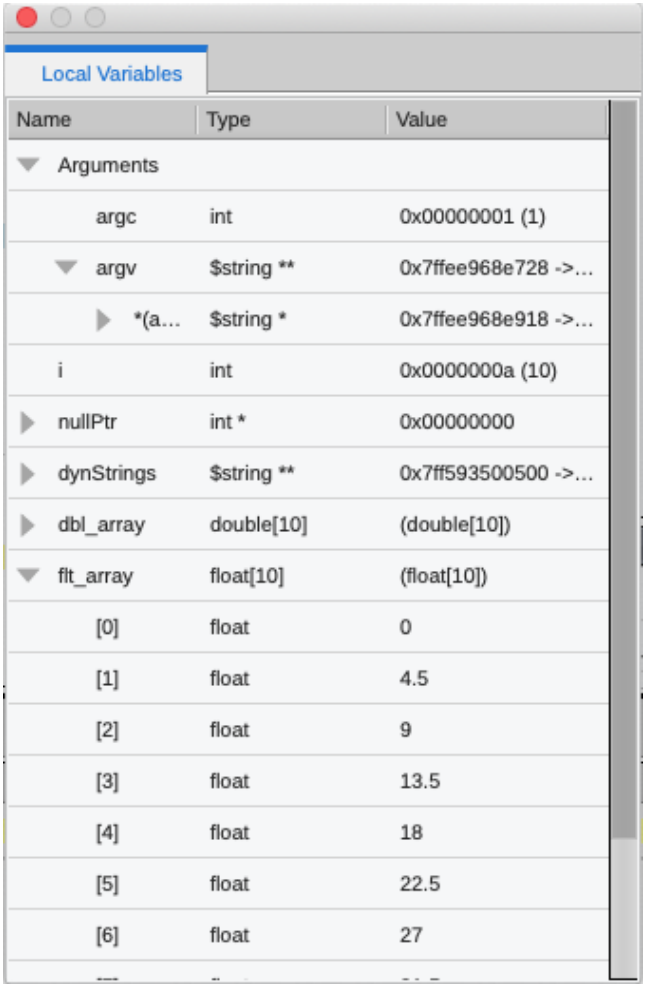
The basic process is to embed the **tvconnect** command in a batch script; when the batch job runs, the **tvconnect** process connects with the CodeDynamics client to start the debugger server process on the batch node. The CodeDynamics client would typically run on a front-end node, where the application is built and batch jobs are submitted. For more information, see Chapter 11, "Reverse Connections," in the *CodeDynamics User Guide.*

Debugging Through Starter Shell Scripts

Often developers will wrap the startup sequence of their application with a shell script in order to set up the run time environment for the application and then run it. The scripts would often need a special "debug" switch in order to start the application under CodeDynamics. In the 2019.2 release, CodeDynamics now recognizes when given a shell script as the debug target and will chain through the startup sequence of the shell script, including recursing through multiple startup scripts if needed, to find the resulting target that should be debugged.

Local Variable View

In previous releases, the Local Variable tab was part of the Call Stack View and locked in a set position. For the CodeDynamics 2019.2 release, the display of local variables is a separate view, which enables users to move the view to new locations or even undock it completely. In addition, the local variable view functionality has been enhanced and allows structures to be expanded in place.



Python pybind/pybind11 Debugging Support

CodeDynamics' mixed language Python and C++ debugging support now properly identifies pybind/pybind11 callstack intermediate frames and will remove them, so that a clean call sequence between Python and C++ is presented.

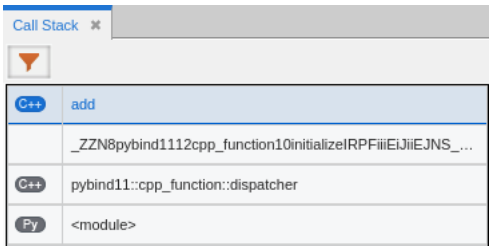


Figure 1 - pybind intermediate frames

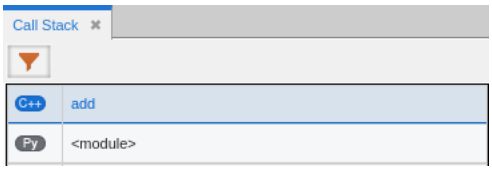


Figure 2 - Cleaned up pybind frames

Automatic Process/Thread Breakpoint Focus Preference Change

In past releases, the "Automatically focus on threads/processes at breakpoint" preference defaulted to false. This could create confusion during a debugging session when a thread or process not in focus hits a breakpoint and stops. In this case, the process/thread is not automatically brought into focus, requiring that users manually examine the set of processes and threads and change the focus.

For the 2019.2 release, the preference defaults to true so that any process or thread that hits a breakpoint will be brought into focus in your debugging session. If you prefer the old behavior, simply change the preference in the Action Points tab in the Preferences dialog or add "dset TV::GUI::pop_at_breakpoint false" to your .tvdrc file.

New Input/Output View available

In the CodeDynamics 2019.1 release, the ability to debug applications requiring standard input was added, allowing users to enter input into the terminal in which CodeDynamics was launched.

For the 2019.2 release, this input can be entered directly in the user interface instead of through the terminal. To activate this workflow, turn on the "Input Output view support" in the RW Labs tab on the Preferences dialog and restart the debugger. A new Input/Output view will be available that allows you to enter input to your running application.

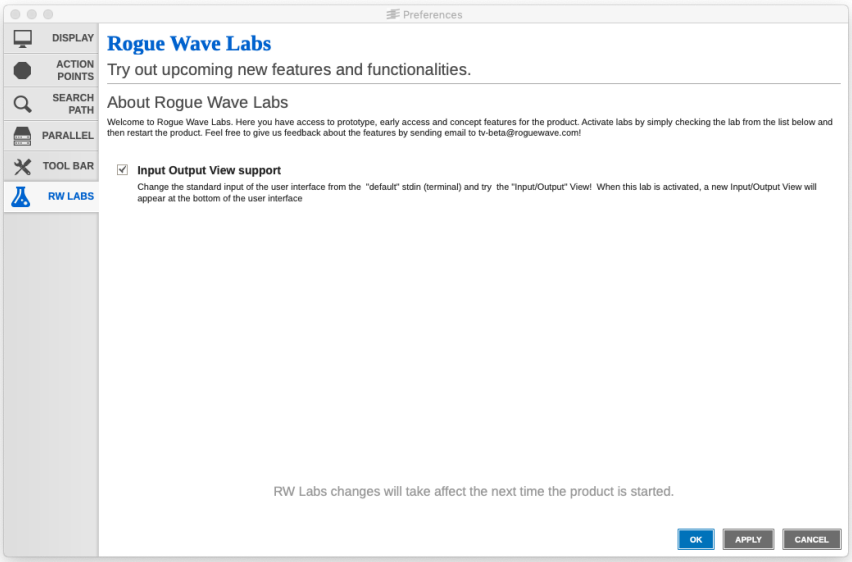


Figure 3 - Activate the new Input/Output View under RW Labs

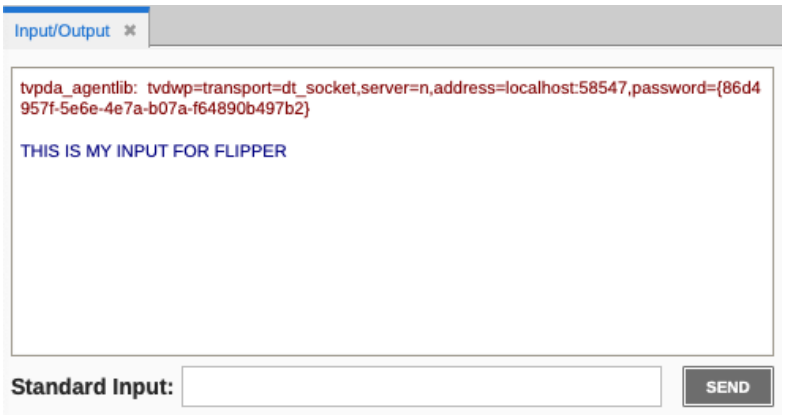


Figure 4 - Enter input to your application in the new Input/Output view

Change UI Font Size

The ability to change the overall font size for the user interface has been added to the CodeDynamics 2019.2 release. To change the font size, bring up the Preferences dialog and select the Display tab. Use the Font Size slider to change the font size to small, medium, large or extra-large. The default font size is medium. The next time you start CodeDynamics the UI will use the specified font size.

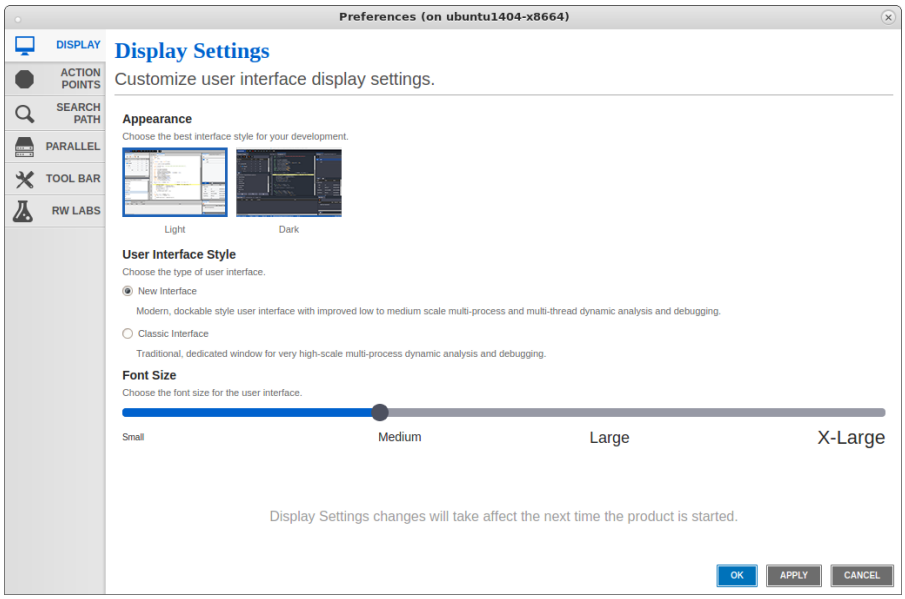


Figure 5 - Font Size slider on Preferences Display tab

Stability Improvements and Platform Updates

For the 2019.2 release, CodeDynamics has also added operating system support for Fedora 30 and GCC 9 compiler support. Numerous bug fixes and performance improvements have been made to the new UI including properly focusing on main for MPI, execve and Fortran applications.

Platform Updates

CodeDynamics 2019.2 introduces support for the following platforms:

OS:

- Fedora 30

Compiler:

- GCC 9

Bug Fixes for 2019.2

TODO

Deprecation Notices

Intel Xeon Phi Offload Coprocessors

Starting with release 2020, CodeDynamics will no longer support the Intel Xeon Phi offload coprocessors.

Known Issues

Python Debugging

Anaconda

CodeDynamics supports debugging of the python interpreter in release 4 of Anaconda but is not working with the recent release of Anaconda 5. Debugging python with GDB also does not work. Something in the way they build the python interpreter has broken the ability to debug python.

Ubuntu 16.04

When debugging python on Ubuntu 16.04 CodeDynamics is not detecting the python interpreter automatically and does not turn on python filtering. Filtering can be turned on by clicking the "filter" icon in the toolbar of the Call Stack view.

macOS

Mojave not yet supported

Attempts to run CodeDynamics on the new macOS Mojave result in a message indicating the Mach system call task_for_pid() is not working correctly. This is a permissions issue which is being investigated. Invoking CodeDynamics with sudo privilege could be a possible workaround.

Physical console access needed when running CodeDynamics on macOS High Sierra

Due to new security changes in macOS High Sierra, CodeDynamics will only run from the console and cannot be run through a remote desktop technology such as VNC. We are still assessing what changes need to be made to CodeDynamics so that it will run remotely on macOS High Sierra systems.

Linux

Intel 17 and 18 compilers not generating debug information for a declared integer

The Intel 17 and 18 version compilers are not generating proper debug information for declared integers in Fortran applications. As a result, CodeDynamics is not able to proper evaluate the variable expression and display the variable values. This is a compiler bug but a workaround is available by simply adding the `-debug extended` compilation flag option which adds symbols for local scalar variables and parameters.