Last Reviewed: 8/99

CodeCenter ChangeLog:

CodeCenter Versions 4.1.0, 4.1.1, 4.2.0 and 4.2.1

Table of Contents

This document provides details on all new features added to the CodeCenter product since version 4.1.0. This includes all new features for versions 4.1.0, 4.1.1, 4.2.0, and 4.2.1. It also describes the changes made between versions 4.0 and 4.1, specifics on the CodeCenter Documentation and product limitations.

> **NOTE:** All features added to each release of CodeCenter are included in the releases made available later. For example, features added to CodeCenter Version 4.1.0 are still available in CodeCenter Version 4.1.1, 4.2.0, etc. unless otherwise documented.

More details about the CodeCenter releases can be found within the *CodeCenter Release Notes* as well as the *CodeCenter Platform Guides* for each release. Refer to the main

CodeCenter page on CenterLine's website to access these documents relative to your specific release of CodeCenter.

New Features in CodeCenter Version 4.2.1

We have added the following new features in Version 4.2.1

- Support for Solaris 2.7 operating systems.
- Bug fixes
- Online documentation files and documentation viewer upgraded.

All features supported in CodeCenter Version 4.2.0 are also supported in CodeCenter Version 4.2.1, with the exception of instrumentation. Refer to the ***"Object Code Instrumentation Capability Removed"*** section of the **CodeCenter Version 4.2.1 Release Notes** for more details.

64-bit Support on Solaris 2.7

With the introduction of Solaris 2.7, comes 64-bit technology. CodeCenter Version 4.2.1 and the CenterLine-C compiler (**clcc**) are products that can operate on a Solaris 2.7 system since they are "64-bit aware". Meaning they will compile and interpret C programs running on a 64-bit aware operating system, such as Solaris 2.7. But these products have not been enhanced to provide any special 64-bit support.

If you have any 64-bit related questions/issues when using CodeCenter Version 4.2.1 on Solaris 2.7, please direct all inquiries to CenterLine Technical Support at (781) 444-8000 or via email at **info@centerline.com**

Online Documentation

With the introduction of CodeCenter Version 4.2.1 comes a new format of the online documentation. We have replaced the original *Dyna*Text Viewer with **Adobe Acrobat Reader** for SunOS, Solaris and HP systems. If you don't have a copy of **Adobe Acrobat Reader** for your particular system, the CodeCenter Version 4.2.1 installation process will automatically install a copy of **Adobe Acrobat Reader** for those platform releases of CodeCenter being installed. For example, if you install the Solaris and HP versions of CodeCenter, you will automatically obtain a Solaris and HP version of **Adobe Acrobat Reader**.

The documentation files used with the *Dyna*Text viewer have also been replaced. All documentation files used with **Adobe Acrobat Reader** are in **PDF** format.

The online documentation files in **PDF** format can be found in:

.../docs

To start the online documentation separately from the CodeCenter product, simply run the command:

.../bin/cldoc2 &

Along with being able to start the **Adobe Acrobat Reader** stand-alone and separately from CodeCenter using the above command, you can also start it from within CodeCenter via the **Manual Browser** option in the **Browser** menu.

All future releases of CodeCenter following CodeCenter Version 4.2.1 will use **Adobe Acrobat Reader** and **PDF** documentation file format.

Support for Sun's Debugging Information

CenterLine has now added support for Sun's debugging information, such that CodeCenter users who use Sun's C compiler instead of the CenterLine-C compiler (**clcc**) no longer need to use the **-xs** switch during compilation and linnking in order to have CodeCenter's **pdm** work properly. In earlier releases, the **-xs** switch was necessary when debugging fully built executables within CodeCenter's **pdm** if the executable was built using Sun's C Compiler instead of CenterLine's.

New Features in CodeCenter Version 4.2.0

We have added the following new features in Version 4.2.0:

- Support for Solaris 2.5, 2.5.1, and 2.6 and SunOS 4.1.4 operating systems.
- Support for Sun UltraSPARC workstations
- HP 9000 Series 700 workstations running HP-UX 10.01, 10.10 and 10.20.
- Bug fixes
- New install and licensing procedures

All features supported in CodeCenter Version 4.1.1 are also supported in CodeCenter Version 4.2.0.

Like Version 4.1.1, CodeCenter Version 4.2.0 supports the following SUN and HP Compilers (in addition to CenterLine-C):

- **clcc** (CenterLine-C)
- Sun/SPARC-C (all versions)
- **gcc** v2.5.8 **--ONLY--**
- FORTRAN
- HP-C (all versions)

Refer to the **Product Compatibility Matrix**, the **CodeCenter Version 4.2.0 Release Notes** or the **CodeCenter Version 4.2.0 Platform Guide** on CenterLine's website (**www.centerline.com**) for additional product support/compatibility information.

Licensing and

With the release of CodeCenter Version 4.2.0, we have enhanced the

Installation Enhancements in CodeCenter 4.2.0

installation process once again. Instead of running a **tar** command followed by the **RUN_ME** script, we provide a single **install.sh** script to perform these two tasks for you in one single step. To reflect these changes, we have also updated the ***Installing CenterLine Evaluations*** and ***Installing and Managing CenterLine Products*** guides.

Additionally, we provide a later release of FlexLM licensing software. CodeCenter Version 4.2.0 now uses FlexLM Version 5.0a.

New Features in CodeCenter Version 4.1.1

SUN Platform Support

This release adds support for the following:

- The Solaris 2.4 operating system
- The Sun SPARCompiler C Version 3.0.1
- Sun SPARC 5 workstations running SunOS 4.1.3_UI or Solaris 2.3 or 2.4

Support for Solaris 2 non-PIC Shared Libraries

We have added support for shared libraries that were not built for position independent loading to the Solaris 2.x platform. You can now load shared libraries with non-PIC relocations, including the Sun **XGL** and **XIL** graphics libraries.

Online Documentation

Several of the CodeCenter manuals are now available online. To access the new online documentation, select **Manual Browser** from the **Browsers** menu on any primary window, click on the "?" button in the Main Window, or issue the **cldoc** command from a shell.

In the left panel of the **Library** window are one or more collections of books. Click on the name of a collection to display the names of all the books in that collection in the **Books** panel. You can open a specific book by double-clicking on its name, or selecting its name and clicking the **Open** button.

You can perform a simple search from the *Dyna*Text **Library** window, and you can perform more complex searches from a book window. Select **Forms** from the **Search** menu to see which other search forms are available. Use the **Next**, **Previous**, and **Go Back** buttons to navigate through the book.

Underlined text is hot - clicking on it scrolls the window to the section of the book referenced, or opens a new window if the reference is to another book.

You can create a history of your movement through the book by selecting

New Journal from the **File** menu and selecting **Start Record** in the dialog that pops up.

Print sections by selecting **Print** from the **File** menu and highlighting the sections you want to print. Export sections to a file by selecting **Export** from the **Edit** menu, highlighting the sections you want to export, selecting **Content** as the **Export format**, providing a filename, and clicking the **Export** button.

Figures, tables, and tips are shown as icons. Double-click to open them.

New Features in CodeCenter Version 4.1.0

Thread Support on Solaris

We've added support for threaded applications on the Solaris 2.3 platform in process debugging mode (**pdm**), with the ability to debug threads in executables and a graphical **Thread Browser** to show the status of all the threads in your program.

In process debugging mode, the **Thread Browser** gives you information about the threads and lightweight processes in your program. This information includes a list of all threads, and the state of each thread. The state information includes the function the thread is executing, the execution state (for example, **running**, **sleeping**) of the thread, and the start function for the thread.

At any given time, the **Thread Browser** focuses on a single thread or light-weight process (**LWP**), known as the "current active entity." You control execution of the current thread with the **cont**, **next**, **nexti**, **step**, and **stepi** commands. You can display a traceback of the thread execution stack with the **where** command. You can also perform these operations on another thread at the break level by making it the current active entity. To make another thread the current active thread, you use the **thread** command with the new thread number as an argument.

CodeCenter's **pdm** supports threads and threaded applications, where the types of threads that can be used are **Solaris-based** thread types. **POSIX** and other thread types are not supported by CodeCenter. Solaris is the only operating system where threads are supported. CodeCenter for HP-UX or SunOS do not support threads.

For more information about debugging threaded applications, see the **thread**, **threads**, and **thread support** entries in the **Manual Browser**.

Emacs Main Window

If you're an **FSF GNU Emacs 19** user, you can start up a CodeCenter session from within Emacs.

First you need to add the following lines to your **.emacs** file:

(setq load-path (cons ".../lib/lisp" load-path))
(load "clipc")

The **M-x codecenter** command starts the environment as a subprocess of Emacs, with the menus from the **CenterLine Main Window** replacing the menus at the top of your **Emacs window**. Edit the path name supplied if you want to run a different version of CodeCenter. You can give a project name as an argument. To invoke CodeCenter in process debugging mode, use the **-pdm** switch. Edit your code directly in the **Source area** at the top of the **Emacs Main Window**.

All the browsers are available with the same commands and menu items you use when you invoke CodeCenter from a shell. A separate **Button Panel** window, including your own user-defined buttons, can be launched from the **Browsers** menu in the **Main Window**.

For more information, see the **emacs integration** entry in the *CodeCenter Reference Guide*.

Printing from the Cross-Reference Browser

You can now print the contents of the **Cross-Reference Browser** to a postscript file. When you click on the **Print...** button in the **Browser**, the **Print from Browser** dialog box is displayed. You can specify what paper size to use, the title of the printout, and the name and location of the output file. The default location is your current directory. You can also specify how many pages the output should be printed on. For example if you specify an output page height of **2** and width of **3**, CodeCenter resizes the output to fit on six sheets of paper.

Frequently Asked Questions in the Manual Browser

We've taken the questions that customers asked most often and put the answers in the **Manual Browser**. Click on the "?" button in the **Main Window** to open the **Manual Browser**.

Select ***CodeCenter User's Guide*** from the list of books, and then select ***Appendix A, "Frequently Asked Questions"***, from the **Table of Contents** panel. Click on the "+" icons in the **Table of Contents** panel to view the topics covered. You'll find topics such as *"How do I fix unresolved symbols?"* and *"I'm having trouble loading with make"*.

Workspace Improvements

We have made several changes to enhance performance and behavior in the **Workspace**. The most important changes are described here. In addition to the changes described below, you can now do the following in the **Workspace**:

- Set actions on variables defined in the **Workspace**

- Define **macros** to have the value **0**
- Declare a single-declaration **extern** function without enclosing the single declaration in braces
- Define a **main** with a single line and set breakpoints on it.

New Command:
`load_header`

We have added a new command, **load_header**, for loading the definitions from header files into CodeCenter. Use the **load_header** command to load non-local header files without specifying a path, and to load multiple header files into a single module. If you want to load a header file in your working directory or path, you can use use the **load** command to load it.

The **load_header** command replaces the **#include <header_file.h>** syntax. There were several problems related to the use of **#include** in the **Workspace**. For example, because definitions were parsed one at a time, if an error was encountered while parsing a header file, previous definitions were not undefined. As a result, users often had to issue an **unload** workspace command before they could reload a header file.

In addition, the **Workspace** does not match the separate compilation model of C. To enable the **Workspace** to function as a debugger, definitions in a header file included in the **Workspace** are visible across modules. As a result, using **#include** in the **Workspace** occasionally causes CodeCenter to pick up incorrect definitions from included files.

The new **load_header** command lets you load a header file into the environment in a separate module, using the paths defined in the **load_flags** and **sys_load_flags** options to locate the file. You can use **load_header** to load system header files such as **stdio.h** without specifying the path to the file.

You can also load multiple header files into a single module with **load_header**. For example, if the local header file **rect.h** has dependencies on definitions in **math.h** and **limits.h**, you can load all three header files into a single module with this command:

-> load_header <math.h> <limits.h> "rect.h"

For complete syntax and usage information, please refer to the **load_header** entry in the **Manual Browser**.

We have provided a new option, **workspace_include**, to provide backwards compatibility for users who have existing project files that use the **#include** syntax. We do not recommend that you set this option for new projects. The option is described in the **load_header** and **options** entries in the **Manual Browser**.

`edit workspace`
Enabled

You can now save code you define in the **Workspace** more easily. During a CodeCenter session, all C definitions you enter are stored in a **Workspace**

scratchpad. The **edit workspace** command lets you save the scratchpad to a file, by default **workspace.c**, and then edit the file.

For example, suppose you create a program fragment in the **Workspace**. You can create stubs for external functions called by the code, and then execute the code to test it.

After testing your code, you can use **edit workspace** to create a file containing the code you defined in the **Workspace**. You can enter your own name for the file or accept the default, **workspace.c**.

-> edit workspace

Appending all workspace definitions to a file.
Default filename is "workspace.c" in the current directory.
Please specify a filename, press Return to accept default,
or <CTRL-D> to abort:

If you want to test a particular set of definitions, edit the file so that it contains the definitions you want to test. Then use the **unload workspace** command to unload all the definitions and objects you created in the **Workspace**, and use the **source** command to load the definitions in your saved file back into the **Workspace**. Note that the **source** command will report errors if you've unloaded any definitions that the saved file depends on.

If you want to use the new file as source code, add any **#include** lines you need and remove any extraneous lines. For example, some CodeCenter commands, such as **whatis**, will appear in the file.

> **NOTE:** When using code developed in the **Workspace**, remember that **static** functions and variables are visible at global scope in the **Workspace**. As a result, you may have to make **static** functions externally visible to use the **Workspace** sources as a separate file.

`unload workspace` Improved

The **unload workspace** command now only undefines user-defined variables. Previously, when you used the **unload workspace** command, all variables were undefined, including predefined macros such as **__CENTERLINE__**. CodeCenter now resets predefined macros after an **unload workspace**.

HP `clcc` Compiler

CenterLine's C compiler (**clcc**) is now available on the HP platform. The CenterLine-C compiler is an ANSI C optimizing compiler designed to achieve small code size. The compiler is also compliant with K&R C and is link compatible with Sun and HP compilers.

Enhancing pdm Debugging

In addition to support for debugging threaded applications, we've made some other improvements to pdm. The process debugging mode in CodeCenter 4.1.0 is based on Version 4.12 of **gdb** and takes advantage of its new features.

Error Browser Layout Improvements

We've streamlined both the appearance and the performance of the **Error Browser**. We've moved some of the buttons from the bottom of the **Browser** to the side and added **Edit**, **Reload**, and **Build** buttons to the side button panel to make it easier to fix and reload your code. We removed the redundant buttons used to suppress warning messages, so now you use just the **Suppress** menu to control suppressions. The new **Error Browser** uses less memory and performs faster.

Changing the Size of Your Windows

We've added Motif-style resizeable panes to the **Main Window** and **Thread Browser**. For example, to change the relative size of the **Source** panel and **Workspace**, place your cursor on the pane control sash, which is a square box at the bottom of the **Source** panel pane, just under the **Error Sentinel**. Hold down the **Left** mouse button and drag the sash up or down to resize the panes.

X Resource Changes

The Graphical User Interface in this release of CodeCenter was built with a new release of the **OI** toolkit. This change made it necessary to set more specific font and color resources for some elements of the GUI. You can override some of these settings by using the switches described in **Table 7** in the *CodeCenter Reference Guide* (online) or by changing your X resources as described in ***"Modifying X Resources"*** in the *CodeCenter Reference*.

HP Softbench 3.2 Support

The CodeCenter integration with **SoftBench** supports **Softbench** messages through CenterLine's Application Programming Interface, or API, using a Gateway to translate the messages. With this release we've added a new switch to the **codecenter** and **objectcenter** commands that starts up the Gateway automatically.

To use your CenterLine environment and **Softbench** together, first start up **Softbench**. Then invoke CodeCenter with the **-softbench** switch:

% codecenter -softbench

This command sends a **START** message to the **SoftBench Tool Manager**, which places CodeCenter in the **Tool Manager's window**.

The CenterLine environment and the **SoftBench Tool Manager** send messages to each other through the Gateway. For example, CodeCenter sends an **EDIT-WINDOW** message to the **Broadcast Message Server**

when it needs to bring up an editor. The **SoftBench** editor loads and displays the file.

> **NOTE:** Pretty much all versions of Softbench compiler are supported by CodeCenter. Any issues using a specific release beyond version 3.2, please contact CenterLine Technical Support at (781) 444-8000 or via email at **support@centerline.com**.

To terminate the Gateway and remove CodeCenter from the **Tool Manager window**, use the CodeCenter **quit** command.

For more information about CenterLine's API or Gateway, enter **man CLIPC** or **man CenterLine API** in the **Workspace**, or **man clms_gateway** in a shell.

Licensing and Installation Enhancements

To make the installation process easier, we've made some changes to the **RUN_ME** script and made the licensing error messages more informative. We've also updated our installation manuals, ***Installing CenterLine Evaluations*** and ***Installing and Managing CenterLine Products***.

Changes Between Version 4.0.0 and Version 4.1.0

This section describes improvements made in CodeCenter in the point releases since Version 4.0.0.

Memory Leak Detection

Memory leak detection identifies potential memory leaks by reporting on the memory that the program allocates while running and fails to free before exiting.

The memory leak detection report lists leaks by the size of the memory allocated and identifies where the program allocated the memory in the stack trace. In addition, it shows the number of times the leak occurred there. For more information, see the **memory leak detection** entry in the **General Topics** category.

Run Window Changes

By default, the input and output of your program previously went to the terminal in which you invoked CodeCenter without spawning an independent **Run Window** and without returning control to the shell.

Now, CodeCenter by default opens a separate **Run Window** for your program's input and output and returns control to the shell in which you invoked CodeCenter. A CenterLine program called **clxterm** creates this **Run Window**, which is a standard version of **xterm**, the **X11** terminal

emulator.

To avoid creating the separate **Run Window** and avoid returning control to the shell, use the **-no_run_window** switch when you invoke CodeCenter. The program's input and output goes to the shell in which you invoked CodeCenter. Using the **-no_run_window** switch means you are unable to interrupt CodeCenter and unable to place it in the background. This option is intended for debugging applications that need specific terminal support rather than a generic terminal such as **xterm**.

> **NOTE:** Avoid starting CodeCenter in the background using the **-no_run_window** switch. Your program could have undesirable input/output behavior.

To create a separate **Run Window** and avoid returning immediate control to the shell, use the **-no_fork** switch. With **-no_fork**, control returns when you enter the suspend character (usually **^Z**) in the shell or exit CodeCenter. After you type the suspend character in the shell, you must type **bg** to enable your program to direct output again to the **Run Window**. Without **-no_fork**, the shell prompt comes back immediately.

By default, issuing the **run** or **start** command deiconifies the **Run Window**. To prevent deiconifying the **Run Window**, use the **win_no_raise** option. Setting this option prevents the deiconification of the **Run Window** when you issue **run** or **start**.

The **close-window** (**f.delete**) **window-manager** operation now iconifies the **Run Window**. This is to prevent the accidental destruction of the **Run Window**.

We added unique resource names for the **Run Window**, generic terminals, and the **vi** Edit window.

- The **Run Window** resource is:

  CodeCenter*RunWindow.xterm-resource

- The generic-terminal resource is:

  CodeCenter*Terminal.xterm-resource

- The **vi** Edit window is:

  CodeCenter*EditWindow.xterm-resource

Scrollbars are now the default in the **Run Window** and in the generic terminal windows. Disabled scrollbars continue to be the default for the **vi** Edit window.

For more information about setting resources for these windows, see the ***"Run and Edit Windows"*** entry in the **Manual Browser**.

`clxterm` Resources in `app-defaults`

CodeCenter used to require you to use **.Xdefaults** for setting **clxterm** resources. This is no longer true. CodeCenter's **clxterm** now reads the **app-defaults** files of the Graphical User Interface as well as **.Xdefaults**.

This means you can put all GUI-related resources (those for the user interface itself as well as for **clxterm**) into the same **app-defaults** file. You no longer have to split them up into two different files, nor do you have to use your **.Xdefaults** file for program-specific resources.

The app-defaults file for CodeCenter is:

$XAPPLRESDIR/CodeCenter

See the **Xresources** entry in the **Manual Browser** for more information.

More Flexible Resource Setting

The CodeCenter Graphical User Interface used to require that you use a model-specific resource setting to set the colors of some objects. This is no longer true. You can set colors of objects in the same way in **Motif** and in **OPEN LOOK**. This is the syntax for the Graphical User Interface:

CodeCenter*Color*OI_scroll_text.@text.Background: color

This is the syntax for the **Workspace**:

CodeCenter*Color*Workspace.@text.Background: color

Although a model-specific setting is no longer a requirement, you can choose model-specific settings if you want a certain standard set of localizations for **Motif** users, and a different set for **OPEN LOOK** users.

New Resource For Control Buttons

We have added a new resource that can improve performance by reducing the **X11** server traffic that results from dimming the control buttons in the GUI. This resource is especially valuable when running CodeCenter with slow X servers or low-speed connections such as X over serial lines.

In previous releases of CodeCenter, the control buttons dimmed as soon as the component debugger became busy. The new resource enables you to specify the length of time that the debugger must be busy before the control buttons on the GUI dim. By default, the buttons on the GUI dim when the debugger has been busy for 1.15 seconds:

CodeCenter*dimButtonsWhenDebuggerBusy: 1.15

You can change the value of this resource in the site-wide application defaults file for CodeCenter, or in your local **.Xdefaults** file. The value can be:

- The string **Always** if you want the buttons to dim as soon as the debugger is busy.
- The string **Never** if you never want the buttons to dim.
- Any positive floating-point number, to indicate the number of seconds you want to elapse before the buttons start dimming.

Support of Direct Pasting into Emacs

Previously, the Graphical User Interface of CodeCenter did not handle **CUT_BUFFER0**, the text transfer mechanism that **GNU Emacs Versions 18 and 19** use to paste text from other applications. If you wanted to copy text from CodeCenter and paste it into an **emacs** buffer, you had to run the **xcutsel** program to act as an intermediary.

With this version of CodeCenter, running **xcutsel** is unnecessary. The user interface now automatically exports text to **CUT_BUFFER0** whenever you highlight text in labels, entry fields, and multi-line text objects. You can paste this text directly into emacs in the same way you paste text from an **xterm** into **emacs**.

Emacs-like Keybindings

These shell-like and Emacs-like keybindings are now available in CodeCenter Version 4.1 by default:

Control-a	**Beginning of line**
Control-e	**End of line**
Control-b	**Backward character**
Control-f	**Forward character**
Meta-b	**Backward word**
Meta-f	**Forward word**
Control-n	**Next line**
Control-p	**Previous line**
Control-d	**Delete next character**
Control-u	**Delete to beginning of line**
Control-k	**Delete to end of line**
Control-w	**Delete previous word**

In **Motif**, some windows may use **Meta + B** and **Meta + F** as menu mnemonics, rendering them unavailable in text objects.

As a result of this change, the information in the X resources entry in the *CodeCenter Reference* about setting translations for underlying objects in your **.Xdefaults** file is now obsolete.

User-Defined Commands Access

A number of customers said they missed the **\L** (current **Source** area line number) facility that CodeCenter used to provide in user-defined

Line Numbers

commands. Without **\L**, writing certain kinds of user-defined commands was impossible. For example, you were unable to write a command to select a line in the **Source** area and set a breakpoint or action on that line.

Although CodeCenter Version 4.1 allows no editing in the **Source** area and the concept of current **Source** area line number does not really apply, we have provided an equivalent facility that works in terms of the **Source** area text selection.

These four new keywords are now available in user-defined commands:

$first_selected_line $first_selected_char $last_selected_line $last_selected_char

The **$first_selected_line** and **$last_selected_line** keywords provide you with the starting and ending line numbers of the **Source** area's current text selection. Lines are numbered beginning with **1**. If no text is selected in the **Source** area, both of these keywords return **0**.

The **$first_selected_char** keyword provides the position of the first character selected on **$first_selected_line**. The **$last_selected_char** keyword provides the position of the last character selected on **$last_selected_line**. Character positions are numbered beginning with **1**, and tabs are considered to be a single character. If no text is selected in the **Source** area, both of these keywords return **0**.

Directing Output to the Workspace

You can direct your output to the **Workspace** by unsetting the **win_io** option.

We recommend that you keep the **win_io** option set, however, for complicated programs that use curses-style input and output. Unsetting **win_io** has the following limitations:

- Controlling-**tty** semantics are unavailable in the **Workspace**.

  This means that **tcgetpgrp** / **tcsetpgrp** and **tty**-generated signals will not work as expected.

- If your program affects the **tty** mode, it may affect the **Workspace** output.

- The **tty** mode may not be preserved across **Workspace** interactions. For example, when you continue from a breakpoint, the **tty** settings may not be the same as when you stopped.

To unset the **win_io** option, enter this in the **Workspace**:

unsetopt win_io

Your output will go to the **Workspace** at your next **reinit**, whether it is an implicit **reinit** (for example, when you issue the **run** command) or an explicit **reinit** (by issuing the **reinit** command).

Two New pdm Options and Switches

We have added two new options and switches in process debugging mode (**pdm**). The new options are **class_as_struct** and **full_symbols**.

The option **class_as_struct** has these characteristics:

Type: Boolean
Default Value: FALSE
Commands Affected: debug

When **class_as_struct** is false, CodeCenter reads the class debugging information produced by the compiler. This enables CodeCenter to provide full class information when needed.

When **class_as_struct** is true, CodeCenter ignores class debugging information produced by the compiler. This causes CodeCenter to treat classes as C structures. For example, the **whatis** command will only display data members and not member functions. Setting this option will give shorter initialization time but less debugging information for classes.

You can set the value of **class_as_struct** by adding this line to your **.pdminit** file:

```
setopt class_as_struct
```

You can also set it by issuing this command in the **Workspace** before issuing the **debug** command:

```
-> setopt class_as_struct
```

Use the **unsetopt** command to reset the **class_as_struct** option to false.

The option **full_symbols** has these characteristics:

Type: Boolean
Default Value: FALSE
Commands Affected: debug

When **full_symbols** is false, CodeCenter reads only part of the debugging information to shorten initialization time. Additional debugging information will be read as needed, such as when you issue the **whatis** or **list** command. When **full_symbols** is true, CodeCenter reads all the debugging information at initialization.

You can set the value of **full_symbols** by adding this line to your **.pdminit**

file:

setopt full_symbols

You can also set it by issuing this command in the **Workspace** before issuing the **debug** command:

-> setopt full_symbols

Use the **unsetopt** command to reset the **full_symbols** option to false.

For more information about options, including setting, unsetting, and displaying them in the **Workspace**, see the **options** entry in the **Manual Browser** or the *CodeCenter Reference*.

The new switch **-class_as_struct** corresponds to the **class_as_struct** option. The new switch **-full_symbols** corresponds to the option **full_symbols**. For example, with the following command, you can invoke CodeCenter in **pdm** mode with **full_symbols** set:

% codecenter -pdm -full_symbols

New Switches For `contents` and `link` Commands

We have added a switch called **-ascii** to the **contents** command and a switch called **-list** to the **link** command.

The **contents -ascii** command displays the output of the **contents** command in the **Workspace** instead of invoking the **Project Browser**. The **link -list** displays the library link order in the **Workspace**. This switch is useful for diagnosing link-order related problems in the interpreter.

New Scope for `unsuppress` Command

We have added **everywhere** as a new scope argument for the **unsuppress** command.

You use **everywhere** in combination with a CodeCenter violation number (error or warning) to unsuppress a CodeCenter violation while you are debugging.

The scope argument **everywhere** unsuppresses a violation wherever you had suppressed it without a location-specific scope argument. (A location-specific argument specifies a line number, file, directory, function, library, or identifier).

If you had used a **location-specific** scope to suppress the violation to begin with, the violation stays suppressed at that location if you use **unsuppress** with **everywhere**. To unsuppress the violation at that location, you must use the **unsuppress** command with the **location-specific** scope you had used to suppress it.

This is the syntax for global unsuppression of a violation with the number **num**:

unsuppress num everywhere

Alternatively, you can unsuppress all occurrences of a violation with one command regardless of whether you suppressed them with a **location-specific** argument. This is the syntax:

unsuppress num

For more information about suppressing and unsuppressing violations, see the **suppress** and **unsuppress** entries in the **Manual Browser** or *CodeCenter Reference*.

Mnemonics

Each CodeCenter primary window provides **Motif**-style mnemonics for almost every menu item on its menu bar.

Mnemonics are not available for items that you can create and destroy on the fly during a session , such as items on the **User Defined** submenu of the **CodeCenter** menu in the **Main Window**.

A menu item with a mnemonic has one of its letters underlined, usually the first one. To select an item with mnemonics,

1. Press the **Meta** key and the underlined-letter key at the same time, to display the menu.
2. Press the underlined-letter key of the menu item.

New Warning Message for `free(0)`

CodeCenter has added the following new warning message for **free(0)**:

Warning #106 Freeing NULL pointer

CodeCenter used to give this warning message for **free(0)**:

#95 (Cannot free memory address (not within data space))

The reason for the new warning is that **POSIX** and **SVID** allow **free(0)**. With the separate warning message, you can suppress the warning separately and, thus, run **POSIX / SVID** compliant code without losing the ability to check bad calls to **free()**.

Here is an example of the new warning:

"d410fix.c":432, d4_10_3(), Freeing NULL pointer (Warning #106)

```
431: #if ANSI && D410B
432: free(NULL);
433: #endif /* ANSI */
```

Use of **free(0)** is not portable.

Options Browser has Longer Editable Fields

In the **Options Browser**, the maximum input lengths of text entry fields used to be 512 characters. We have increased this maximum to 1000 characters.

If you need an editable field longer than 1000, you must use the equivalent **Workspace** command rather than the Graphical User Interface to perform the task. For example, if you want to set an option's value to more than 1000 characters, you must use **setopt** in the **Workspace**. The **Options Browser**, however, displays only the first 1000 characters of the value you set.

Longer Line Limit in the Source Area

The longest line the **Source** area could display used to be 500 characters. Now, the **Source** area can display up to 10,000 characters per line.

Variable Prevents Unexpected Behavior

Rapid signal delivery occurs when you use **setitimer(2)** to set an interval timer to expire after less than about 300 milliseconds or when a separate process sends signals to the CodeCenter process via **kill(2)** at intervals of less than about 300 milliseconds.

We added **CENTERLINE_RAPID_SIGNALS**, a new environment variable, to ensure robust behavior in the the presence of rapid signal delivery. In the presence of rapid signal delivery, CodeCenter can experience unexpected behavior, such as reporting that the user program generated a **SIGSEGV** or **SIGBUS** at some unknown location.

If you strongly suspect that rapid signal delivery is causing unexpected behavior, set **CENTERLINE_RAPID_SIGNALS** to any value before invoking CodeCenter.

Do not set this variable unless you suspect a problem with rapid signal delivery; doing so would cause a serious performance problem.

CodeCenter Documentation

This section describes the documentation for CodeCenter. CodeCenter Version 4.1.1 comes with hardcopy documentation and online documentation. CodeCenter Version 4.2.0 and 4.2.1 only come with online documentation.

Online Documentation

Access online versions of most CodeCenter manuals by selecting **Manual Browser** from the **Browsers** menu or by typing **cldoc** (for all CodeCenter

versions up through and including Version 4.2.0) or **cldoc2** (for CodeCenter Version 4.2.1 and up) in a shell. The following manuals are available online:

- *CodeCenter User's Guide*
  A task-based description of CodeCenter, explaining how to use the graphical user interface to load, manage, run, and debug programs within CodeCenter. An appendix to the online *User's Guide* contains *Frequently Asked Questions : Answers to some of the questions most often asked of CenterLine Technical Support*.

- *CodeCenter Reference*
  A complete reference for CodeCenter, containing an alphabetical listing and description of CodeCenter commands, functions, and informational topics. Appendices to the online *CodeCenter Reference* include *About The CodeCenter Release* (this guide, formerly called *About This Release*) and *CodeCenter Platform Guides* containing platform-specific information about CodeCenter.

- *CodeCenter Tutorial*
  A step-by-step introduction to CodeCenter features.

- *CenterLine-C Programmer's Guide*
  Information about the CenterLine-C compiler.

  NOTE: The older versions of CodeCenter came with the *About The CodeCenter Releases* (this guide, formerly called *About This Release*) and *Platform Guide* as part of the installation of the product. With the introduction of CodeCenter Version 4.2.1, CenterLine has now provided these manuals separately from the product installation. To review the CodeCenter *Platform Guides* related to your release of the product, see the main **CodeCenter page** on CenterLine's website. *Platform Guides* and *About The CodeCenter Release* (this guide, formerly called *About This Release*) additions for future releases of CodeCenter can be found on CenterLine's website.

In addition to the online documentation described above, the following information is available:

- Access context-sensitive help in the GUI version of CodeCenter by moving the cursor over the item you want information about and pressing **F1** or the **Help** key if your keyboard has one, or by selecting **"On Context"** from the **Help** menu and moving the ? cursor over the item and clicking. A **Help** window appears describing that item.

  If you have bound the **F1** key to a window manager operation, you are unable to access context-sensitive help with the **F1** key.

- Access information on a variety of topics from the **Help** menu, which

appears on every primary window.

- Access information about a command by typing help in the **Workspace** followed by the name of the command.

- Access any entry in the *CodeCenter Reference* by typing **man** and the name of the entry in the **Workspace**.

The online documentation available outside the CodeCenter environment is in this directory:

path/c_4.0.0/<arch>/docs

The word **path** represents the path to the CenterLine directory, and **<arch>** is a platform-specific directory, for example **sparc-sunos4**, **sparc-solaris2**, **pa-hpux8**, **i486-svr4**, **powerpc-aix** or **m88k-svr4**.

The **online** directory contains a file called **README**, which describes the files in this directory. Among the files are:

- **bugs.open**, which describes the known bugs, limitations, and workarounds for CodeCenter.

- **bugs.fixed**, which describes bugs fixed since the most recent version of CodeCenter

NOTE: Some of the above listed files/directories may not be available for your version of CodeCenter. Additionally, the **README** file in the **online** directory may not be available for your version of CodeCenter.

Hardcopy Documentation

This is the hardcopy documentation that comes with CodeCenter:

- *CodeCenter Read Me First Release Bulletin*
  The latest hardcopy information, containing any updates necessary to other hardcopy documentation.

- *Installing and Managing CenterLine Products*
  How to install CodeCenter and administer it, including how to troubleshoot licensing problems.

- *Getting Started with CodeCenter*
  A step-by-step introduction/overview to CodeCenter features.

- *CenterLine-C Programmer's Guide*
  Information about the CenterLine-C compiler.

NOTE: Many of the above listed documents are available on CenterLine's website (main **CodeCenter Page**). Additionally,

many of the documents included in the Online documentation section (above) can be obtained in Hard Copy format. Contact CenterLine for details.

Product Limitations

The following is a list of known current limitations with our product.

GUI Behavior

CodeCenter currently does not allow you to change the placement of the scrollbar. In addition, you cannot set the scrollbar placement with a window manager X resource (such as **Motif**'s **XmNscrollLeftSide** or **XmNscrollRightSide** or **OPEN LOOK**'s **OpenWindows.ScrollbarPlacement**). This is because CodeCenter is developed using the **Object Interface** (**OI**) toolkit, not the **Motif** or **OPEN LOOK** (**XView**) toolkits. The **OI** toolkit currently does not offer a resource to change scrollbar placement.

Caution if Editing `sys_load_flags`

CodeCenter supplies custom versions of header files, such as **stdarg.h** and **varargs.h**. The CodeCenter versions of the **stdarg.h** and **varargs.h** header files reside in the **CenterLine/include** directory.

The global **ccenterinit** file automatically supplies the **-Idirs/include/CodeCenter** value to the **sys_load_flags** option so that CodeCenter header files will be included vefore the standard versions in **/usr/include**.

If you edit **sys_load_flags**, be sure to keep the location of the **-Idirs/include/CodeCenter** directory before other **-I** switches so this support for variable argument functions remains unchanged.

ANSI Mode Interprets `#define` Incorrectly

CodeCenter in **ansi** mode does not interpret the ANSI C **#define** correctly. In the following example, CodeCenter produces the warning **Macro 'A' requires 1 parameters, but only 0 are given**, when the code should produce the output **HELLO**:

```
#define HELLO
#define A(x) #x
main() {
char *s = A(HELLO);
printf("%s\n", s);
}
```

This is because defining **HELLO** gives empty content to the definition and the parser never sees the argument. The work-around is to define **HELLO** as something, any value will do. For example:

#define HELLO 1

Support for Variable Argument Functions

CodeCenter supports the use of variable function arguments loaded in source and object code. It supports both ANSI C (**stdarg.h**) and K&R C (**varargs.h**).

However, the following restriction applies to loading source code. You must not use a **structure** or **union** whose size is larger than **8 bytes** as a fixed argument in a variable-argument function. If you do, the interpreter computes the address of the arguments incorrectly.

Data Browser

Due to window size limitations in X, the **Data Browser** has a limit to the number of items it can contain. The limit is determined at run time. The font and model (**Motif** or **OPEN LOOK**) you use affect the limit. Using default fonts, you can create about 900 fields per data item under **Motif** and about 1500 under **OPEN LOOK**.

Source Area

There is a limitation to the number of lines that you can list in the **Source** area. You can list files up to about 30,000 lines.

Limited Thread Support

CodeCenter's **pdm** and CodeCenter's C compiler currently support threads on Solaris platforms **ONLY** (Solaris 2.3 and *_possibly_* later releases) as long as **Solaris-based** threads are defined. **POSIX** style and other thread types (such as GNU-based threads) are not supported. Threads are not supported at all on the HP and SunOS platforms. CodeCenter's **cdm** does not support threads at all on any platform supported by CodeCenter.

Limited Signal Handling Support

CodeCenter does not support the **siginterrupt()** and **sigstack()** signal functions in component debugging mode.

CodeCenter Version 4.1 and up supports calling **sigsetjmp()** and **siglongjump()** from an application running in component debugging mode with the following limitation:

> Every call to **sigsetjmp()** behaves as if the second argument were the value **0**, regardless of the actual value of the second argument. This means that a call to **siglongjmp()** will never restore the signal mask to the value it had at the time of the corresponding call to **sigsetjmp()**. In other respects, **sigsetjmp()** and **siglongjmp()** behave exactly like **setjmp()** and **longjmp()**.

There may be additional limitations in signal handling support on your platform. Please refer to the "Anomalies" section in your *Platform Guide*. For the *Platform Guide* related to your release of CodeCenter, see the main **CodeCenter Page** on CenterLine's website.