What's New in the Oracle® Developer Studio 12.5 Release

Part No: E60742
July 2016

What's New in the Oracle Developer Studio 12.5 Release

Part No: E60742

Référence: E60742

Contents

Using This Documentation

- **Overview** – Describes the new and changed features in the compilers and tools with this Oracle Developer Studio 12.5 release
- **Audience** – Application developers, system developers, architects, support engineers
- **Required knowledge** – Programming experience, software development testing, aptitude to build and compile software products

Product Documentation Library

Documentation and resources for this product and related products are available at `http://docs.oracle.com/cd/E60778_01`.

Feedback

Provide feedback about this documentation at `http://www.oracle.com/goto/docfeedback`.

♦ ♦ ♦ C H A P T E R 1

Introducing the Oracle Developer Studio 12.5 Release

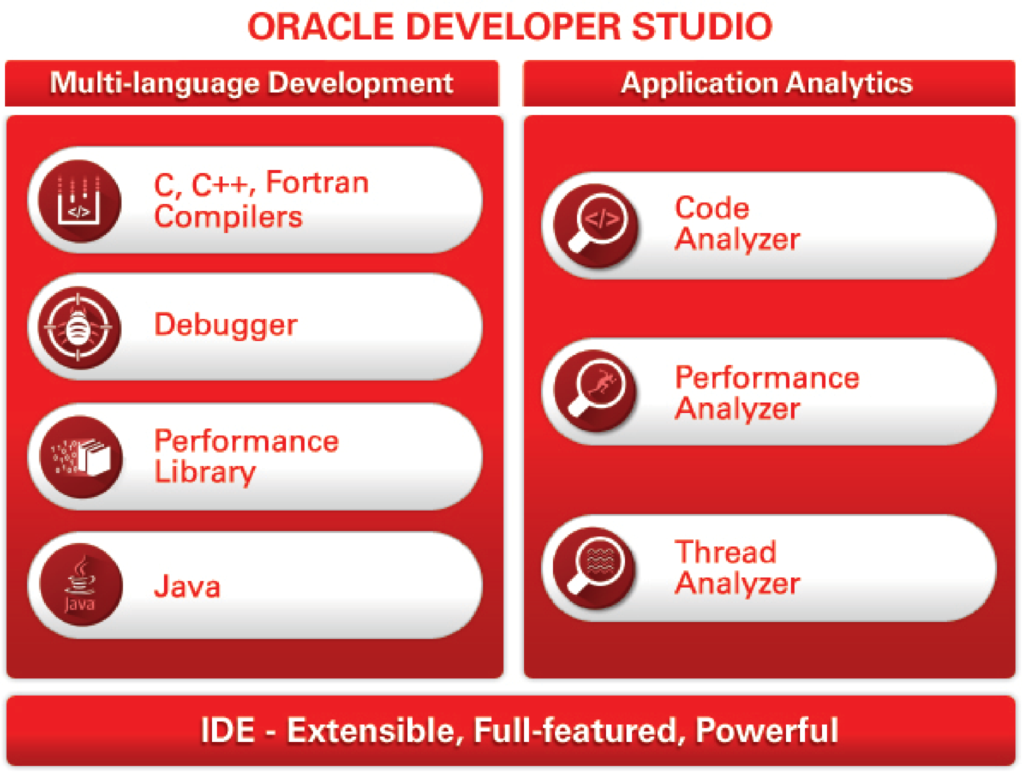
This chapter provides an overview of the key updates in this release.

- "Overview of Oracle Developer Studio" on page 9
- "Key Features in this Release" on page 10

Overview of Oracle Developer Studio

Oracle Developer Studio includes a compiler suite, an analysis suite, and a graphical integrated development environment (IDE) that is tailored for use with the compilers and tools from both suites. Together they provide a development environment that is optimized for developing applications with the best performance on Oracle Sun hardware.

FIGURE 1 Diagram shows compiler and analysis suites integrate with the IDE



Key Features in this Release

Oracle Developer Studio 12.5 delivers highly optimized compilers, advanced analysis tools, and a multi-language aware IDE for easy development of fast, reliable, and secure applications for Oracle Solaris and Linux operating systems. Oracle Developer Studio tools are optimized to complement the complete hardware and software stack and enable development teams to write better code faster.

- **Raise the Bar on Security**

    - Oracle SPARC M7 Silicon Secured Memory (SSM) integration in developer tools delivers real-time memory access checking and a new library enables apps to use SSM in custom memory allocators. See Chapter 4, "Code Analysis Tools" for more information.
    - Built-in security checking during source code compilation helps identify security vulnerabilities earlier in the development and testing process. See Chapter 2, "C++ Compiler", Chapter 3, "C Compiler", and "Changes to Compilers" on page 47 for more information.
    - Automatic stack overflow protection at application runtime minimizes potential security vulnerabilities
    - Secure coding hints in IDE to detect use of unsecure and deprecated functions and offer suggestions on more secure alternatives. See Chapter 7, "Oracle Developer Studio IDE" for more information.
- **Increase Productivity**
    - Easy compilation of open source applications with expanded GNU support and binary compatibility. See *Oracle Developer Studio 12.5: GCC Compatibility Guide* for more information.
    - Popular C++14 features and full support for C++11 and C11, including concurrency and atomics libraries
    - Improved testing and support of latest Boost libraries
    - Accurate and easy performance analysis of Java, C, C++ applications. See Chapter 5, "Performance Analysis Tools" for more information.
    - Efficient code editing of large enterprise applications, delivering up to 7x faster parse time than open source alternatives. See Chapter 7, "Oracle Developer Studio IDE" for more information.
- **Maximize Performance**
    - Generate up to 4.5x faster code with performance optimizations for the latest Oracle systems (SPARC and x86), including SPARC S7
    - Updated Performance Analyzer with latest SPARC hardware counters. See Chapter 5, "Performance Analysis Tools" for more information.
    - All tools provided as 64-bit binaries

♦ ♦ ♦ C H A P T E R 2

C++ Compiler

This chapter describes the new and changed features in this release of the Oracle Developer Studio C++ compiler.

About the C++ Compiler

This section provides a summary list of the new features and changed functionality introduced in the Oracle Developer Studio 12.5 C++ 5.14 Compiler release.

The C++ compiler (`CC`) produces code that is targeted for specific operating systems, processors, architectures, memory models (32-bit and 64-bit), floating-point arithmetic, and more, according to command-line options you specify. The compiler automatically parallelizes serial source code to produce binaries with better performance on multicore systems and can also prepare binaries for enhanced debugging or analysis by other Oracle Developer Studio tools. The compiler also supports GNU C/C++ compatibility features.

Partial Support for the C++14 Standard

The new C++14 standard strengthens C++, giving you additional tools to help you make your code even cleaner and safer. The compiler retains accelerated SPARC and x86 performance on Oracle hardware.

This is the first Oracle Developer Studio release to include support for the C++14 standard. The following features of the C++14 Standard are supported:

- Binary literals
- Sized deallocation
- `deprecated` attribute
- Single-quote digit separator
- Member initialization and aggregates

Additional C++ Compiler Changes

The following lists the new and changed features in this release of version 5.14 specific to the C++ compiler.

The C++ compiler changes include the changes that are described in "New and Changed Features Common to the Compilers" on page 47.

For details, see the *Oracle Developer Studio 12.5: C++ User's Guide* and the `CC`(1) man page.

- **Change in default compilation mode** — The default compilation mode on Oracle Solaris is `-compat=5` with `-library=Cstd` (C++03 mode with the Sun ABI and `libCstd` library). The default compilation mode on Linux is `-std=c++03` (C++03 mode with the g++ ABI and runtime libraries).
- **Support for C++11 standard features:**

  Oracle Developer Studio 12.5 C++ completes its support for C++11 with the addition of the following items:

  - Concurrency and atomic operations

    Note - Using the atomics feature requires special attention to runtime support. For more information, see "Bundled Atomics Library" in *Oracle Developer Studio 12.5: C User's Guide*.

  - User-defined literals
- **New compiler options:**
  - `-pedantic` — Emits warnings or errors for code that is accepted by default but does not conform to the C++ Standard.
  - `-abiopt=[mangle5|mangle6]` — Available only in `-compat=5` mode. The default is `mangle6`, for correct name mangling. On Oracle Solaris SPARC and on Oracle Solaris x86 with the `-m32` option, you can specify `mangle5` for compatibility with possibly buggy name mangling of older compilers.

- `-xcheck=noreturn` — Informs the compiler to add code to cause a runtime error if a routine which as been described as `does_not_return` returns.
- `-xatomic` specifies which atomics support runtime library is linked.

♦ ♦ ♦ C H A P T E R 3

C Compiler

This chapter describes changes to the C compiler for the Oracle Developer Studio 12.5 release.

Moving From the `-xc99` Flag to `-std` Flags

With the introduction of support for C11, the language dialect is no longer a simple binary choice between C89 & C99, there is now a third choice: C11.

With Oracle Solaris Studio 12.3 the choice was C99 vs C89 and was controlled by the `-xc99` flag:

Flag	Language Dialect
`-xc99=all`	C99 language
`-xc99=none`	C89 language

With Oracle Solaris Studio 12.4 and Oracle Developer Studio 12.5 the choice of language dialect (C89, C99, or C11) should be controlled using the `-std` flag:

Flag	Language Dialect
`-std=c11`	C11 language
`-std=c99`	C99 language
`-std=c89`	C89/C90 language

The following is a simple mapping between `-xc99` flag and the `-std` flag:

`-xc99` Flag	`-std` Flag
`-xc99`	`-std=c99`

`-xc99` Flag	`-std` Flag
`-xc99=all`	`-std=c99`
`-xc99=all,lib`	`-std=c99`
`-xc99=all,no_lib`	`-std=c99 -xlang=c89`
`-xc99=none,lib`	`-std=c89 -xlang=c99`
`-xc99=none,no_lib`	`-std=c89`
`-xc99none`	`-std=c89`

The following options for controlling language dialec are or will soon be deprecated in the Oracle Developer Studio C Compiler: `-Xc`, `-Xa`, `-Xt`, `-xc99`.

- `-xc99`: Choose between ISO C99 or C89 language
- `-Xc`: Issue errors and warnings for programs that use non-ISO C constructs
- `-Xa`: Accept ISO C plus extensions to the C language
- `-Xt`: Accept ISO C plus K&R C compatibility extensions
- `-Xs`: Accept K&R C

Instead, the `-std` option should be used, along with the `-pedantic` option if you were using `-Xa`. Legacy code using `-Xt` or `-Xs` will need conversion to an ISO C dialect.

The `-xlang` option can be used to control the behavior of specific libc functions that relate to standard conformance. In the Oracle Solaris Studio 12.4 C compiler, the default behavior was C11 language constructs and C89 library behavior. In this default mode, `___STDC_VERSION__` (199409L) indicates the C89 standard.

In the Oracle Developer Studio 12.5 C Compiler, both the language features and library behavior defaults to C11 mode, and `__STDC_VERSION__` (201112L) reflects C11.

Note: In Oracle Solaris Studio 12.3, this behavior was controlled by the suboption `-xc99=lib`.

New Default for the C Compiler on Oracle Solaris

The following features are new defaults for the C compiler on Oracle Solaris:

- The default mode for the C compiler has changed
- The new default mode might affect your application
- The old mode is available, if needed

`__STDC_VERSION__` Changes

The following changes describe what is new with `__STDC_VERSION__`:

- By default, previous versions of the C compiler on Oracle Solaris accepted all of the C99 and C11 features they knew about, but only claimed to conform to C89 by predefining `__STDC_VERSION__` to 199409L.
- By default, the new C compiler predefines `__STDC_VERSION__` to 201112L, claiming C11 compliance.

Effect on User Applications on Oracle Solaris

The following information describes the effect on user applications on the Oracle Solaris platform:

- Included files and feature tests that use macros such as `_XOPEN_SOURCE`, `_POSIX_SOURCE`, and `_POSIX_C_SOURCE` will likely resolve differently.

  For example, if you use `_POSIX_SOURCE`, this error is likely:

  ```
  Compiler or options invalid for pre-UNIX 03 X/Open applications
  ```

  The problem is that `_POSIX_SOURCE` technically requests a C89 compiler, as noted at the `standards(5)` man page.

  If your application tests for an older version of a standard (for example, `_POSIX_SOURCE`), consider code changes to try a newer version (in this example, `_XOPEN_SOURCE=600`)
- User applications that test `__STDC_VERSION__` will likely resolve differently

Temporary Workarounds for `-xlang=c89`

You probably do not want to explicitly select `-std=c89`, because doing so will disable C99 and C11 features.

Instead, if you select `-xlang=c89`, that will enable the new C compiler to accept the same programs and define the same `__STDC_VERSION` as previous versions of the C compiler.

Other Changes to the C Compiler

The C compiler changes include the changes that are described in "New and Changed Features Common to the Compilers" on page 47, and the following additional changes.

- Many SIMD (`__m128{d|i}`) intrinsic functions are provided for SPARCACE and SPARCACE+.
- New compiler options:
  - `-xcheck=noreturn` informs the compiler to add code to cause a runtime error if a routine which has been described as "do not return" returns.
  - `-xatomic` specifies which atomics support runtime library is linked.
- Support for the following C11 features:
  - Atomic objects

    Note - Using the atomics feature requires special attention to runtime support. For more information, see "Bundled Atomics Library" in *Oracle Developer Studio 12.5: C User's Guide*.

  - Type-generic expressions (`_Generic`)

See the `cc`(1) man page and the *Oracle Developer Studio 12.5: C User's Guide* for more information.

♦ ♦ ♦ CHAPTER 4

Code Analysis Tools

The code analysis tool suite ensures application reliability and stability by detecting common coding errors, including memory leaks and access violations, enabling developers to write better code with fewer errors faster.

This chapter describes the new and changed features in the code analysis tools in this Oracle Developer Studio release and contains the following sections:

- "About the Code Analysis Tools" on page 21
- "New `codean` Features" on page 21
- "New `discover` Features" on page 22
- "New `uncover` Features" on page 23

About the Code Analysis Tools

The code analysis tools help you make your application more reliable by using static, dynamic, and code coverage analysis to detect many common coding errors, including memory leaks and memory access violations. Previse performs static analysis at compilation and `discover` performs dynamic analysis at application runtime to identify code quality issues. The `uncover` tool analyzes code coverage data to provide information about functions that are not covered by your test suite and tells how you can benefit by covering those functions.

Use the Code Analyzer graphics tool or the `codean` command-line utility to view the three types of analysis to get a comprehensive view of your application's vulnerabilities so you can improve its correctness and reliability.

New `codean` Features

The following features were added to the Code Analyzer command-line tool `codean`.

- **New Labels feature:**
  - Reported issues from Previse, `discover`, and `uncover` can now be labeled with `false_positive`, `wont_be_fixed`, or `verified`. Labeling is useful in managing issues found by the code analysis tools.
  - Issues can be suppressed using labels.
- **New test suites:**
  - Reports from `discover` that are saved with the `-a` option can be accumulated by `codean` to reflect issues found when running the application through a test suite
  - The summary page shows what percentage of functions are uncovered by a test suite

For more information about Code Analyzer in general, see the Help in Code Analyzer, the *Oracle Developer Studio 12.5: Code Analyzer User's Guide*, *Oracle Developer Studio 12.5: Code Analyzer Tutorial*, the `codean`(1) man page, and the `code-analyzer`(1) man page.

New `discover` Features

The following features were added in the `discover` memory analysis tool in this release. For more information, see the `discover`(1) man page and *Oracle Developer Studio 12.5: Discover and Uncover User's Guide*.

- **Hardware-Assisted Checking Using Application Data Integrity (ADI)** — This provides faster memory access checking on SPARC M7 platforms. This feature was introduced in the Oracle Solaris Studio 12.4 04/15 PSE. For more information, see "Hardware-Assisted Checking Using Silicon Secured Memory (SSM)" in *Oracle Developer Studio 12.5: Discover and Uncover User's Guide*.
- **New ADI helper library** — The new library is for use of ADI functionality for programs that do not use standard `malloc()` and `free()` calls for memory management. Users should use the ADI APIs documented in "Custom Memory Allocators and the discover ADI Library" in *Oracle Developer Studio 12.5: Discover and Uncover User's Guide* and link with the `libadihelpder.so` library. For more information, see "Oracle Developer Studio Code Security Check — Discover ADI" in *Oracle Developer Studio 12.5: Overview* and the `libadiplugin`(3) man page.
- **Reduced False positives of UMR and PIR** — The common causes for the false positives were presence of non-Oracle Developer Studio built libraries and system calls that the `discover` tool was not aware of.
- **SIGCHLD signal handle improvement** — Binary instrumented with `discover` no longer fails if the program installs a `SIGCHLD` signal handle.
- **Less memory usage** — Binaries instrumented with `discover` use less memory.

- **Interactive debugging improvement** — Improvement of interactive debugging of binary instrumented with `discover` when using `dbx`.
- **Address Space Layout Randomization handling** —The **discover** utility can handle applications for which Address Space Layout Randomization (ASLR) is enabled.

New `uncover` Features

The following features were added to the Uncover code coverage tool in this release.

- **Improved instrumentation time and memory consumption.**

For more information, see the `uncover`(1) man page and the *Oracle Developer Studio 12.5: Discover and Uncover User's Guide*.

♦ ♦ ♦ C H A P T E R 5

Performance Analysis Tools

The performance analysis tools work together to enable you to analyze your application's behavior and find trouble spots that impact performance.

This chapter describes the new and changed features in the performance analysis tools in this Oracle Developer Studio release.

About Performance Analyzer

Performance Analyzer provides insight into the behavior of your application to enable you to find problem areas in your code. Performance Analyzer identifies which functions, code segments, and source lines are using the most system resources. Performance Analyzer can profile single-threaded, multithreaded, and multi-process applications, then present the profiling data to help you identify where you can improve your application's performance.

The Performance Analyzer consists of a set of commands and tools including: the collect utility, which gathers profiling data on user-level programs; the `er_kernel` utility, which gathers profiling data on the Oracle Solaris kernel; the `er_print` utility which presents profiling information in text form; and the Performance Analyzer GUI, which presents profiling information graphically.

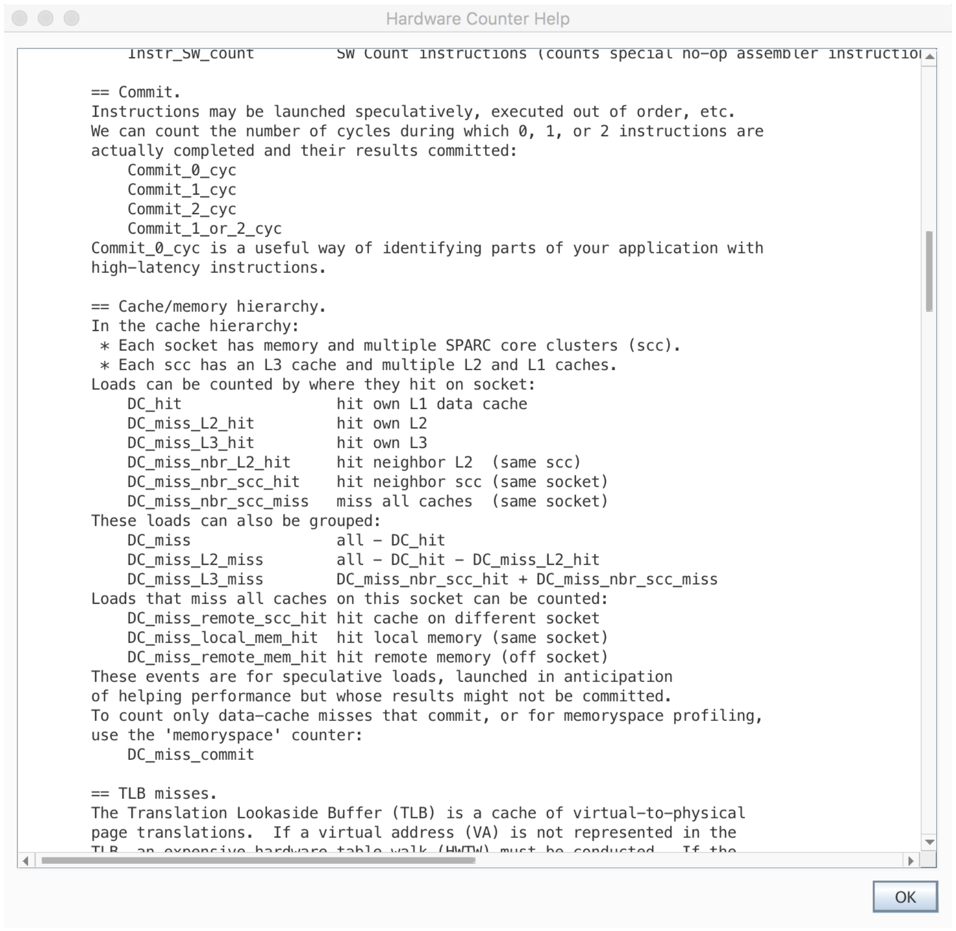
Thread Analyzer is a related tool that enables you to focus on multithreading problems.

For more information, please refer to *Oracle Developer Studio 12.5: Performance Analyzer* and *Oracle Developer Studio 12.5: Performance Analyzer Tutorials*.

Performance Analyzer New Features

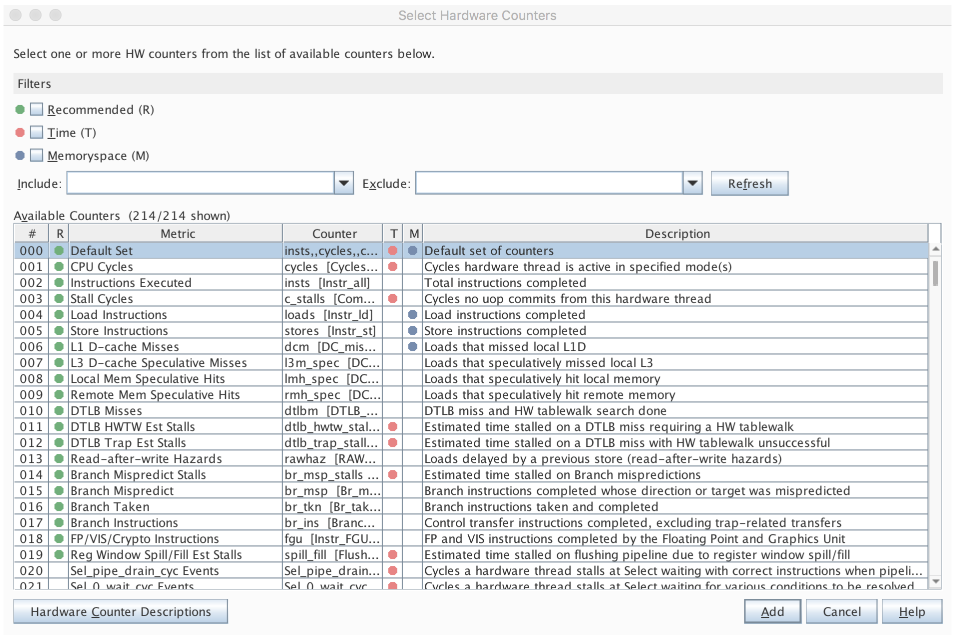
This section summarizes the new features in this release of the Performance Analyzer and related tools. For more information, see the Help in Performance Analyzer.

- **Simplified Hardware Counter Profiling**
  - New, processor-specific help provides overviews of key hardware counters (SPARC only)



```
    Instr_Sw_count         SW Count instructions (counts special no-op assembler instructio

== Commit.
Instructions may be launched speculatively, executed out of order, etc.
We can count the number of cycles during which 0, 1, or 2 instructions are
actually completed and their results committed:
    Commit_0_cyc
    Commit_1_cyc
    Commit_2_cyc
    Commit_1_or_2_cyc
Commit_0_cyc is a useful way of identifying parts of your application with
high-latency instructions.

== Cache/memory hierarchy.
In the cache hierarchy:
 * Each socket has memory and multiple SPARC core clusters (scc).
 * Each scc has an L3 cache and multiple L2 and L1 caches.
Loads can be counted by where they hit on socket:
    DC_hit                 hit own L1 data cache
    DC_miss_L2_hit         hit own L2
    DC_miss_L3_hit         hit own L3
    DC_miss_nbr_L2_hit     hit neighbor L2  (same scc)
    DC_miss_nbr_scc_hit    hit neighbor scc (same socket)
    DC_miss_nbr_scc_miss   miss all caches  (same socket)
These loads can also be grouped:
    DC_miss                all - DC_hit
    DC_miss_L2_miss        all - DC_hit - DC_miss_L2_hit
    DC_miss_L3_miss        DC_miss_nbr_scc_hit + DC_miss_nbr_scc_miss
Loads that miss all caches on this socket can be counted:
    DC_miss_remote_scc_hit hit cache on different socket
    DC_miss_local_mem_hit  hit local memory (same socket)
    DC_miss_remote_mem_hit hit remote memory (off socket)
These events are for speculative loads, launched in anticipation
of helping performance but whose results might not be committed.
To count only data-cache misses that commit, or for memoryspace profiling,
use the 'memoryspace' counter:
    DC_miss_commit

== TLB misses.
The Translation Lookaside Buffer (TLB) is a cache of virtual-to-physical
page translations.  If a virtual address (VA) is not represented in the
TLB, an expensive hardware table walk (HWTW) must be conducted.  If the
```

  - New dialog box Select Hardware Counters describes counters in more detail. Additionally, it enables you to filter counters and to add counters more simply.



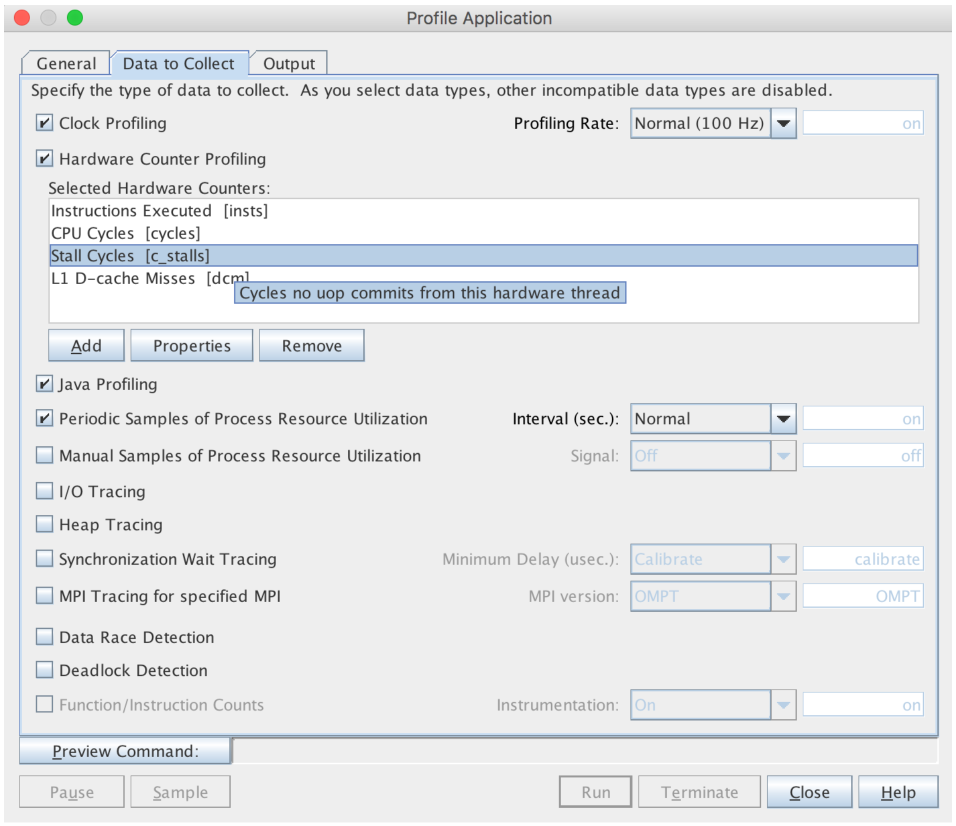
Select one or more HW counters from the list of available counters below.

● Recommended (R)
● Time (T)
● Memoryspace (M)

Available Counters (214/214 shown)

#	R	Metric	Counter	T	M	Description
000	●	Default Set	insts,,cycles,,c...	●	●	Default set of counters
001	●	CPU Cycles	cycles [Cycles...	●		Cycles hardware thread is active in specified mode(s)
002	●	Instructions Executed	insts [Instr_all]			Total instructions completed
003	●	Stall Cycles	c_stalls [Com...	●		Cycles no uop commits from this hardware thread
004	●	Load Instructions	loads [Instr_ld]		●	Load instructions completed
005	●	Store Instructions	stores [Instr_st]		●	Store instructions completed
006	●	L1 D-cache Misses	dcm [DC_mis...		●	Loads that missed local L1D
007	●	L3 D-cache Speculative Misses	l3m_spec [DC...			Loads that speculatively missed local L3
008	●	Local Mem Speculative Hits	lmh_spec [DC...			Loads that speculatively hit local memory
009	●	Remote Mem Speculative Hits	rmh_spec [DC...			Loads that speculatively hit remote memory
010	●	DTLB Misses	dtlbm [DTLB_...			DTLB miss and HW tablewalk search done
011	●	DTLB HWTW Est Stalls	dtlb_hwtw_stal...	●		Estimated time stalled on a DTLB miss requiring a HW tablewalk
012	●	DTLB Trap Est Stalls	dtlb_trap_stall...	●		Estimated time stalled on a DTLB miss with HW tablewalk unsuccessful
013	●	Read-after-write Hazards	rawhaz [RAW...			Loads delayed by a previous store (read-after-write hazards)
014	●	Branch Mispredict Stalls	br_msp_stalls ...	●		Estimated time stalled on Branch mispredictions
015	●	Branch Mispredict	br_msp [Br_m...			Branch instructions completed whose direction or target was mispredicted
016	●	Branch Taken	br_tkn [Br_tak...			Branch instructions taken and completed
017	●	Branch Instructions	br_ins [Branc...			Control transfer instructions completed, excluding trap-related transfers
018	●	FP/VIS/Crypto Instructions	fgu [Instr_FGU...			FP and VIS instructions completed by the Floating Point and Graphics Unit
019	●	Reg Window Spill/Fill Est Stalls	spill_fill [Flush...	●		Estimated time stalled on flushing pipeline due to register window spill/fill
020		Sel_pipe_drain_cyc Events	Sel_pipe_drain...	●		Cycles a hardware thread stalls at Select waiting with correct instructions when pipeli...
021		Sel_0_wait_cyc Events	Sel_0_wait_cyc...	●		Cycles a hardware thread stalls at Select waiting for various conditions to be resolved

- New `auto` option to automatically select appropriate profile rates.
- Updated default hardware counter sets for supported hardware.
- Updated views for memoryspace profiling for supported platforms. (On x86 systems, memoryspace profiling requires at least Oracle Solaris 11.3)
- Simplified workflow for selecting hardware counters


Profile Application

Specify the type of data to collect. As you select data types, other incompatible data types are disabled.

☑ Clock Profiling — Profiling Rate: Normal (100 Hz) — on

☑ Hardware Counter Profiling

Selected Hardware Counters:

```
Instructions Executed  [insts]
CPU Cycles  [cycles]
Stall Cycles  [c_stalls]
L1 D-cache Misses  [dcm]
```

Cycles no uop commits from this hardware thread

☑ Java Profiling

☑ Periodic Samples of Process Resource Utilization — Interval (sec.): Normal — on

☐ Manual Samples of Process Resource Utilization — Signal: Off — off

☐ I/O Tracing

☐ Heap Tracing

☐ Synchronization Wait Tracing — Minimum Delay (usec.): Calibrate — calibrate

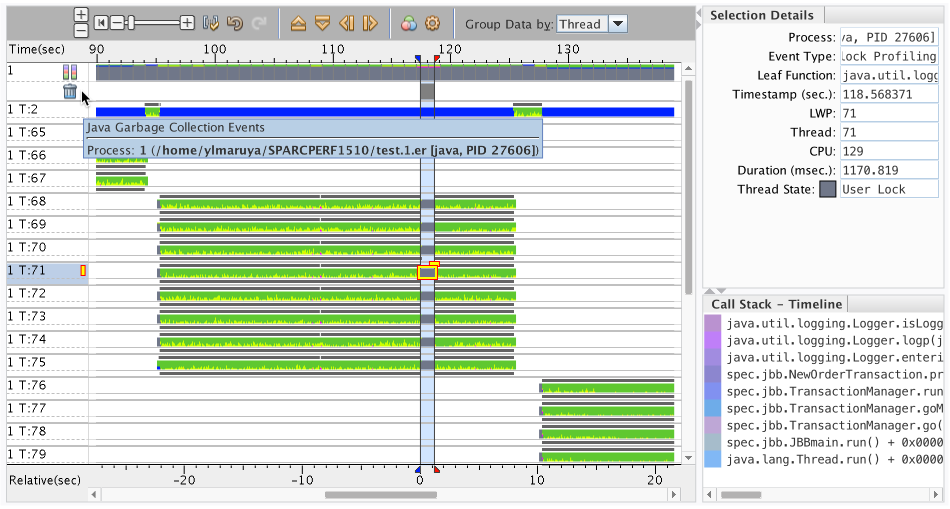
☐ MPI Tracing for specified MPI — MPI version: OMPT — OMPT

☐ Data Race Detection

☐ Deadlock Detection

☐ Function/Instruction Counts — Instrumentation: On — on

- **Java Profiling Enhancements**
  - Information about Java garbage collector events is now shown on the Overview and in Timeline.



Java Garbage Collection Events

Process: 1 (/home/ylmaruya/SPARCPERF1510/test.1.er [java, PID 27606])

Process:	va, PID 27606]
Event Type:	.ock Profiling
Leaf Function:	java.util.logg
Timestamp (sec.):	118.568371
LWP:	71
Thread:	71
CPU:	129
Duration (msec.):	1170.819
Thread State:	User Lock

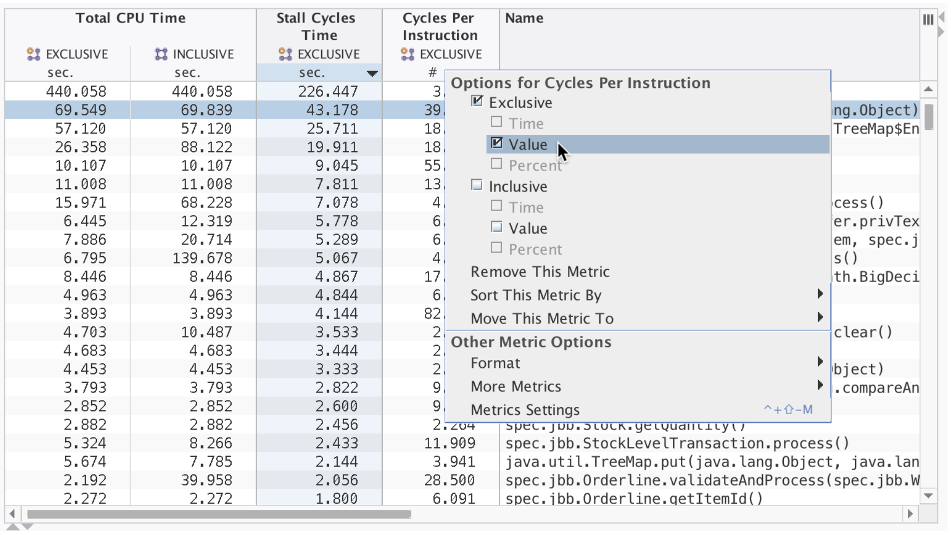
```
java.util.logging.Logger.isLogg
java.util.logging.Logger.logp(j
java.util.logging.Logger.enteri
spec.jbb.NewOrderTransaction.pr
spec.jbb.TransactionManager.run
spec.jbb.TransactionManager.goM
spec.jbb.TransactionManager.go(
spec.jbb.JBBmain.run() + 0x0000
java.lang.Thread.run() + 0x0000
```

- Improved attribution of performance metrics to functions, source, bytecode, and machine code.

- **Metric Presentation Enhancements**

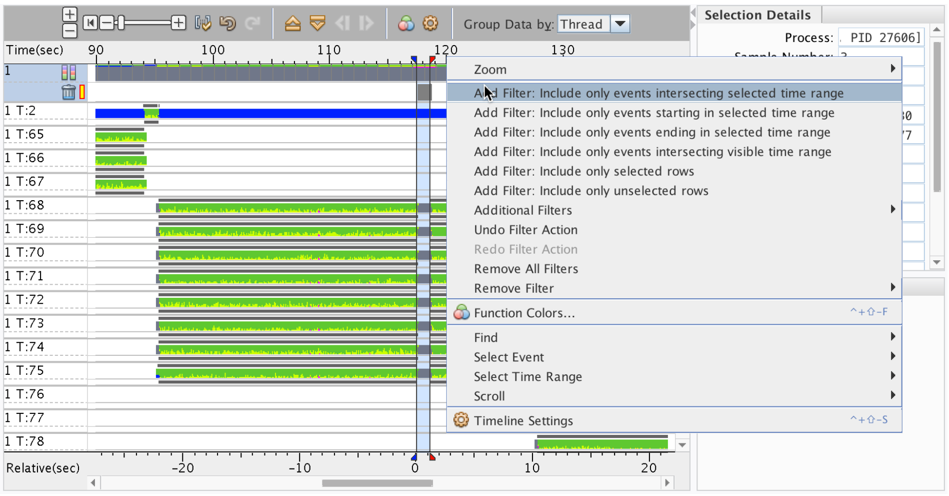
  Most data views have improved presentation of metrics and enable you to more conveniently control how data is displayed:

  - Data columns are grouped to better show inclusive and exclusive metrics.
  - Column headings provide controls in the top corners that you can click to configure the metrics or delete them altogether from the view.
  - The controls for selecting time or percent for displaying metrics that were previously located in the Overview screen are now available in the column heading controls.



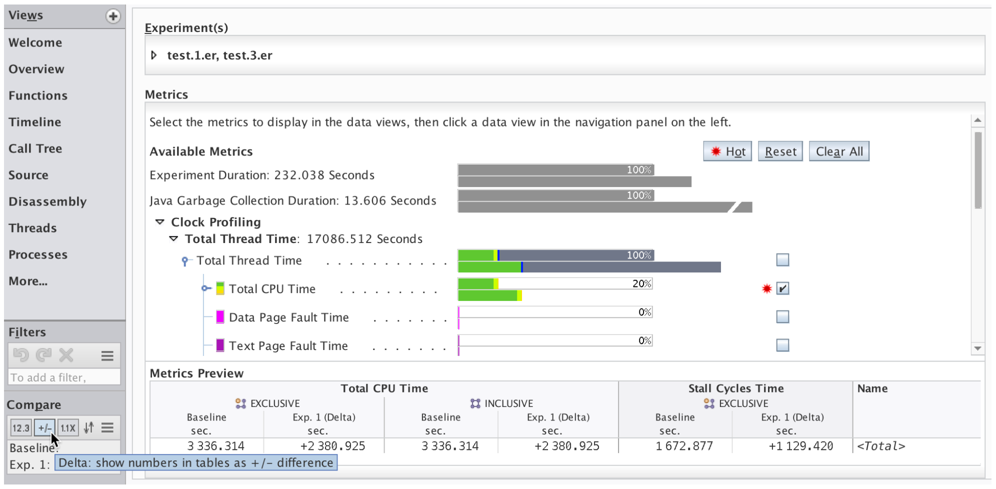
Total CPU Time		Stall Cycles Time	Cycles Per Instruction	Name
EXCLUSIVE	INCLUSIVE	EXCLUSIVE	EXCLUSIVE	
sec.	sec.	sec.	#	
440.058	440.058	226.447	3	
69.549	69.839	43.178	39	ng.Object)
57.120	57.120	25.711	18	TreeMap$En
26.358	88.122	19.911	18	
10.107	10.107	9.045	55	
11.008	11.008	7.811	13	
15.971	68.228	7.078	4	cess()
6.445	12.319	5.778	6	er.privTex
7.886	20.714	5.289	6	em, spec.j
6.795	139.678	5.067	4	s()
8.446	8.446	4.867	17	th.BigDeci
4.963	4.963	4.844	6	
3.893	3.893	4.144	82	
4.703	10.487	3.533	2	clear()
4.683	4.683	3.444	2	
4.453	4.453	3.333	2	bject)
3.793	3.793	2.822	9	.compareAn
2.852	2.852	2.600	9	
2.882	2.882	2.456	2.264	spec.jbb.Stock.getQuantity()
5.324	8.266	2.433	11.909	spec.jbb.StockLevelTransaction.process()
5.674	7.785	2.144	3.941	java.util.TreeMap.put(java.lang.Object, java.lan
2.192	39.958	2.056	28.500	spec.jbb.Orderline.validateAndProcess(spec.jbb.W
2.272	2.272	1.800	6.091	spec.jbb.Orderline.getItemId()

- **Timeline Enhancements**
  - Selection of a time range can now be made from the rulers.
  - Filtering by time or row is now accessible from context menus in the rulers.


Group Data by: Thread
Selection Details
Process: PID 27606]
Time(sec) 90 100 110 120 130
1
1 T:2
1 T:65
1 T:66
1 T:67
1 T:68
1 T:69
1 T:70
1 T:71
1 T:72
1 T:73
1 T:74
1 T:75
1 T:76
1 T:77
1 T:78
Relative(sec) -20 -10 0 10 20
Zoom
Add Filter: Include only events intersecting selected time range
Add Filter: Include only events starting in selected time range
Add Filter: Include only events ending in selected time range
Add Filter: Include only events intersecting visible time range
Add Filter: Include only selected rows
Add Filter: Include only unselected rows
Additional Filters
Undo Filter Action
Redo Filter Action
Remove All Filters
Remove Filter
Function Colors... ^+⇧-F
Find
Select Event
Select Time Range
Scroll
Timeline Settings ^+⇧-S

- **Comparing Experiments Enhancements**

A new Compare panel on the Performance Analyzer main window enables you to switch between viewing the compared data in absolute, delta, and ratio modes using buttons.



Experiment(s)

test.1.er, test.3.er

Metrics

Select the metrics to display in the data views, then click a data view in the navigation panel on the left.

Available Metrics

Experiment Duration: 232.038 Seconds 100%

Java Garbage Collection Duration: 13.606 Seconds 100%

Clock Profiling

Total Thread Time: 17086.512 Seconds

Total Thread Time 100%

Total CPU Time 20% ✓

Data Page Fault Time 0%

Text Page Fault Time 0%

Metrics Preview

Total CPU Time				Stall Cycles Time		Name
EXCLUSIVE		INCLUSIVE		EXCLUSIVE		
Baseline sec.	Exp. 1 (Delta) sec.	Baseline sec.	Exp. 1 (Delta) sec.	Baseline sec.	Exp. 1 (Delta) sec.	
3 336.314	+2 380.925	3 336.314	+2 380.925	1 672.877	+1 129.420	*<Total>*

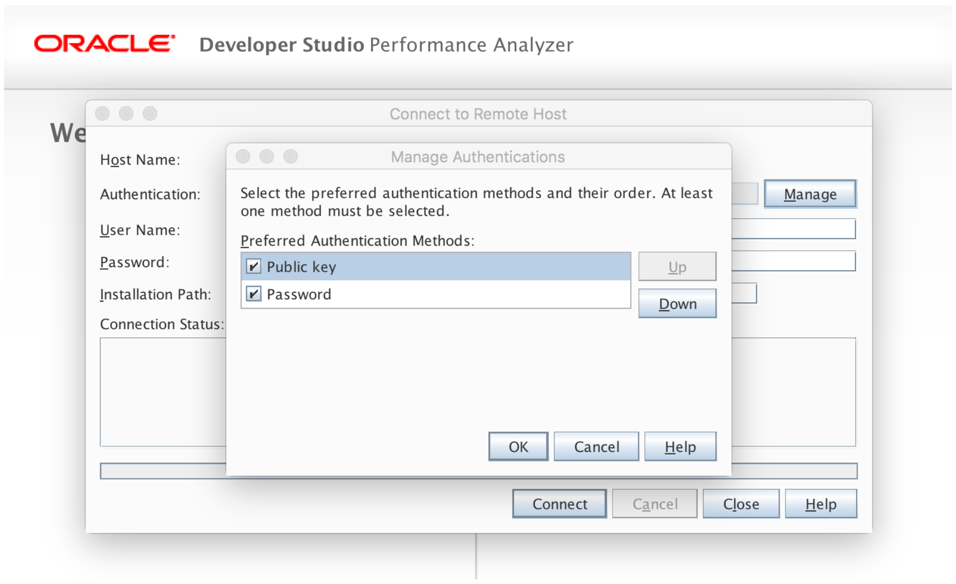
See Comparing experiments for more information.

- **Call Tree Enhancements**

  Call Tree view has new Copy All option that copies the calltree in text form, which you can paste into a text file.

- **Remote Analyzer Enhancements**

  - The Remote Analyzer now supports multiple authentication methods.



Select the preferred authentication methods and their order. At least one method must be selected.

Preferred Authentication Methods:

- [✓] Public key
- [✓] Password

- Remote Analyzer provides better error messages, and has improved performance.

Changes to Command-Line Tools

This section describes changes made to various command-line performance analysis tools. For more information, see the corresponding man pages for each command-line tool.

Changes to Data Collection Tools

Data collection tools include the `collect` command, `dbx collector` command, and `er_kernel` command. Each of these tools is used to profile programs to collect data and create experiments that can be read by Performance Analyzer or `er_print`. All data collection tools have the following changes.

- Tracing of Java garbage collection is automatically done for Java experiments.
- For Java, the synctrace feature has been extended with an optional `<scope>` modifier; `<scope>` can be `n` for native API tracing, `j` for Java API tracing, or `nj` for tracing both APIs. `nj` is the default.

- Hardware counters on Oracle SPARC and x86 (including Haswell-E/EP) have updated support and memoryspace views.
- New hardware counter option `-auto` helps select appropriate profile rates.
- New guidance on using SPARC hardware counters is available using `collect -h` and `er_kernel-h`.
- Issues concerning `dlopen()`, `dlmopen()`, `dlclose()`, `exit()` and `Exit()` are fixed.
- The threads created on Linux when `CLONE_VM` is specified will not be followed.
- The limit on the maximum number of threads has been removed.

`collect` Utility Changes

The `collect` utility is a tool you use to profile your application as it runs to collect data and create an experiment that can be read by Performance Analyzer or `er_print`.

In addition to the changes common to all data collection tools, the `collect` utility is changed in this release as follows:

- Heap tracing on Java targets will be allowed, but it will only trace native allocations, not Java allocations
- The `-R` argument is no longer recognized.

`dbx collector` Changes

The `dbx collector` is a subcommand of the `dbx` debugger that you can use for performance data collection. See the `collector`(1) man page for more information.

In addition to the changes common to all data collection tools, the `dbx collector` command is changed in this release as follows:

- Several bugs in the 12.4 version have been fixed.

`er_kernel` Utility Changes

The `er_kernel` command profiles the Oracle Solaris kernel and generates an experiment that you can examine in Performance Analyzer or `er_print`.

In addition to the changes common to all data collection tools, the `er_kernel` utility is changed as follows:

- Process creation and termination are more accurately tracked.
- The formatting of the `er_kernel -h` output has been improved.

See the `er_kernel`(1) man page for more information.

`er_print` Utility Changes

The `er_print` utility generates a plain-text version of the data views presented by the Performance Analyzer. The output is displayed on the standard output.

The `er_print` utility is changed in this release as follows:

The `er_print` utility is changed as follows:

- Several bugs in the 12.4 version have been fixed.
- The flags used when compiling with Oracle Developer Studio are shown in the source and disassembly reports.
- Machine model information is reported in the experiment header.
- The `overview` command has been implemented.

See the `er_print`(1)man page for more information.

Changes to Other Commands and APIs

The `libcollector` API has the following updates:

- Static versions of `libcollectorAPI.a` and `libfcollector.a` are now available.
- The man page has been rewritten to better describe the Java API. See `libcollector`(3).

♦ ♦ ♦ CHAPTER 6

Debugging Tools

Oracle Developer Studio provides the command-line `dbx` debugger, and the `dbxtool` graphical tool for using `dbx`. The debugger is also integrated into the IDE. For more information about debugging with the IDE, see Chapter 7, "Oracle Developer Studio IDE".

This chapter contains the following topics about what's new in the debugging tools:

- "About the `dbx` Debugger" on page 35
- "New and Changed `dbx` Features" on page 35

About the `dbx` Debugger

The `dbx` debugger is an interactive, source-level, postmortem and real-time debugging tool. You can use it at the command line, through the `dbxtool` graphical interface, and in the Oracle Developer Studio IDE. The `dbx` debugger is scriptable and multithread-aware.

New and Changed `dbx` Features

The following features were added or changed in `dbx` . For more information, see *Oracle Developer Studio 12.5: Debugging a Program with dbx*, the `dbx`(1) man page and the `dbx` help file. To access the `dbx` help file, type the following:

```
% dbx
(dbx) help
```

- `dbx` version updated to 8.1 from 8.0.
- Support for compressed debug information. See "Compressed Debug Sections (Oracle Solaris Only)" in *Oracle Developer Studio 12.5: Debugging a Program with dbx*.
- New option `-a` for the `whatis` command. The `-a` option prints data members only.

- New dbx environment variable `output_data_member_only`. If set to `on`, prints data members only.
- Support for handling Position Independent Executables (PIE) added.
- Support added for Secured Silicon Memory (SSM) on Oracle Solaris 11 on SPARC. Use the `dbx` command `adi` to use this feature.
- Support for the following C++11 features: user defined literals.
- Support for the following C++14 features:binary literals; single-quote digit separator.
- Support for the following C11 features: type generic expressions.
- Change in the following register names, to be more consistent across Intel platforms:

Previous Name	**New Name**
`fs_base`	`fsbase`
`gs_base`	`gsbase`
`fcwd`	`fcw`
`fswd`	`fsw`
`ftw`	`fctw`
`mxcr_mask`	Removed
`uesp`	Removed

For more information, issue a `help changes` command under `dbx`, to access the `dbx` help file.

♦ ♦ ♦ CHAPTER 7

Oracle Developer Studio IDE

Oracle Developer Studio IDE integrates many of the components of Oracle Developer Studio for users who prefer a graphical programming environment.

The following topics are covered in this chapter:

About Oracle Developer Studio IDE

Oracle Developer Studio offers a graphical integrated development environment (IDE) that is built on the NetBeans platform and is configured to use the Oracle Developer Studio C, C++, and Fortran compilers, the `dmake` distributed make command, and `dbx` debugger. The IDE also integrates with some of the Analyzer tools of the analysis suite so you can analyze your code without leaving the IDE.

The command to start the IDE is `devstudio`. For details about this command, see the `devstudio`(1) man page.

For complete documentation of the IDE, see the Help in the IDE. For step-by-step instructions to use the basic features of the IDE, see the *Oracle Developer Studio 12.5: IDE Quick Start Tutorial*.

New and Changed IDE Features

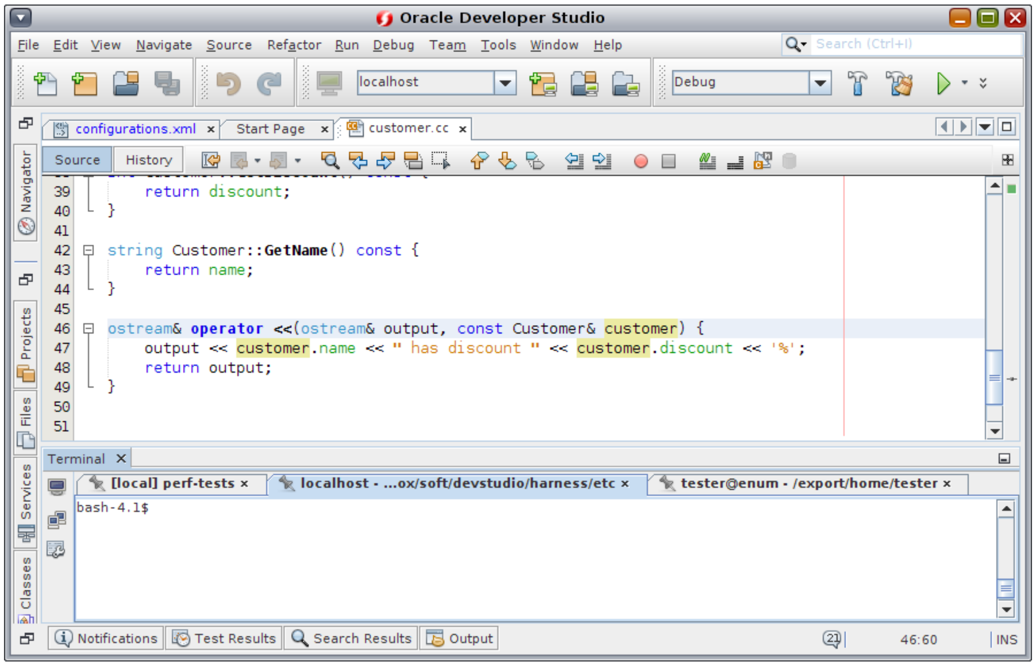
The following features were added or changed in Oracle Developer Studio IDE:

- **New Pinnable Terminal** - For more information, see "Pinnable Terminal" on page 38.
- **New Project with Existing Sources wizard now makes pre-build and build steps more clear and understandable.** - For more information, see "New Project Wizard" on page 39.
- **Remote Development has been improved.** - For more information, see "Improvements to Remote Development" on page 40.
- **Mixed Development (Java and C/C++) support.** - For more information, see "Mixed Development (Java and C/C++) Support" on page 41.
- **Edit Properties of multiple files** - For more information, see "Edit Properties of Multiple Files" on page 42.
- **Support Doxygen C++ comments** - For more information, see "Support Doxygen C++ Comments" on page 43.
- **Code Folding for compound statements** - For more information, see "Code Folding for Compound Statements" on page 43.
- **Navigating from compiler hint in editor to output log** - For more information, see the IDE help topic "Navigate to Build Log from a Compiler Hint".
- **SendTo Utility** - For more information, see the IDE help topic "SendTo Utility".
- **Generate missing switch clauses**
- **Call Graph enhancements** - For more information, see the IDE help topic "Using the CallGraph".
- **New Audits and Hints - Including Security Coding suggestions** - In the IDE, select Options → Editor → Hints and select the C/C++ language to see the many new hints available. For more information, see the IDE help topic "Using the Static Code Analyzer".
- **New Refactorings** - For more information, see the IDE help topic "Introducing Functions and Variables by Refactoring".

Pinnable Terminal

The "Pin Tab" action was added to the Terminal popup menu.

FIGURE 2 Pinnable Terminal



```
        return discount;
    }

    string Customer::GetName() const {
        return name;
    }

    ostream& operator <<(ostream& output, const Customer& customer) {
        output << customer.name << " has discount " << customer.discount << '%';
        return output;
    }
```

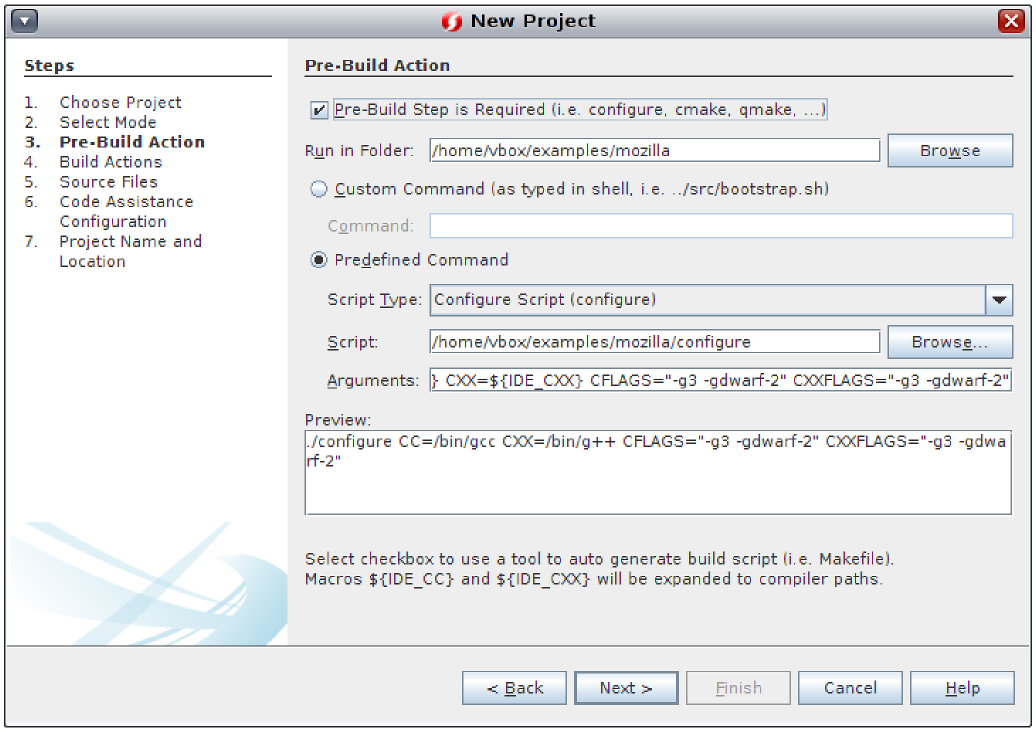
```
bash-4.1$
```

For more information, see the IDE help topic "Pinning a Terminal".

New Project Wizard

In the New Project with Existing Sources wizard, the Pre-Build step can be customized later with the new property tab "Pre-Build". You can access this tab from Project Properties → Build → Pre-Build. The following figure shows the New Project Wizard, in the Pre-Build Action tab:

FIGURE 3 New Project Wizard



[✓] Pre-Build Step is Required (i.e. configure, cmake, qmake, ...)

Run in Folder: /home/vbox/examples/mozilla

○ Custom Command (as typed in shell, i.e. ../src/bootstrap.sh)

Command:

● Predefined Command

Script Type: Configure Script (configure)

Script: /home/vbox/examples/mozilla/configure

Arguments: } CXX=${IDE_CXX} CFLAGS="-g3 -gdwarf-2" CXXFLAGS="-g3 -gdwarf-2"

Preview:

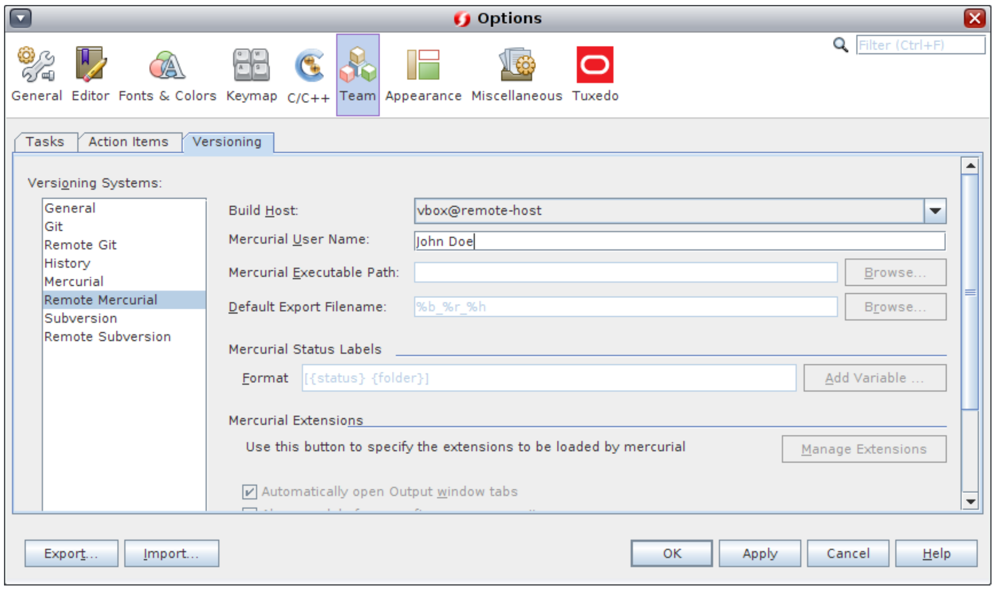
```
./configure CC=/bin/gcc CXX=/bin/g++ CFLAGS="-g3 -gdwarf-2" CXXFLAGS="-g3 -gdwa
rf-2"
```

Select checkbox to use a tool to auto generate build script (i.e. Makefile).
Macros ${IDE_CC} and ${IDE_CXX} will be expanded to compiler paths.

Improvements to Remote Development

The following features were added to remote development:

- SVN, Git and Mercurial support in Remote mode - This feature enables you to use VCS in full remote mode as a local user. You can customize remote VCS preferences in Tools → Options → Team → Remote Mercurial/Git/Subversion and in Tools → Options → Fonts & Colors → Remote Mercurial/GIT/Subversion. The following screenshot shows the Versioning options in the Options window:



Versioning Systems:
General
Git
Remote Git
History
Mercurial
Remote Mercurial
Subversion
Remote Subversion

Build Host: vbox@remote-host
Mercurial User Name: John Doe
Mercurial Executable Path:
Default Export Filename: %b_%r_%h

Mercurial Status Labels
Format [{status} {folder}]

Mercurial Extensions
Use this button to specify the extensions to be loaded by mercurial

☑ Automatically open Output window tabs

- Code Assistance support for Full Remote projects
- New options for New Remote Host Setup wizard

For more information, see the IDE help topics "Using Full Remote Mode with Version Control Systems" and "Configuring a Remote Host for C/C++/Fortran Development".

Mixed Development (Java and C/C++) Support

The IDE can help you integrate C/C++ projects with Java projects that use Java Native Interface (JNI) and Java Native Access (JNA) technologies. These are technologies that enable you to call native C and C++ programs from Java programs.

If you have a Java project that uses JNI methods, you can generate a C++ JNI project from the Java project. The JNI project implements all the native interfaces in Java code. See the IDE help topic "Working with JNI in Mixed Development Projects" and "Working With JNA in Mixed Development" for information on how to create a C++ JNI Library project and which IDE features are supported for the associated Java and native projects.

In order to use mixed development, you will need to perform the following prerequisite steps.

1. **Check to make sure you have a Java JDK and run IDE with Java JDK installed on your system to use the feature.**

The IDE normally only requires a Java JRE.

2. **Install the Java SE plugin from the IDE update center.**

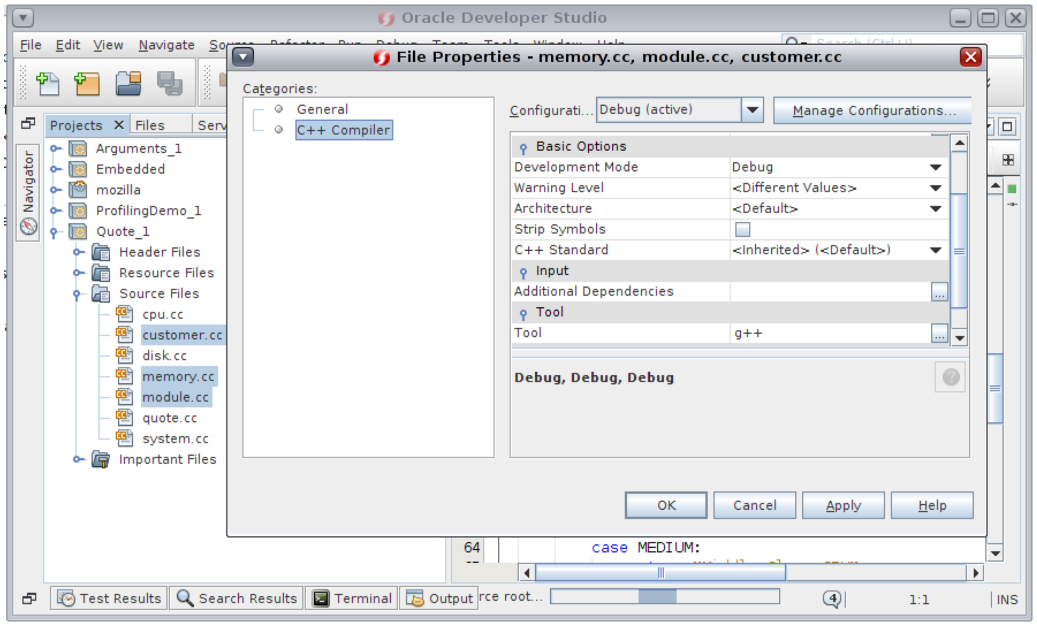
   There are five modules that should be installed.

3. **Take note of your user directory and be aware of the following:**
   - All required additional modules will be installed with a specified user directory.
   - Mixed development will only work if you run the IDE with the user directory that has the additional modules associated with it.
   - If you do not install to the correct user directory, this feature will not work or will require a Java SE re-installation with a new user directory.

Edit Properties of Multiple Files

You can now edit the properties of multiple files within a project. Shift-click to select multiple files in the Projects window, then right-click and choose Properties. The Properties window will appear:

FIGURE 4 Edit Properties of Multiple Files



Configurati... Debug (active)

Basic Options	
Development Mode	Debug
Warning Level	<Different Values>
Architecture	<Default>
Strip Symbols	☐
C++ Standard	<Inherited> (<Default>)
Input	
Additional Dependencies	
Tool	
Tool	g++

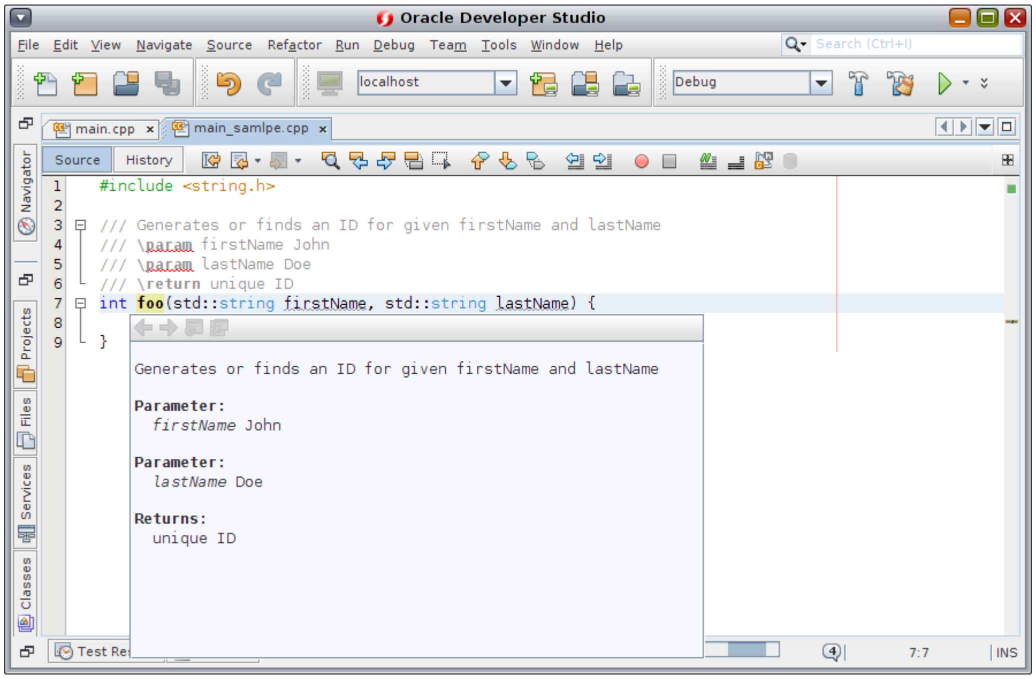
Debug, Debug, Debug

```
case MEDIUM:
```

Support Doxygen C++ Comments

Projects can now use doxygen-style single-line "///" comments for documentation.

FIGURE 5 Using Doxygen C++ Comments



```
#include <string.h>

/// Generates or finds an ID for given firstName and lastName
/// \param firstName John
/// \param lastName Doe
/// \return unique ID
int foo(std::string firstName, std::string lastName) {

}
```

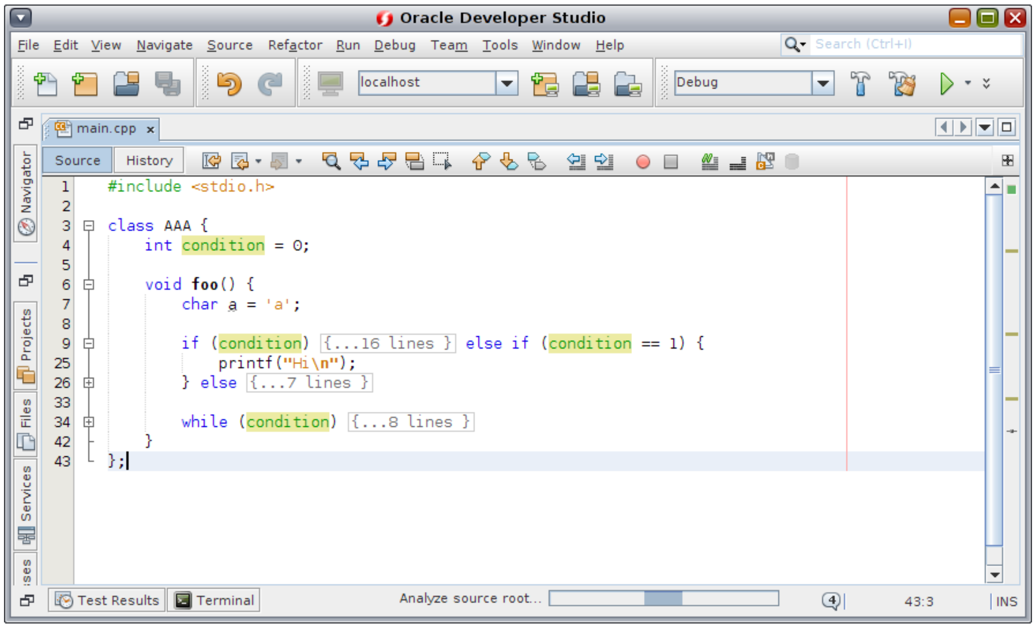
```
Generates or finds an ID for given firstName and lastName

Parameter:
  firstName John

Parameter:
  lastName Doe

Returns:
  unique ID
```

For more information, see the IDE help topic "Adding Documentation to Your Code".

Code Folding for Compound Statements

You can now fold compound statements, such as if-else, do-while, etc.

FIGURE 6 Code Folding in IDE



```
#include <stdio.h>

class AAA {
    int condition = 0;

    void foo() {
        char a = 'a';

        if (condition) {...16 lines } else if (condition == 1) {
            printf("Hi\n");
        } else {...7 lines }

        while (condition) {...8 lines }
    }
};
```

For more information, see the IDE help topic "Folding Blocks of Code in C and C++ Files".

♦ ♦ ♦ C H A P T E R 8

OpenMP API

This chapter describes the changes for OpenMP API support in this release of Oracle Developer Studio.

OpenMP

This section discusses new features and updates to the OpenMP API.

- The default number of threads for OpenMP and `autopar` programs is a multiple of the number of `cores_per_chip`. The algorithm for computing the number of threads starts with `cores_per_chip`, then it checks successive multiples of that number and picks the highest one that does not exceed 32 and does not exceed the number of cores on the machine.

For more information about OpenMP, see *Oracle Developer Studio 12.5: OpenMP API User's Guide*.

♦ ♦ ♦ CHAPTER 9

Other Changes

This chapter describes new and changed features for other components of the Oracle Developer Studio software.

- "Changes to Compilers" on page 47
- "Performance Library Changes" on page 51

Changes to Compilers

The following section describes changes made to the compilers and includes the following topics:

- "New and Changed Features Common to the Compilers" on page 47
- "Fortran Compiler" on page 48
- "New Static Analysis Features" on page 49

New and Changed Features Common to the Compilers

The following changes were made to the C, C++, and Fortran compilers since the previous release. Details can be found in the compiler man pages. The changes specific to the C++ compiler are detailed in Chapter 2, "C++ Compiler". The changes specific to the C compiler are detailed in Chapter 3, "C Compiler".

Application Performance on New Hardware

Every Oracle Developer Studio release includes performance improvements for Oracle Sun hardware servers. This release includes expanded support for the SPARC M6, SPARC M7, SPARC T7, SPARC S7 and Intel Broadwell/avx2_i processors.

Other Compiler Changes

The following list describes other changes that affect the C, C++, and Fortran compilers:

- Support for SPARC M6, M7, T7 and S7 processors.
- Support for x86 dataspace profiling.
- `-xcheck` has a new default `-xcheck=stkovfl`.
- New Compiler Options:
  - `-features=[no]mergestrings` causes the compiler to put string literals and other suitable const or read-only data into a special section of the binary where the linker removes duplicate strings.This option is available only on SPARC.
  - `-xsecure_code_analysis` enables compiler secure code analysis to find and display possible memory safety violations at compile time.
  - `-xtarget=S7`, `-xchip=S7` are supported in the compiler driver.

Fortran Compiler

The Fortran compiler supports technical and scientific application development with record-setting runtime performance and compatibility options for the Fortran77, Fortran90, and Fortran95 standards. The majority of Fortran 2003 features and OpenMP 4.0 support is included. The Fortran compiler uses the same high-performance code generation technology as the C and C++ compilers, ensuring that the resulting application generates the highest-performance parallel code for the newest SPARC and x86-based Oracle systems.

The Fortran compiler changes include the changes that are described in "New and Changed Features Common to the Compilers" on page 47 and the following changes:

- The maximum length of a free-source form line has been increased from 132 to 250 characters.

For more information, see the `f95`(1)man page and the *Oracle Developer Studio 12.5: Fortran User's Guide*.

New Static Analysis Features

Compile time warnings for memory safety checks are generated by default for the C and C++ compilers.

The following message tags are included:

- `SEC_UNINITIALIZED_MEM_READ`
- `SEC_UNINITIALIZED_BITOP`
- `SEC_UNDEFINED_RETURN_VALUE`
- `SEC_ARR_OUTSIDE_BOUND_READ`
- `SEC_ARR_OUTSIDE_BOUND_WRITE`
- `SEC_FREED_PTR_RETURN`
- `SEC_READ_FREED_PTR`
- `SEC_WRITE_FREED_PTR`
- `SEC_NULL_PTR_DEREF`

The warnings are emitted to `stderr` along with all other compile time errors and warnings. They are controlled by the `-erroff`, `-errtags`, and `-errwarn` command line options and the `error_messages()` pragma, similar to all other compile time messages.

How to use `-errwarn` Command Line Option

The `-errwarn` command line option can be used to turn the memory safety check warnings into fatal errors. For example:

- `-errwarn=SEC_NULL_PTR_DEREF` makes any null pointer dereferences fatal errors.
- `-errwarn=%all` makes all compile time warnings fatal errors.

How to Disable Static Error Checking

The static error checking is run in parallel with the compiler back end processing. In certain situations, such as heavily loaded systems or extremely large modules containing extremely complex flow control, compile time can be increased. For scenarios where compile time is absolutely critical, static error checking can be disabled at the expense of suppressing possibly serious diagnostic messages via the command line option `-xsecure_code_analysis=no`.

Alternatively, you can use Oracle Developer Studio's default `config` file feature to disable static error checking for an entire site. For example, you can use the config files `cc.defaults` or `CC.defaults`. You can also use `c89.defaults` or `c99.defaults`. The install path is *install-dir*`/lib/compilers/etc/config.`.

You can use the `SPRO_DEFAULTS_PATH` environment variable to disable default static error checking without the need for makefile changes: `SPRO_DEFAULTS_PATH=`*path*, where `path` is the directory containing the defaults file.

Using the `error_messages()` Pragma to Enable and Disable SEC Messages

The `error_messages()` pragma can be used to selectively enable and disable SEC messages according to specified regions of your source files.

The following example shows how to disable specific SEC warning on a specific statement:

```
cat foo.c:
     <some code>
#pragma error_messages (off, tag-of-interest)
   <statement of interest>
#pragam error_messages (default|on, tag-of-interest)
   <remainder of code>
```

where *tag-of-interest* is one of the SEC front end warnings.

To disable all SEC warnings in a region of code:

```
    cat foo.c:
     <some code>
#pragma error_messages (off, SEC_UNINITIALIZED_MEM_READ,SEC_UNINITIALIZED_BITOP,
SEC_UNDEFINED_RETURN_VALUE,SEC_ARR_OUTSIDE_BOUND_READ,
SEC_ARR_OUTSIDE_BOUND_WRITE,SEC_DOUBLE_FREE,SEC_FREED_PTR_RETURN,SEC_READ_FREED_PTR,
SEC_WRITE_FREED_PTR,SEC_NULL_PTR_DEREF
)
   line-or-lines-of-interest
#pragam error_messages (default|on, SEC_UNINITIALIZED_MEM_READ,SEC_UNINITIALIZED_BITOP,
SEC_UNDEFINED_RETURN_VALUE,
SEC_ARR_OUTSIDE_BOUND_READ,SEC_ARR_OUTSIDE_BOUND_WRITE,SEC_DOUBLE_FREE,
SEC_FREED_PTR_RETURN,SEC_READ_FREED_PTR,SEC_WRITE_FREED_PTR,
SEC_NULL_PTR_DEREF
)
   <remainder of code>
```

Performance Library Changes

Oracle Developer Studio Performance Library is a set of optimized, high-speed mathematical subroutines for solving linear algebra and other numerically intensive problems. Oracle Developer Studio Performance Library is based on a collection of public domain subroutines available from Netlib at `http://www.netlib.org`. Oracle enhanced these public domain subroutines and bundled them as the Oracle Developer Studio Performance Library.

For this release, the following changes were made:

- LAPACK in Oracle Developer Studio Performance Library is upgraded to version 3.5.0. All of the new features in LAPACK 3.5.0 are implemented in Oracle Developer Studio Performance Library, including the following:
  - Removal of optional arguments for Fortran 95 interfaces.
  - Base LAPACK routines updated to match LAPACK 3.5.0
- New required minimum for the `STACKSIZE` environment variable set to 8 MBytes per thread.

  The Oracle Developer Studio Performance Library divides large data into blocks that fit within the hardware cache, and copies those blocks into the stack for better locality. Faster hardware means larger caches and thus larger blocks and larger stack sizes. To better accommodate current hardware, the required minimum stack size per thread is 8 Mbytes on all platforms.

For more information, see *Oracle Developer Studio 12.5: Performance Library User's Guide*.

Math Library Changes

In Oracle Solaris Studio 12.4 and earlier releases of the compilers, the `<sunmath.h>` header file included C/C++ prototypes for "wrapper" functions intended to be called from Fortran to invoke many of the nonstandard math functions and floating-point utility routines in libsunmath. These "wrapper" functions themselves simply call corresponding functions that can be called from C/C++ directly, so there is no need for a C/C++ program to use them. For example, the "wrapper" function `r_atan2d_` is equivalent to the C function `atan2df`.

In Oracle Developer Studio 12.5 , the "wrapper" functions are deprecated. Their prototypes in `<sunmath.h>` are now guarded by the preprocessor macro `__SUNMATH_DEPRECATED`; using these functions in a C program produces a warning:

```
example% cat func.c
    #include <sunmath.h>
    float func(float x, float y) {
        return r_atan2d_(&x, &y);
    }
    example% cc -c func.c
    "func.c", line 4: warning: implicit function declaration:
    r_atan2d_
    example%
```

The resulting program will likely produce incorrect results. Such a program should be rewritten to use the corresponding C function directly:

```
return atan2df(x, y);
```

Alternatively, you can add`#define __SUNMATH_DEPRECATED prior` to the inclusion of `<sunmath.h>` or add `-D__SUNMATH_DEPRECATED` to the compiler flags specified on the command line, but note that the "wrapper" functions might be removed altogether in a future release.

Index