Oracle® Developer Studio 12.6: Release Notes

Part No: E77783
July 2017

Oracle Developer Studio 12.6: Release Notes

Part No: E77783

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Référence: E77783

Accès aux services de support Oracle

Les clients Oracle qui ont souscrit un contrat de support ont accès au support électronique via My Oracle Support. Pour plus d'informations, visitez le site `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` ou le site `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` si vous êtes malentendant.

Contents

Using This Documentation

- **Overview** – Provides information about known problems and other issues with this Oracle Developer Studio 12.6 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://www.oracle.com/pls/topic/lookup?ctx=E77782-01`

Feedback

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

Oracle Developer Studio 12.6 Release Notes

This document provides information that you need to know before installing and using the product. See *Oracle Developer Studio 12.6: Installation Guide* for installation instructions.

System Requirements

Oracle Developer Studio 12.6 software can be installed on Oracle Solaris 10 or Oracle Solaris 11.3 operating system on SPARC-based or x86-based platforms, or on the Oracle Linux operating system.

TABLE 1 System Requirements

System Resources	Oracle Solaris on SPARC Based Systems	Oracle Solaris on x86 Based Systems	Linux on x86 Based Systems
Operating System	Oracle Solaris 11.3 (SRU20 or later is required) Oracle Solaris 10 1/13	Oracle Solaris 11.3 (SRU20 or later is required) Oracle Solaris 10 1/13	OL 6.7+, UEK 3, UEK 4 OL 7.2+, UEK 3, UEK 4 RH 6.7+, 7.2+
CPU	SPARC M Series SPARC T3 system or later	Intel Nehalem system or later	Intel Nehalem system or later
Memory	Minimum: 4GB	Minimum: 4GB	Minimum: 4GB
Temporary disk space required by the installer	2.2 GB	2 GB	2 GB
Disk space required by the installed product Use the `df -k` command to check your disk space	1.2 GB	1.0 GB	1.0 GB
Swap Space	Oracle Solaris 10: Minimum 8 GB Oracle Solaris 11: See Oracle Solaris Documentation for swap size recommendations	Oracle Solaris 10: Minimum 8 GB Oracle Solaris 11: See Oracle Solaris Documentation for swap size recommendations	Minimum: 8 GB

System Resources	Oracle Solaris on SPARC Based Systems	Oracle Solaris on x86 Based Systems	Linux on x86 Based Systems
System software dependencies	Oracle Solaris 11: system/header, sunpro-incorporation, and developer-studio-utilities packages Oracle Solaris 10: Entire Solaris Software Group Plus OEM Support, Entire Solaris Software Group, or Developer Solaris Software Group	Oracle Solaris 11: system/header, sunpro-incorporation, and developer-studio-utilities packages Oracle Solaris 10: Entire Solaris Software Group Plus OEM Support, Entire Solaris Software Group, or Developer Solaris Software Group	Linux: Development/Libraries Package Group, including glibc.i686, glibc-devel, and glibc-devel.i686
Java Version	Java 7 (1.7.0_72) or later; Java 8 (1.8.0_20)	Java 7 (1.7.0_72) or later; Java 8 (1.8.0_20)	Java 7 (1.7.0_72) or later; Java 8 (1.8.0_20)

You can choose to install the IDE, Performance Analyzer and dbxtool on a local (client) system and use them to connect to a remote server running the Oracle Developer Studio product. Supported configurations for the client system include the following:

- OS: Microsoft Windows 7 Professional or later; Ubuntu 9.10; Macintosh OS X 10.7 or later
- Java: One of the Java versions specified in the previous table
- Disk space required for these components: 300 MB

Required System Software Packages

Before you execute the Oracle Developer Studio 12.6 compilers and tools, verify that the system on which you are running the software has the required system packages. The following table lists the required software packages for each supported operating system.

TABLE 2 System Software Required by Developer

Operating System Version	System Software Packages
Oracle Solaris 11.3	Please see the *Oracle Developer Studio 12.6: Installation Guide* to ensure that your system has the appropriate version of the sunpro-incorporation and developer-studio-utilities packages. Check if the following packages are installed: pkg:/developer/base-developer-utilities pkg:/developer/library/xprofile pkg:/diagnostic/cpu-counters pkg:/library/glib2 pkg:/library/libxml2 pkg:/library/zlib pkg:/runtime/python-27 pkg:/system/library/c++-runtime

Operating System Version	System Software Packages

	`pkg:/system/library/libv12n`
	`pkg:/system/library/mmheap`
	`pkg:/system/library/openmp`
	`pkg:/system/linker`
	`pkg:/developer/library/lint`
	`pkg:/shell/bash`
	`pkg:/shell/ksh93`
	`pkg:/system/header`
	`pkg:/system/library/math`
	`pkg:/system/resource-mgmt/resource-pools`
	`pkg:/library/c++/stdcxx`

Oracle Solaris 10 1/13	`SUNWhea`
	`SUNWarc`
	`SUNWcpp`
	`SUNWlibstdcxx4`
	`SUNWpiclh`
	`SUNWpiclr`
	`SUNWPython-devel`
	`SUNWcar`
	`SUNWcpcu`
	`SUNWcsl`
	`SUNWcslr`
	`SUNWdpl`
	`SUNWgnome-base-libs`
	`SUNWlibC`
	`SUNWlibm`
	`SUNWlibms`
	`SUNWlibmsr`
	`SUNWlxml`
	`SUNWpython`
	`SUNWsprot`
	`SUNWzlib`

Oracle Linux 6
Oracle Linux 7
	Development/Libraries Package Group and 32-bit runtime support libraries including:
	`glibc`
	`glibc.i686`
	`glibc-devel`
	`glibc-devel.i686`
	`elfutils-libelf-devel`
	`elfutils-libelf-devel.i686`
	`libgcc-4.8.5-4.el7.x86_64`
	`zlib`
	`zlib.i686`
	`libstdc++`
	`libstdc++.i686`
	`libgcc libgcc.i686`
	`openssl`
	`openssl.i686`

Note - `developer-studio-utilities` and `studio-runtime` are “group” packages, which are an IPS shorthand to simplify installation of groups of related packages. They consist entirely of dependences on other packages but have no content of their own.

`studio-runtime` is a group package that installs system runtime support libraries used by programs compiled using Oracle Developer Studio compilers. Examples include the Fortran and Sun Performance libraries. It should be installed on systems where applications are deployed.

`developer-studio-utilities` is a group package that installs `studio-runtime` and system utility programs used by programmers using Oracle Developer Studio. Examples include make, sccs, and assemblers. It should be installed on systems where Oracle Developer Studio 12.6 is deployed.

Installation Instructions

See the *Oracle Developer Studio 12.6: Installation Guide* for installation instructions for all distributions.

End of Features Notices

Features that have been removed in this release and announcements for features that might be removed in future releases are listed on End of Features (EOF) Planned for Future Releases of Oracle Developer Studio.

Third-Party Software Information

Oracle Developer Studio 12.6 software includes third-party technologies governed by the *License Information User Manual Oracle Developer Studio 12.6 Last Updated: JUNE 2017*.

Usage Data to Oracle

The Usage Data to Oracle feature periodically sends information on your usage of Oracle Developer Studio components to Oracle Corporation. This information is used by Oracle

Corporation to improve future Oracle Developer Studio software releases. This information is anonymous and cannot be associated to any individual or organization.

However, if you wish to disable Usage Data to Oracle, set the `SUNW_NO_UPDATE_NOTIFY` environment variable to any value other than zero (`0`).

Documentation For This Release

The following documentation is available with this Oracle Developer Studio 12.6 release.

- *Oracle Developer Studio 12.6: Installation Guide*. This document includes information for installing from a tar file, a graphical package installer, and a command-line package installer.
- *What's New in the Oracle Developer Studio 12.6 Release*. This document describes the new features of the Oracle Developer Studio 12.6 release.
- *Oracle Developer Studio 12.6: Overview*. This document provides a general description about how to use the compilers and tools and how they work together.
- Online help. The online help available through the Help menu in the IDE provides task-oriented information on using all of the components of the IDE. Online help for Performance Analyzer, Thread Analyzer, Code Analyzer, and `dbxtool` is available through the Help menu in each tool.
- These online reference manual pages describe user commands, libraries that are supplied with the compilers, and other types of commands. Man pages contain reference information, including command syntax, usage, and related commands.

  Oracle Developer Studio 12.6 Man Pages. This document contains all the man pages for the Oracle Developer Studio12.6 release. You can also access these documents through the `man` command in the installed Oracle Developer Studio 12.6 software.
- Oracle Developer Studio 12.6 manuals and tutorials. You can access these documents in HTML format or PDF format through the documentation library at `http://docs.oracle.com/cd/E60778_01/`.

Known Problems, Limitations, and Workarounds in This Release

This section describes some of the known issues at the time of this Oracle Developer Studio 12.6 release, and provides information about how to work around these problems.

Compiler Issues

This section describes known issues, problems, and workarounds for the compilers in this release.

Issues Common to the Compilers

This section describes known issues that apply to the `cc`, `CC`, and `f95` compilers.

Hardware Capability Issues

The Oracle Developer Studio compilers will not execute on UltraSPARC T1 based systems, including SPARC T1000 and SPARC T2000 systems.

`libsunperf` in Oracle Solaris

On Oracle Solaris, the `libsunperf` packages and patches, do not provide the new LAPACK 3.6.1 routines for `libsunperf.so`. A workaround is to statically link by using the `-staticlib=sunperf` option, which will link with `libsunperf.a`, available in Oracle Developer Studio 12.6.

Problem with OpenMP Processor Binding Using Reserved Processors

OpenMP processor binding does not respect processor reservations using the `taskset` command on Linux.

x86 Compiler

Using flag `-xarch=amdsse4a` or `-msse4a` in the compilation line to generate an executable program, then when running this program, it causes a segmentation fault on a processor which does not support ISA `sse4a`, for example, an Intel processor.

`-xtarget=S7` and `-xchip=S7` Options

`-xtarget=S7` and `-xchip=S7` options are supported in the compiler driver but not documented in the compiler man pages.

C++ Compiler Issues

This section describes known issues, problems, and workarounds for the C++ compiler in this release.

Problems Compiling BOOST Libraries

Not all of BOOST libraries compile successfully. The results you get might depend on the version of BOOST you use, and how you exercise the libraries. BOOST libraries are included in Oracle Developer Studio compiler testing and progress is being made toward compiling all of BOOST. If you run into a problem that prevents you from making progress, please report it.

Apache Standard Library Issue on Oracle Solaris

The Apache `stdcxx` library installed in Oracle Solaris 10 8/11 and later, and in the initial releases of Oracle Solaris 11, 11.1, and 11.2, does not work correctly with the new compiler default of `-template=no%extdef`. You will probably need to add the option `-template=extdef` to `CC` command lines that use this library. The problem is fixed in Oracle Solaris 11.3 SRU2. No fix for Oracle Solaris 10 is currently available. For more information, see Understanding the Effects of the Changed Default C++ Template Compilation Model (`http://www.oracle.com/technetwork/articles/servers-storage-dev/changed-default-cpp-template-model-2292727.html`).

`-xustr=ascii_utf16_ushort` Option is Incompatible with `-std=c++11`

Specifying the flag `-xustr=ascii_utf16_ushort` results in an error if `-std=c++11` or `-std=c++14` option is in effect.

Use the C++11 Unicode character and string options, when compiling in these modes.

Name Mangling Linking Problems

The name mangling performed by Oracle Developer Studio in `-compat=5` mode has errors. Fixing the errors would have created binary incompatibilities. The errors were found and corrected prior to Oracle Studio releases on x86/Oracle Solaris with `-m64` and on Linux, but remain on the other platforms. A new option, `-abiopt=mangle6` corrects the mangling errors on SPARC/Oracle Solaris and x86 Oracle Solaris with the `-m32` option.

Refer to “–abiopt=[mangle5|mangle6]” in *Oracle Developer Studio 12.6: C++ User’s Guide*.

No Support for Referencing a Non-Global Namespace Object From a Template

A program using templates and static objects causes link-time errors of undefined symbols if you compile with `-instances=extern`. This is not a problem with the default setting `-instances=global`. The compiler does not support references to non-global namespace-scope objects from templates. Consider the following example:

```
static int k;
template<class T> class C {
        T foo(T t) { ... k ... }
};
```

In this example, a member of a template class references a static namespace-scope variable. Keep in mind that namespace scope includes file scope. The compiler does not support a member of a template class referencing a static namespace-scope variable. In addition, if the template is instantiated from different compilation units, each instance refers to a different `k`, which means that the C++ One-Definition Rule is violated and the code has undefined behavior.

Depending on how you want to use `k` and the effect it should have, the following alternatives are possible. The second option is available only for function templates that are class members.

- You can give the variable external linkage:

  ```
  int k; // not static
  ```

  All instances use the same copy of `k`.
- You can make the variable a static member of the class:

  ```
  template<class T> class C {
          static int k;
          T foo(T t) { ... k ... }
  };
  ```

  Static class members have external linkage. Each instance of `C<T>::foo` uses a different `k`. An instance of `C<T>::k` can be shared by other functions. This option is probably what you want.

`#pragma align` Inside Namespace Requires Mangled Names

When you use `#pragma align` inside a namespace, you must use mangled names. For example, in the following code, the `#pragma align` statement has no effect. To correct the problem, replace `a`, `b`, and `c` in the `#pragma align` statement with their mangled names.

When compiling as C++11, you can use the standard keyword `alignas` to avoid problems with pragmas.

```
namespace foo {
  #pragma align 8 (a, b, c) // has no effect
  //use mangled names: #pragma align 8 (__1cDfooBa_, __1cDfooBb_, __1cDfooBc_)
  static char a;
  static char b;
  static char c;
}
```

Fortran Compiler Issues

This section describes known issues and problems for the Fortran compiler in this release.

- Blank space before the end of a no advance print line does not affect output position.

  Having the X edit descriptor at the end of a format of an output statement does not affect the position of the next character in the output record. This causes a difference if the output statement has **`ADVANCE='NO'`** and there are more characters to be transferred to the same record by subsequent output statements.

  In many cases, this can be worked around by having a blank character string edit descriptor instead of the *n* **`X`** edit descriptor. They are not exactly the same since the blank character string edit descriptor actually causes blank characters to go into the record whereas the *n* **`X`** only skips over the next *n* characters, usually causing blanks to be in those skipped positions by default.
- Valid code rejected when a line consists of two continuation ampersands.

  An empty continuation line with a single ampersand is forbidden by the Fortran standard. However, with two ampersands on the same line, an empty continuation line can still be created without falling under the standard restriction. The compiler does not handle that case and gives an error. The workaround is to delete that line which only affects the readability of the program without adding any semantics. BOZ constants sometimes get truncated.
- In some relatively more complex scenarios, such as an array construct, a BOZ constant might get truncated to the default integer size of 4 bytes even though the corresponding item it is supposed to be assigned to is an 8-byte integer entity. The workaround is to use constants of correct type and kind in array construct instead of BOZ constants.
- Correction of rounding algorithm in the new release might cause differences in outputs of list-directed, name-list directed, and formatted-writes sing the `ROUND='NEAREST'` and `ROUCH='COMPATIBLE'` compared to outputs in previous releases. The rounding difference should only be in the least-significant digit.

Lacking Oracle Solaris `libfsu` Patches for Oracle Developer Studio 12.6

The version of `libfsu` on Oracle Solaris might not have the latest patches needed to make it work properly with the Parameterized Derived Types feature. Therefore, programs using this feature does not work properly on Oracle Solaris even when they work on Linux.

For this feature to work on Oracle Solaris, you can use the `-Bstatic` option to link with the archive version of `libfsu`.

`libsunmath` Library is not Supported on Linux

The `libsunmath` library is not supported on Linux. Therefore, some non-intrinsic math functions will become unresolved, when compiled on Linux platforms.

The Fortran-callable functions in `libsunmath` are:

```
abrupt_underflow, convert_external, d_acos, d_acosd, d_acosh, d_acosp, d_acospi,
d_addran, d_addrans, d_aint, d_anint, d_annuity, d_asin, d_asind, d_asinh,
d_asinp, d_asinpi, d_atan2, d_atan2d, d_atan2pi, d_atan, d_atand, d_atanh,
d_atanp, d_atanpi, d_cbrt, d_ceil, d_compound, d_copysign, d_cos, d_cosd,
d_cosh, d_cosp, d_cospi, d_erf, d_erfc, d_exp10, d_exp2, d_exp, d_expm1, d_fabs,
d_floor, d_fmod, d_get_addrans, d_hypot, d_infinity, d_init_addrans, d_j0,
d_j1, d_jn, d_lcran, d_lcrans, d_lgamma, d_lgamma_r, d_log10, d_log1p, d_log2,
d_log, d_logb, d_max_normal, d_max_subnormal, d_min_normal, d_min_subnormal,
d_mwcran, d_mwcrans, d_nextafter, d_pow, d_quiet_nan, d_remainder, d_rint,
d_scalb, d_scalbn, d_set_addrans, d_shufrans, d_signaling_nan, d_significand,
d_sin, d_sincos, d_sincosd, d_sincosp, d_sincospi, d_sind, d_sinh, d_sinp,
d_sinpi, d_sqrt, d_tan, d_tand, d_tanh, d_tanp, d_tanpi, d_y0, d_y1,
d_yn, gradual_underflow, i_addran, i_addrans, i_get_addrans, i_get_lcrans,
i_get_mwcrans, i_init_addrans, i_init_lcrans, i_init_mwcrans, i_lcran,
i_lcrans, i_llmwcran, i_llmwcrans, i_lmwcran, i_lmwcrans, i_mwcran, i_mwcrans,
i_set_addrans, i_set_lcrans, i_set_mwcrans, i_shufrans, id_finite, id_fp_class,
id_ilogb, id_irint, id_isinf, id_isnan, id_isnormal, id_issubnormal, id_iszero,
id_nint, id_signbit, ieee_flags, ieee_handler, ieee_retrospective, iq_finite,
iq_fp_class, iq_ilogb, iq_isinf, iq_isnan, iq_isnormal, iq_issubnormal,
iq_iszero, iq_signbit, ir_finite, ir_fp_class, ir_ilogb, ir_irint, ir_isinf,
ir_isnan, ir_isnormal, ir_issubnormal, ir_iszero, ir_nint, ir_signbit,
nonstandard_arithmetic, q_copysign, q_fabs, q_fmod, q_infinity, q_max_normal,
q_max_subnormal, q_min_normal, q_min_subnormal, q_nextafter, q_quiet_nan,
q_remainder, q_scalbn, q_signaling_nan, r_acos, r_acosd, r_acosh, r_acosp,
r_acospi, r_addran, r_addrans, r_aint, r_anint, r_annuity, r_asin, r_asind,
r_asinh, r_asinp, r_asinpi, r_atan2, r_atan2d, r_atan2pi, r_atan, r_atand,
r_atanh, r_atanp, r_atanpi, r_cbrt, r_ceil, r_compound, r_copysign, r_cos,
r_cosd, r_cosh, r_cosp, r_cospi, r_erf, r_erfc, r_exp10, r_exp2, r_exp,
r_expm1, r_fabs, r_floor, r_fmod, r_get_addrans, r_hypot, r_infinity,
r_init_addrans, r_j0, r_j1, r_jn, r_lcran, r_lcrans, r_lgamma, r_lgamma_r,
r_log10, r_log1p, r_log2, r_log, r_logb, r_max_normal, r_max_subnormal,
r_min_normal, r_min_subnormal, r_mwcran, r_mwcrans, r_nextafter, r_pow,
r_quiet_nan, r_remainder, r_rint, r_scalb, r_scalbn, r_set_addrans, r_shufrans,
r_signaling_nan, r_significand, r_sin, r_sincos, r_sincosd, r_sincosp,
r_sincospi, r_sind, r_sinh, r_sinp, r_sinpi, r_sqrt, r_tan, r_tand, r_tanh,
r_tanp, r_tanpi, r_y0, r_y1, r_yn, sigfpe, smwcran, standard_arithmetic,
u_addrans, u_lcrans, u_llmwcran, u_llmwcrans, u_lmwcran, u_lmwcrans, u_mwcran,
u_mwcrans, u_shufrans_
```

Fortran Padding Value Change

Padding values that are generated using the `-pad=common` compiler option might be different in code that is compiled with Oracle Developer Studio 12.6 compared to prior releases. If you use `-pad=common` to compile two or more files that reference the same common block, you must compile them with the `f95` compiler from the same Oracle Developer Studio release.

Fortran 77 Libraries Removed

This is a reminder that the Oracle Solaris Studio 12.2 release removed the obsolete FORTRAN 77 libraries. This means that old executables compiled with the legacy Sun WorkShop `f77` compiler that depend on the shared libraries `libF77`, `libM77` and `libFposix` will not run.

Array Intrinsic Functions Use Global Registers

The array intrinsic functions `OT_PRODUCT` and `MATMUL` are highly tuned for the appropriate SPARC platform architectures. As a result, they use the global registers`%g2`, `%g3`, and `%g4` as scratch registers.

For interval arithmetic, the array intrinsics `ANY, ALL, COUNT, MAXVAL, MINVAL, SUM, PRODUCT` are also affected.

User code should not assume these registers are available for temporary storage if calls are made to the array intrinsics listed above. Data in these registers will be overwritten when the array intrinsics are called.

F95 Modules in Archive Libraries Not Included in Executable

The debugger `dbx` requires all object files used in the compilation to be included in the executable file. Usually, programs satisfy this requirement with no extra work on the part of the user. An exceptional case arises from the use of archives containing modules. If a program uses a module, but does not reference any of the procedures or variables in the module, the resulting object file will not contain references to the symbols defined in the module. The linker only links with a object file from an archive if there is a reference to a symbol defined in the object file. If there is no such reference, the object file will not be included in the executable file. `dbx` will give a warning when it tries to find the debugging information associated with the module that was used. It will not be able to provide information about the symbols whose debugging information is missing.

Use the `-u` linker option to work around this problem. This option takes a symbol as its option argument. It adds that symbol to the set of undefined linker symbols, so it will have to be resolved. The linker symbol associated with a module is normally the module name with all letters in lower case followed by an underscore.

For example, to force the object file containing the module `MODULE_1` to be taken from an archive, specify the linker option `-u module_1_`. If linking using the `f95` command, use `-Qoption ld -umodule_1_` on the command line.

`gethrtime(3F)` on Linux Platforms

There is no reliable way to accurately obtain the clock rate on AMD processors when system power saving is enabled. As a result, using timing functions based on `gethrtime` (3F) (the Fortran compiler's Linux version of the Oracle Solaris `gethrtime` (3C) function) to get high resolution real time on Linux platforms will only be accurate on AMD systems with power saving disabled. A reboot of the system might be required to disable the power-saving features.

Tools Issues

This section lists known limitations of the debugging tools and performance analysis tools.

`dbx` Limitations and Incompatibilities

`dbx` has the following limitations:

- Using older copies of `libC.so.5` or `libC.so.4` might cause problems for `dbx` in the area of C++ exceptions. Warning messages about bad stabs and unhandled exceptions might result.

  Workaround: Install the latest `libC.so.5` on all systems.
- Fortran users should compile with the `-stackvar` option to take full advantage of runtime checking.
- Some programs might not work properly with the `-stackvar` option. In such cases, try the `-C` compiler option, which enables array subscript checking without RTC.
- The `dbx` command line interpreter is an older version of the Korn shell (ksh) that does not support Code Set Independence (CSI). Multi-byte characters can be misinterpreted when typed on the `dbx` command line.
- The following features of `dbx` are not available on the Linux OS:
  - Fix and continue
  - Performance data collection on multithreaded applications.
  - Breakpoints on the following events:
    - `fault`
    - `lastrites`
    - `lwp_exit`
    - `sysin`
    - `sysout`
    - `sync`
  - Index DWARF (compiler option `-xs=no`)
- The following problems might occur when debugging programs on Linux platforms:
  - TLS access is not supported.
  - To debug 32-bit programs, you must start `dbx` with the `-x exec32` option.
  - The pipe operator in the Korn shell is limited on Linux platforms. Any `dbx` command that needs to access the target process does not work as part of a pipeline. For example, the following command is likely to cause `dbx` to hang:

    ```
    where | head -1
    ```

    Workarounds:
    - Type Ctrl-C to display a new `dbx` prompt.
    - `dbx` caches a lot of information, so for the above example, the following sequence of commands works:

      ```
      where
      where | head –1
      ```
    - Redirect command output to a file and then display the file contents.

```
(dbx) > bag where
(dbx) cat bag
```

- `dbx` does not support the following features for the GNU C and C++ compilers:
  - VL array (`gcc` 4.1 and before)
  - OpenMP
  - RTTI
  - Template definition
  - Default argument
  - `using` directive
  - `friend`
- The following issues exist for `dbx` on Oracle Linux 6:
  - Indirect reference symbols used in system libraries can sometimes cause `dbx` to set breakpoints at the reference rather than the actual function.
  - When debugging code compiled with `gcc` 4.4.4 compilers, be aware that `dbx` does not see macros defined using the `-D` compiler option.
- The stabs format for debugging information is supported in Oracle Developer Studio, but Oracle has announced that the format might eventually be discontinued in favor of DWARF format. Oracle is not required to implement stabs support for new features or for enhancements to existing features.

  The following debugging features are not supported by the stabs format.
  - New C++11 and new C++14 features.
  - New C11 features
  - Parameters and local variables in optimized code.
  - Macros (when code is compiled with the `-g3` option).
  - Subsets of debugging info (generally when compiled with `-xdebuginfo={...}`).
  - Object Oriented Fortran
- See “`dbx attach` Profiling (`collect -P`)” on page 25 for information about data collection problems when `dbx` is attached to a process.
- Debugging support for exception handling is limited if GNU runtime libraries are used.
- The following features are not supported with `discover` instrumented binaries:
  - Fix and continue
  - Runtime Checking customization variables (`dbxenv`)
- `discover` analysis will not be displayed in the `dbx` console if the debuggee was instrumented in HTML mode. A workaround is to use the text mode option `-w` during instrumentation.

Performance Analyzer and `er_print` Utility Limitations

This section describes known problems with the Performance Analyzer tool and `er_print` utility.

- On Linux, using the `collect -P` command to attach to a running multithreaded process (including Java), will not properly collect data.
- Performance Analyzer crashes with Java 7U65. Use at least Java 7U75 when using the Performance Analyzer tool.
- The Library Visibility functionality for shared objects does not always work properly in conjunction with filtering.
- Performance Analyzer and `er_print` cannot find shared objects embedded in jar files.
  In addition, under some circumstances Performance Analyzer and `er_print` cannot find Java class files or source files. To work around these issues, use `addpath` to point to the directories containing the files. See the Troubleshooting section of the Help in Performance Analyzer for more detailed information.
- When you compare experiments you might see the following issue:

  Source and Disassembly views show inclusive and exclusive metrics instead of only inclusive metrics as expected.
- You might see the following issues related to CPUs with variable clock rate:
  - On a system with a variable clock rate, hardware counter profiling metrics for cycle-based counters are under-reported when the processor runs at less than maximum speed and the metrics are converted to times.
  - On systems with multiple CPUs running at different clock frequencies, experiments will be processed based on the clock rate for one CPU, which can lead to over-counting or under-counting on other CPUs.
- Profiling OpenMP applications that use many threads on SPARC T5, M5, and M6 systems might produce very large experiments that cannot be read. The workaround is to set the environment variable `SP_COLLECTOR_NO_OMP` before you run `collect`. Note that when you set this variable, you cannot see OpenMP constructs and metrics in the resulting experiment and the User, Expert, and Machine view modes all look the same.

Limitations for Profiling Applications on a Remote Host

Remote login with a pass phrase is not yet supported.

`collect` Utility

This section describes known problems with the `collect` utility and data collection for the Performance Analyzer tool.

- Profiling of mixed Java and C++ applications might not be complete when using JDK 1.7.0_40 through JDK 1.7.0_59 due to a bug in the JDK. The stack for calls from Java to C++ are not properly unwound.
- Heap tracing on Linux does not trace calls to `calloc`.
- Collection of count data (`collect -c`) does not work for binaries that were compiled with `-std=c++11`.
- Clock profiling on SPARC T4/T5 M5/M6 might have irregular samples due to a bug in Oracle Solaris setitimer (ITIMER_REALPROF, ...)
- Hardware counter profiling with attribute system=1 has the following known issues. Some system time might not be counted correctly for `usleep()`. On Oracle Solaris 10, using the system attribute can hang the application.
- Clock profiling might incorrectly report User CPU Time as Wait CPU Time due to a bug in Oracle Solaris
- You might see following problems with profiling signals in kernel zones:
  - Clock-profiling signals in kernel zones are erratically delivered, leading to missing data.
  - Hardware counter `dtlb_misses.any` doesn't work on Westmere kernel zones
  - Memoryspace profiling on x86 is not supported on kernel zones
  - On SPARC, hardware counter profiling samples might be delayed by many seconds; lost samples will result in undercounts
  - Clock-profiling samples in kernel zones might report the wrong microstate
  - HWC-profiling samples on a the control domain of a system which is running kernel zones might be lost.
  - HWC-profiling samples on a kernel zone might be lost.
- Profiling SPECJBB2015 on Linux can cause a crash or hang.
- Various cornercases of stack unwind might fail on x86, both Oracle Solaris and Linux

`er_kernel` Utility

This section describes known problems with the `er_kernel` utility.

- `er_kernel` refuses to run on an Oracle Solaris system that is running under Oracle VM to avoid an Oracle VM bug that causes a reboot

- Callstacks might sometimes omits a frame, typically when the leaf function is within its epilogue
- Hardware counter profiling samples on the control domain of a system which is running kernel zones might be lost.
- Hardware counter profiling samples on a kernel zone might be lost.
- Hardware counter profiling with `er_kernel` in a kernel zone might only support a single hardware counter

`dbx collector` Profiling

This section describes problems with profiling an application when using the `dbx collector` command.

- There is a limitation in using the `dbx collector` on Java. You cannot specify a Java class file for the target. Instead, you must specify the JVM as the target and specify the class file as a parameter to the `dbx run` command. For example:

  ```
  > dbx /path-to-your-jdk/bin/java
  (dbx) collector enable
  (dbx) collector java on
  (dbx) run [JVM-options] [Java-class-file-to-execute]
  ```

- On Linux, profiling of any Java applications with `dbx` or `collect -P` might fail.

`dbx attach` Profiling (`collect -P`)

This section describes problems with profiling a running application when using the `dbx attach` command:

- On Linux, attaching to a Java program with `dbx` or `collect -P` might fail.
- On Linux, attaching to a target that is in the middle of a blocking or non-blocking call might cause the target to fail.
- On Linux, attaching to a multithreaded target will not properly record data for threads that are already created at the time of the attach. No warning about missing data is provided. Note that this includes any Java targets because the JVM is multithreaded.
- If you attach `dbx` to a running process that was started without `LD_PRELOAD` to preload the collector library `libcollector.so` a number of errors can occur. You cannot collect any tracing data: synchronization wait tracing, heap tracing, I/O tracing, or MPI tracing.

Tracing data is collected by interposing on various libraries, and if `libcollector.so` is not preloaded, the interposition cannot be done.

- If the target program installs a signal handler after `dbx` is attached to the process, and the signal handler does not pass on the SIGPROF and SIGEMT signals in Oracle Solaris or the SIGIO signal in Linux, some or all profiling data might be lost.
- If the target program uses `libcpc.so` hardware-counter experiments might fail because both the collector and the program are using the library.
- If the target program calls `setitimer` (2), clock profiling experiments can be corrupted because both the collector and the program are using the timer.
- Attaching to a target that is in the middle of a call into the `malloc` library might cause the target to fail. Requesting hardware counters when attaching greatly increases the probability of such failures.

IDE Limitations

This section describes known limitations in the IDE.

To create a project from binary on a remote host, the remote host must have Oracle Solaris Studio 12.3 or 12.4 or Oracle Developer Studio 12.5 available. This feature is not supported on earlier releases.

Code Analysis Tools Limitations

This section describes known limitations to the code analysis tools.

- False positives or negatives might be found if you do not properly instrument your code. For more information, see "Supported Binaries" in *Oracle Developer Studio 12.6: Discover and Uncover User's Guide*.
- The `discover` and `uncover` tools do not work with an x86_64 binary compiled with `-xmodel=medium` when the program size (text + data) is large enough to extend beyond the maximum range of `-xmodel=small` (2^{32} - 2^{24} - 1).
- The `discover` tool doesn't work on an ancillary file, which is a file containing debug information created with the linker option `-z ancillary` in Oracle Solaris 11.3.
- To use any of the Code Analysis tools on Oracle Linux, the original binary must be compiled and linked with the `-xannotate=[yes]` option.
- To use Discover SSM in a kernel zone of an SSM capable machine, the configuration of the zone should include `host-compatibile=native`

Install and Uninstall Issues

This section describes known issues, problems, and workarounds for the Java Installer.

Installation of Oracle Instant Client With Java Installer

The command line, non-interactive mode `./developerstudio.sh --non-interactive` will not install the OIC component.

To avoid this, the command line, non-interactive mode for installation must specify OIC together with all the Compiler and Tools components after the `--install-components` option as follows:

```
./developerstudio.sh --non-interactive --install-components oic, dbxtool, code-analyzer-
tool, performance-and-thread-analysis-tools, performance-library, c-and-cpp-compilers,
 dmake
```

In addition, if a Japanese or Chinese locale is required, add the components (comma-separated) `japanese-localization` and `simplified-chinese-localization` to the `--install-components` line.

Uninstaller Does Not Work When Japanese Locale is Set

If you have your locale set to Japanese in your environment, the uninstaller does not correctly display the Japanese characters. The workaround is to unset the Japanese locale and set your `LC_ALL` environment to `C`.

```
$ set LC_ALL=C
```

If your locale makes sense, use the `--locale` installer flag.

```
$ LC_ALL=C ./developerstudio.sh --locale=ja
```