

Getting Started with the Intel® Parallel Inspector

The Intel® Parallel Inspector, an Intel® Parallel Studio product, helps you find the challenging memory and threading errors unique to multi-core software development.

This guide shows you how to use basic Intel® Parallel Inspector features and a debug build of a Windows* C++ application to identify and analyze a thread- and memory-related issue in the Microsoft Visual Studio* 2005 or 2008 development environment. After reading this guide, you should be ready to use the Intel® Parallel Inspector to identify, analyze, and resolve parallelism issues in your own multithreaded applications.

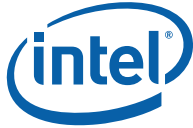
Not available
in the Beta
product.

For a more graphical getting started experience, try the Show Me videos offered at various points in this guide.

NOTE: Show Me videos require Adobe* Flash* Player. For more information, see <http://www.adobe.com/products/flashplayer/>

Contents

| | |
|--|----|
| Disclaimer and Legal Information | 2 |
| 1 Objective and Tools | 3 |
| 2 Launch Environment | 3 |
| 3 Choose Target | 4 |
| 4 Collect Result Data–Threading Errors | 4 |
| 5 Manage Result Data–Threading Errors | 6 |
| 6 Collect and Manage Result Data–Memory Errors | 10 |
| 7 Next Steps | 12 |



Disclaimer and Legal Information

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or by visiting Intel's Web Site.

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. See http://www.intel.com/products/processor_number for details.

This document contains information on products in the design phase of development.

BunnyPeople, Celeron, Celeron Inside, Centrino, Centrino Atom, Centrino Atom Inside, Centrino Inside, Centrino logo, Core Inside, FlashFile, i960, InstantIP, Intel, Intel logo, Intel386, Intel486, IntelDX2, IntelDX4, IntelSX2, Intel Atom, Intel Atom Inside, Intel Core, Intel Inside, Intel Inside logo, Intel. Leap ahead., Intel. Leap ahead. logo, Intel NetBurst, Intel NetMerge, Intel NetStructure, Intel SingleDriver, Intel SpeedStep, Intel StrataFlash, Intel Viiv, Intel vPro, Intel XScale, Itanium, Itanium Inside, MCS, MMX, Oplus, OverDrive, PDCharm, Pentium, Pentium Inside, skool, Sound Mark, The Journey Inside, Viiv Inside, vPro Inside, VTune, Xeon, and Xeon Inside are trademarks of Intel Corporation in the U.S. and other countries.

* Other names and brands may be claimed as the property of others.

Copyright © 2008, Intel Corporation. All rights reserved.

Microsoft product screen shot(s) reprinted with permission from Microsoft Corporation.



1 Objective and Tools

Objective: Identify and analyze a threading error and a memory error that prevent the debug build of a Windows* C++ sample application from correctly displaying an **Intel** banner.

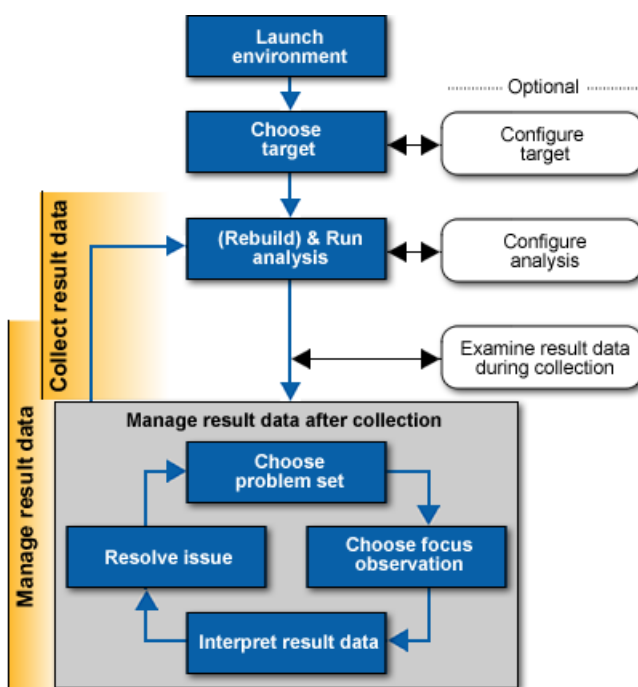
```
1. Banner program has started execution.
2. Banner output from thread pool (Correct output)..
   I I I I   n n n n   t       e e e   l
   I         n n n   t t t   e e   l
   I         n n   t       e e e   l
   I         n n   t       e       l
   I I I I   n n   t t       e e e   l l
3. Banner program has completed execution. Please press enter.
```

Sample Banner Application

Tools: Use the following tools to achieve this objective:

- Microsoft Visual Studio* 2005 or 2008 development environment
- Sample **Banner** and **BannerFixed** solutions/projects in the samples folder in the Intel® Parallel Inspector installation directory
- Basic Intel® Parallel Inspector features
- The basic Intel® Parallel Inspector workflow

See *About Workflows* in *Help* for more information.



Basic Intel® Parallel Inspector Workflow

2 Launch Environment

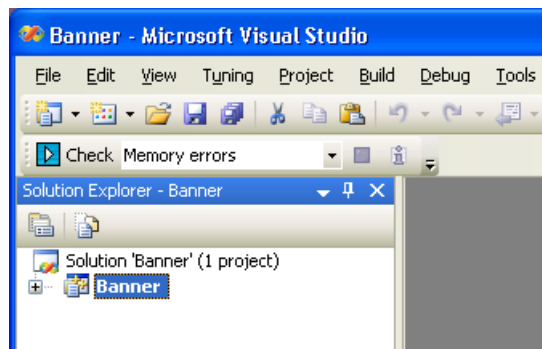
Launch the Visual Studio* 2005 or 2008 development environment.

NOTE: This guide presents screen shots from the Visual Studio* 2005 development environment.

3 Choose Target

A target is an executable file you analyze using the Intel® Parallel Inspector.

1. From the Microsoft Visual Studio menu, choose **File > Open > Project/Solution**.
2. In the **Open Project** dialog box, open the `samples/Banner.sln` file to display the **Banner** solution/project in the **Solution Explorer**.



Show Me

4 Collect Result Data-Threading Errors

Not available in the Beta product.

Build Target

See *Quick Start Tips* in *Help* for information on the most effective compilation switches.

1. Ensure the **Banner** project is highlighted.
2. From the Visual Studio* menu, choose **Build > Build Solution** to compile and link the project.

Configure and Run Analysis

When you run an analysis, the Intel® Parallel Inspector executes the target, identifies parallelism issues that may need handling, and collects result data in a `My Inspector Results\resultdir*.insp` file in the solution folder.

Collecting result data can take significant system memory and time. For example, a target that normally takes a few seconds to execute may take a few minutes to execute during Intel® Parallel Inspector analysis.



See *Quick Start Tips* in *Help* for more information on speeding up collection, such as reducing data set size.

See *About Configuring Analyses* in *Help* for more information about preset levels.

For example:
My Inspector
Results\
result-000-
tc4\
result-000-
tc4.insp,
where tc =
result type
(threading
check) and 4 =
highest preset
level.

An *event* is an instance of a problem.

A *problem* is a small group of closely related observations that indicate an error.

An *observation* is a fact about target source code.

To speed up the collection process, limit:

- The result type to either memory errors or threading errors
- The result scope to one of four preset levels

1. From the Visual Studio* menu, choose **Tools > Intel Parallel Inspector > Configure Analysis...** to display the **Configure Analysis** dialog box.

2. Check the **Check Threading Errors** radio button.

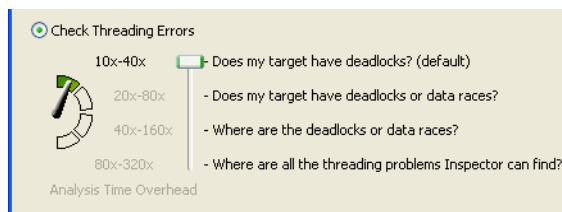
3. Drag the slider to the **Where are all the threading problems Inspector can find?** preset level.

4. Click the **Run Analysis** button to execute the target and collect result data in a My Inspector Results\resultdir*.insp file in the solution folder.

5. Notice the banner displays incorrectly (**In tl** instead of **Intel**).

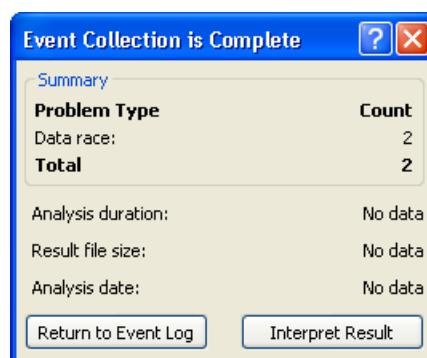
6. Press the Enter key to end target execution and display an **Event Collection is Complete** dialog box identifying problems that may need handling.

7. Click the **Interpret Result** button to start managing result data.



```

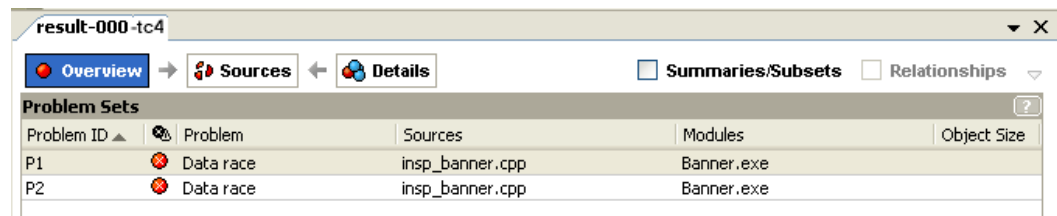
Banner Program has started execution.
Banner Program output from thread pool..
IIIII  nnnnn  t      l
I      n n n  ttt    l
I      n n   t      l
I      n n   t      l
IIIII  n n   tt     ll
Banner Program has completed execution. Please press enter.
  
```



5 Manage Result Data–Threading Errors



NOTE: The sample **Banner** application is non-deterministic. Your screens may vary from the screens shown throughout this guide.

Choose Problem Set



A *problem set* is a larger group of more loosely related observations that could share a common solution.

The **Overview** window is the default starting point for managing result data. The **Problem Sets** pane at the top of this window prioritizes problem sets:

- First by **Severity** – Problem sets containing  (**Error**) problems precede those containing  (**Warning**) problems.
- Then by number of observations – Problem sets containing more observations precede those containing fewer.

Think of the **Problems Sets** pane as a *to-do* list. Start at the top and work your way down.

In fact, the Intel® Parallel Inspector chooses the first problem set for you by default, and displays the observations in this problem set in the **Observations in Problem Set** pane at the bottom of the **Overview** window.

| Observations in Problem Set: Data race | | | | | |
|--|-------------|---------------------|------------|------------|-------------|
| Observation ID | Description | Source | Function | Module | Object Size |
| X4 | Read | insp_banner.cpp:155 | stuffChars | Banner.exe | |
| X10 | Read | insp_banner.cpp:166 | stuffChars | Banner.exe | |
| X5 | Write | insp_banner.cpp:166 | stuffChars | Banner.exe | |
| X9 | Write | insp_banner.cpp:166 | stuffChars | Banner.exe | |


NOTE: To choose another problem set, click the desired problem set.



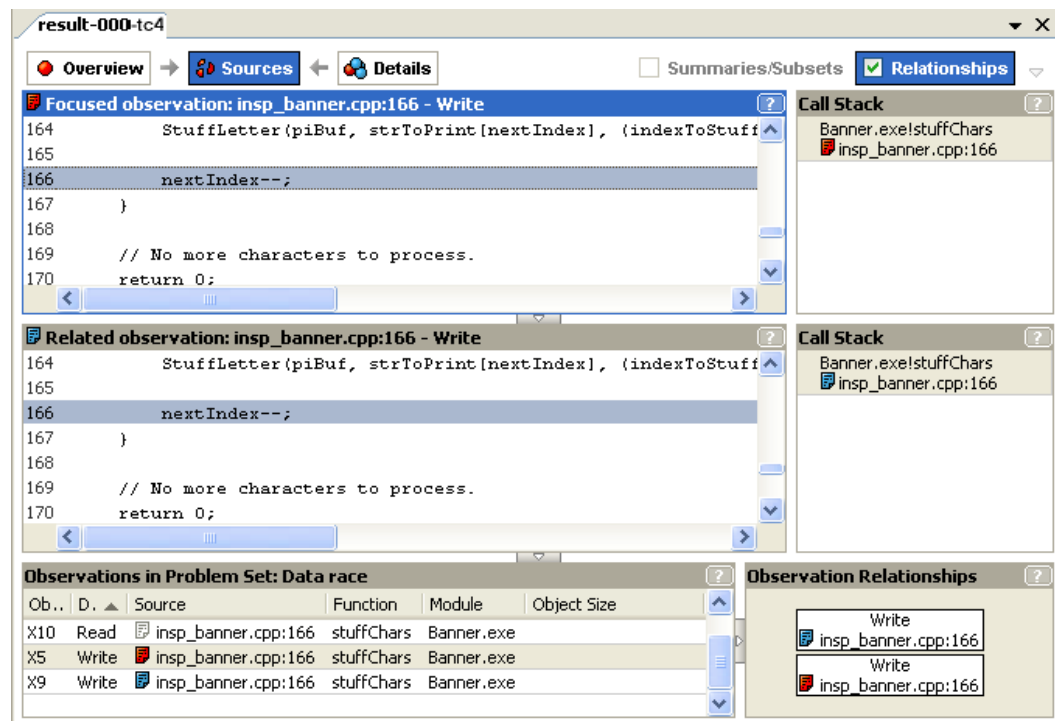
Choose Focus Observation

A *focus observation* is an observation with relationships you choose to explore.



Perhaps the problem type description—in this case, **Data race**—and the observation type descriptions—in this case, **Write**—are all you need to know to resolve a parallelism issue.




Maybe viewing the source code for one or more observations is all you need: click a  icon to display a source code snippet.

But sometimes you need to understand the relationships among observations to resolve a parallelism issue. Double-click the first **Write** observation in the **Observations in Problem Set** pane to display the **Sources** window and explore its relationship to other observations in the problem set.

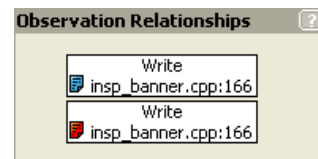


Interpret Result Data

Notice the  and  icons throughout the **Sources** window.

| Icon | Meaning |
|---|--|
|  | This observation is the focus observation selected in the previous window. It has source code available for viewing in the Intel® Parallel Inspector and editing in the Visual Studio* code editor. That source code is currently displayed in the Focus Observation Code pane (at the top of the Sources window). |
|  | This observation is related to the focus observation. It has source code available for viewing and editing, and that source code is currently displayed in the Related Observation Code pane (in the middle of the Sources window). |
|  | This observation has source code available for viewing and editing. |
| None | This observation does not have source code available for viewing and editing. |

Look at the diagram in **Observation Relationships** pane. It shows the relationship between the current focus and related observation.



In relationship diagrams:

- Each box in a diagram represents an observation in a problem.
- A diagram with a single box is a trivial problem with no related observation.
- Vertically stacked boxes indicate concurrent observations.
- Boxes arranged left-to-right with connecting arrows indicate a time ordering.
- Boxes with connecting lines indicate association.

This diagram clearly identifies the parallelism issue: two threads tried to concurrently update a shared memory location.

NOTE: To display source code for a different related observation, click the desired observation in the **Observations in Problem Set** pane. To display source code for a different focus observation, right-click desired observation to display a pop-up menu, then choose **Set as Focus Observation**.



What's Next?

Do one of the following:

- [Resolve the threading issue](#) just as you would under standard debugging circumstances.
- [Skip resolving the threading issue](#).

Resolve the Threading Issue

Under standard debugging circumstances, you would now do the following:

1. Review the source code in the **Focus Observation Code** and **Related Observation Code** panes.
2. Double-click the source code you want to edit.

Double-clicking opens the source file in a new tab where you can edit it with the Visual Studio* code editor.
3. When you're finished editing source code, click the **Overview** button on the **result-000-tc4** tab to return to the **Overview** window.
4. [Manage](#) the second problem set using the same techniques you used to manage the first problem set.
5. When you're finished managing the second problem set, [rebuild the target](#) and rerun the analysis: from the Visual Studio* menu, choose **Tools > Intel Parallel Inspector > Recheck**.

NOTE: Result type and scope are persistent by target until you reconfigure.

If you handled all threading issues appropriately, the Intel® Parallel Inspector displays no problems sets; however, the banner still displays incorrectly because the **Banner** target still contains errors—memory errors.

Skip Resolving the Threading Issue

Proceed to [Collect and Manage Result Data—Memory Errors](#).



Show Me

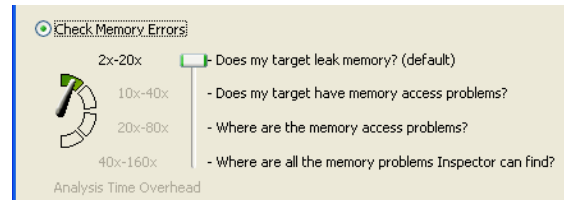
6

Collect and Manage Result Data- Memory Errors

Not available
in the Beta
product.

The process for collecting and managing memory result data is similar to that for collecting and managing threading result data. The key difference is your selection in the **Configure Analysis** dialog box.

1. Build the target (already done).
2. From the Visual Studio* menu, choose **Tools > Intel Parallel Inspector > Configure Analysis...** to display the **Configure Analysis** dialog box.
3. Check the **Check Memory Errors** radio button.
4. Drag the slider to the **Where are all the memory problems Inspector can find?** preset level.



For example:
My Inspector
Results\
result-001-
mc4\
result-001-
mc4.insp,
where mc =
result type
(memory
check) and 4 =
highest preset
result scope.

5. Click the **Run Analysis** button to execute the target and collect result data in another My Inspector Results\
resultdir*.insp file in the solution folder.
6. Notice the banner displays incorrectly (**In te** instead of **Intel**).
7. Press the Enter key to end target execution and display the **Event Collection is Complete** dialog box. Notice the parallelism issue(s) that may need handling.
8. Click the **Interpret Result** button to start managing the result data.

What's Next?

Do one of the following:

- [Manage the memory result data](#) just as you would under standard debugging circumstances.
- [Skip managing the memory result data.](#)



Manage the Memory Result Data

Under standard debugging circumstances, you would now do the following:

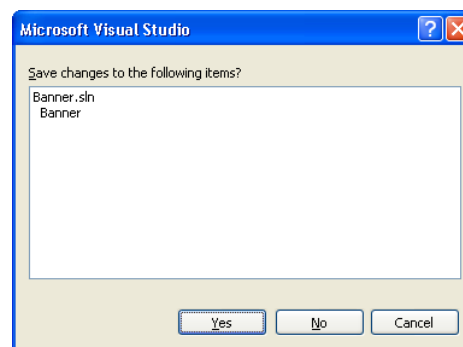
1. [Choose a problem set.](#)
2. [Choose a focus observation](#) in the problem set.
3. [Interpret the result data.](#)
4. [Resolve the issue.](#)
5. [Rebuild the target.](#)
6. [Rerun the analysis.](#)

Skip Managing the Memory Result Data

Examine an Intel® Parallel Inspector result when there are no threading and memory errors.

1. From the Visual Studio* menu, choose **File > Close Solution**.
2. Click the **Yes** button to save changes to the **Solution Explorer**.

NOTE: The Intel® Parallel Inspector automatically saves changes to My Inspector Results folders and files.



3. [Choose a new target](#): the **BannerFixed** solution/project.
4. [Build](#) the **BannerFixed** target.
5. [Configure and run a threading analysis.](#)
 - Notice the banner displays properly.
 - When you press any key to end target execution, the **Event Collection is Complete** dialog box shows no events.
 - When you click the **Interpret Result** button on the **Event Collection is Complete** dialog box, the **Overview** window shows no problem sets.
6. [Configure and run a memory analysis.](#)
 - Notice the banner displays properly.
 - When you press any key to end target execution, the **Event Collection is Complete** dialog box shows no events.
 - When you click the **Interpret Result** button on the **Event Collection is Complete** dialog box, the **Overview** window shows no problem sets.



7 Next Steps

This guide focuses on basic Intel® Parallel Inspector features. To explore more features and get the most out of the Intel® Parallel Inspector, check the following resources.

| Resource | Notes |
|--|--|
| <i>Intel® Parallel Inspector Documentation</i> | Use this HTML page to locate other Intel® Parallel Inspector resources, including a complete user guide. To open this HTML page, from the Windows* Start menu, choose Intel Parallel Studio > Intel Parallel Inspector > Intel Parallel Inspector Documentation . |
| <i>Samples</i> | Use sample code in the <code>samples</code> folder to learn how to interpret and handle different types of threading and memory errors. Read the associated <i>Sample Code Guide</i> available in the <code>documentation</code> folder. NOTE: The <i>Sample Code Guide</i> is not available in the Beta product. |
| <i>Intel® Parallel Studio resources</i> | Intel® Parallel Studio includes Intel® Parallel Advisor, Intel® Parallel Composer, Intel® Parallel Inspector, and Intel® Parallel Amplifier, providing the most comprehensive set of tools for parallelism. <ul style="list-style-type: none">• Intel® Parallel Advisor helps developers understand where to add parallelism to existing source code, including identifying hot spots and common data conflicts.• Intel® Parallel Composer speeds software development incorporating parallelism with a C/C++ compiler and comprehensive threaded libraries.• Intel® Parallel Amplifier helps developers fine-tune parallel applications for optimal performance for multicore processors by helping find unexpected serialization that limits scaling.• To open <i>Documentation</i> that points to more resources for each installed Intel® Parallel Studio product, from the Windows* Start menu, choose Intel Parallel Studio > <i>product name</i> > <i>product name</i> Documentation. |