

# Introducing the PurifyPlus Family:

PurifyPlus for Windows  
PurifyPlus for UNIX  
PurifyPlus for Linux  
PurifyPlus RealTime

Product version 2002 Release 2  
Document version 1.5  
Last revision: November 29, 2002

## High level overview:

- A. Intro – an introduction to the document
- B. Pains in software development
- C. Rational’s Runtime Analysis Solution
- D. Summary

## Table Of Contents:

- A) Intro
- B) Software development pains
  - 1. Reliability problems
  - 2. Performance and scalability problems
  - 3. Durability problems
- C) Rational’s solution for Runtime Analysis
  - 1. **What is Rational PurifyPlus?**
  - 2. **What runtime analysis capabilities are available in Rational PurifyPlus and on what platforms?**
    - 2.1 High level overview of runtime capabilities
  - 3. **What are the differences between PurifyPlus for Linux, PurifyPlus for UNIX and PurifyPlus for Windows?**
    - 3.1 Technology overview
      - 3.1.1 Technology for C/C++ applications
      - 3.1.2 Technology for Java applications
      - 3.1.3 Technology for Visual Basic applications
      - 3.1.4 Technology for .NET managed code applications
      - 3.1.5 User interface and product integrations
    - 3.2 What are the differences in features between PurifyPlus Family members per product?
    - 3.3 What are the differences in licensing models between the members of PurifyPlus Family?
- D) Summary

## **A) Intro:**

The purpose of this document is to introduce the Rational PurifyPlus Family of products. Some of you may have already gained experience with PurifyPlus on the UNIX and the Windows platforms, but now this Family has grown to other platforms: Linux (with PurifyPlus for Linux) and various other host and embedded targets (with PurifyPlus RealTime). The problems in software development that the PurifyPlus Family of tools help to solve are the same on all supported platforms; however, the approach to these problems may vary depending on the operating system. Even PurifyPlus for Windows and PurifyPlus for UNIX don't share the same scope of capabilities across their target platforms. In this document you will first find an introduction to runtime analysis and the description of pains in software development targeted with this practice. This introduction to runtime analysis is then accompanied with the high level overview of the technical concepts of the PurifyPlus Family, followed with a detailed, in-depth comparison of the Family products.

This document is created for all people who work with PurifyPlus on a daily basis, both within and outside Rational. If you have any questions about this document, please send an e-mail to [support@rational.com](mailto:support@rational.com) and we will try to help you.

Yours truly,

PurifyPlus Team

## B) Software development pains

**What are the pains in software development that can be addressed with the help of runtime analysis tools (differentiated by language when relevant)?**

Runtime analysis is the practice aimed at understanding application behavior using data collected during its execution. Often associated with debugging, runtime analysis can also be used with any variety of testing methods to proactively uncover and diagnose:

- Reliability problems
- Performance and scalability problems
- Durability problems

### 1. Reliability problems

The C/C++ programming language allow programmers to directly access and manipulate memory allocated by the application, thus creating an environment in which memory corruption errors and memory leaks can occur.

Examples of memory corruption errors are dangling pointers to freed memory, array bounds writes and reads, uninitialized memory reads, etc. Such corruption can lead to unpredictable, incorrect application behavior, typically ending in a crash.

Parts of an application that were not executed can hide memory and performance related problems, not to mention defects in functionality. In order for the application and its components to be thoroughly evaluated for all aspects of quality, all components of the application need to be exercised thoroughly. Execution paths for each use case simulated by a test suite must be clearly marked to ensure untested methods and uncovered lines of code are known.

What sequence of events occur as the application executes? Does it call the correct methods and do these methods execute in the correct order? What objects are created, when are they destructed, how do they interact, were there any exceptions raised? Only in trivial applications is it easy to follow the exact execution path and quickly spot the call chains that were not expected in the design of the application. Wrong execution paths can lead to both malfunctioning of the developed application and to performance and scalability problems.

### 2. Performance and scalability problems

Bad performance of an application is also known as the “ultimate bug”. Usually, bad performance is understood as the side effect of an inefficient algorithm in the application, but it can also happen due to some logical error (e.g. calling a certain method too often) or due to bad synchronization of the components of the application. The consequences of performance related problems can reflect in poor usability of the application, but also in malfunctioning, or

inability to accommodate a need for introducing new features, or to allow a larger number of users to access the application simultaneously. The software development activity associated to solving performance and scalability problems is often referred to as “profiling”. Successful profiling demands both a detailed and high-level understanding of how the application behaves, and how that differs from the way it was expected to behave.

Multithreading allows developers to split the execution of the application in a number of virtually parallel sequences that enable a much smoother and faster application. For example, your favorite office programs render newly selected spellcheck toolbars while you’re writing the text. However multithreading is demanding because, without care, the application can become the unsuspecting victim of intermittent hangs, data corruption or performance degradation due to deadlocks, race conditions and thread starvation.

### 3. Durability problems

Memory leaks occur when memory is allocated in the dynamic storage (heap) but is not explicitly released back to the system by the application, resulting in unnecessary memory overhead. This can slow performance due to virtual memory swapping/thrashing, and eventually cause unpredictable, incorrect application behavior when memory is exhausted.

In Java and .NET managed code, classic memory corruption leaks errors are not possible due to the automatic memory management (i.e. garbage collection) that controls all the memory allocation and cleans from memory all unused objects lacking that are no longer referenced. Furthermore, in Java and .NET managed applications, it is not possible to directly access the objects and structures in memory thus preventing memory access corruption errors like array bounds violations (ABW – Array Bounds Write or ABR – Array Bounds Read), known to occur in native C/C++ applications. However, a developer can mistakenly leave a reference to an object in memory while believing that this object will be cleared. Such an error can result in excessive memory overhead and is therefore referred to as a memory leak as well, with the same results: degraded performance and eventual incorrect operation.

## **C) Rational’s solution for Runtime Analysis**

### **1. What is Rational PurifyPlus?**

PurifyPlus is a Family of Rational solutions providing runtime analysis functionality to software developers and testers. The PurifyPlus Family consists of the following members:

- Rational PurifyPlus for UNIX
- Rational Purify for UNIX
- Rational Quantify for UNIX
- Rational PureCoverage for UNIX

- Rational PurifyPlus for Windows
- Rational Purify for Windows
- Rational Quantify for Windows
- Rational PureCoverage for Windows
  
- Rational PurifyPlus for Linux
- Rational PurifyPlus RealTime

PurifyPlus for Linux and PurifyPlus RealTime are the newest members of the PurifyPlus Family, first released with v2002 Release 2 in August 2002.

The idea behind the Rational PurifyPlus Family is to provide the best and the most complete runtime analysis solution available on the market. The development activities covered with the PurifyPlus Family of products are the following:

- Memory corruption detection and memory profiling in native C/C++ applications
- Memory profiling in Java and .NET managed code applications
- Performance Profiling

The development activity that deals with isolating and analyzing the performance hotspots of an application with the intention of removing dominant hotspots and improving the overall performance of the application is called performance profiling. This activity answers questions like: "Where is my application slow?" and helps answer questions like "Why is it slow?" and "How can I improve the performance?".

- Thread Profiling

Thread profiling inspects and prevents intermittent hangs, data corruption or performance degradation due to deadlocks, race conditions and thread starvation. It is important to synchronize the threads carefully and to find a way to detect and analyze multithreading conditions as they occur.

- Code coverage analysis
- Runtime tracing

## **2. What runtime analysis capabilities are available in the Rational PurifyPlus Family of products, sorted by platform?**

2.1 High level overview of runtime capabilities:

Terms used in the table:

.NET – managed code, e.g. Visual C#, Visual Basic.NET and managed Visual C++ 7

Java – Java 2 SE SDKs

VB – Visual Basic 6.0 p-code and native compiled VB executables

Legend: + Feature available

- Feature not available

n/a - not applicable

		Windows	Solaris	HP-UX	Linux	IRIX	AIX*	Embedded
<b>Memory corruption detection</b>	<i>C/C++</i>	+	+	+	+ **	+	+	+
<b>Memory leak detection</b>	<i>C/C++</i>	+	+	+	+	+	+	+
	<i>Java</i>	+	-	-	+	-	+	+
	<i>.NET</i>	+	n/a	n/a	n/a	n/a	n/a	n/a
<b>Performance profiling</b>	<i>C/C++</i>	+	+	+	+	-	+	+
	<i>Java</i>	+	+	-	+	-	+	+
	<i>.NET</i>	+	n/a	n/a	n/a	n/a	n/a	n/a
	<i>VB</i>	+	n/a	n/a	n/a	n/a	n/a	n/a
<b>Thread analysis</b>	<i>C/C++</i>	+	-	-	+	-	+	+
	<i>Java</i>	+	-	-	+	-	+	+
	<i>.NET</i>	+	n/a	n/a	n/a	n/a	n/a	n/a
	<i>VB</i>	+	n/a	n/a	n/a	n/a	n/a	n/a
<b>Runtime Tracing</b>	<i>C/C++</i>	-	-	-	+	-	+	+
	<i>Java</i>	-	-	-	+	-	+	+
	<i>.NET</i>	-	n/a	n/a	n/a	n/a	n/a	n/a
<b>Code coverage</b>	<i>C/C++</i>	+	+	+	+	NO	+	+
	<i>Java</i>	+	YES	-	+	NO	+	+
	<i>.NET</i>	+	n/a	n/a	n/a	n/a	n/a	n/a
	<i>VB</i>	+	n/a	n/a	n/a	n/a	n/a	n/a

\*IBM AIX platform is supported in PurifyPlus RealTime. Although P+RealTime is designed for embedded targets, it can also support native software testing as well – necessary, for example, if the customer wants to test his/her code within a simulator. Rational does not have a PurifyPlus solution explicitly for the AIX platform, but since P+RealTime supports AIX as an install platform, P+RealTime can be used to test native applications on AIX as well.

\*\* PurifyPlus for Linux detects a subset of memory corruption errors detected by Purify (Windows), Purify (UNIX), PurifyPlus for Windows and PurifyPlus for UNIX

**3. Rational has four PurifyPlus editions, each targeting a particular host environment. Although all tackle runtime analysis, the methods used are not always the same from one product to the next. What are the differences between PurifyPlus for Linux, PurifyPlus RealTime, PurifyPlus for UNIX and PurifyPlus for Windows?**

PurifyPlus for Linux and PurifyPlus RealTime share the same code base, the same user interface and provide the same set of capabilities. PurifyPlus for UNIX, PurifyPlus for Windows and PurifyPlus for Linux/RealTime do NOT share the same code base, nor the same user interface nor the same set of capabilities on their respective platforms. All, however, help overcome the pains of software development. Here is the in-depth overview of features of the different PurifyPlus Family members:

Glossary:

<i>Runtime analysis:</i>	A practice aimed at understanding application behavior using data collected during its execution
<i>PUT:</i>	Program Under Test – the application being examined by the runtime analysis tool
<i>Instrumentation:</i>	The process of inserting additional instructions into source code, object files, executable images, or byte streams of the tested PUT in order to collect data from the instrumented application PUT during runtime
<i>OCI:</i>	Object Code Insertion – Rational patented technology for performing runtime analysis of native applications. The instrumentation is performed on object files or executable files, depending on the platform, and doesn't require source code or re-compilation
<i>BCI:</i>	Byte Code Insertion – Rational's technology of instrumenting Java byte code and .NET Intermediate Language with the purpose of collecting runtime analysis data during the execution of the instrumented code.
<i>SCI:</i>	Source Code Insertion – runtime analysis technology that instruments source files; requires compilation.
<i>TDP:</i>	Target Deployment Port – Enables usage of runtime analysis functionality on embedded targets
<i>JVM:</i>	Java Virtual Machine
<i>J2ME:</i>	Java 2 Micro Edition
<i>J2SE:</i>	Java 2 Standard Edition
<i>J2EE:</i>	Java 2 Enterprise Edition
<i>JVMPI:</i>	Java Virtual Machine Profiling Interface – A public interface to the JVM that enables collection of runtime analysis data for Java programs running through the JVM
<i>.NET CLR:</i>	.NET Common Language Runtime

3.1 Technology overview:

<b>PurifyPlus for UNIX</b>	<b>PurifyPlus for Windows</b>	<b>PurifyPlus for Linux</b>	<b>PurifyPlus RealTime</b>
Rational's runtime analysis solution for UNIX operating systems. Based on OCI and BCI technology.	Rational's runtime analysis solution for Windows operating systems. Based on OCI and BCI technology.	Rational's runtime analysis solution for Linux operating systems. Based on SCI technology.	Rational's runtime analysis solution for embedded development environments and target platforms. Based on SCI technology.
<p><i>Supported development platforms:</i>            SPARC Solaris 2.6, 7, 8, 9            PA-RISC HP-UX 10.20, 11.0, 11.i(11.11)</p>	<p><i>Supported development platforms:</i>            Intel x86 Windows NT4 sp6a            Windows 2000 (all)            Windows XP Professional</p>	<p><i>Supported development platforms:</i>            Intel x86 Red Hat Linux 7.0, 7.2, 7.3            SuSe Linux 7.2, 7.3</p>	<p><i>Supported development platforms:</i>            Intel x86 Windows NT4 sp6, Windows 2000 Pro, Windows XP Pro            SPARC Solaris 2.6, 7, 8            PA-RISC HP-UX 10.2, 11.i            PowerPC AIX 4.3, 5L (v5.1)            Intel x86 Red Hat Linux 7.0, 7.2, 7.3            SuSe Linux 7.2, 7.3</p>
<p><i>Supported target platforms:</i>            The same as the listed development platforms.</p>	<p><i>Supported target platforms:</i>            The same as the listed development platforms.</p>	<p><i>Supported target platforms:</i>            The same as the listed development platforms.</p>	<p><i>Supported target platforms:</i>            Wide range of target platforms available through custom TDPs including IBM AIX and 8- to 64-bit runtimes. For more information about the TDP technology, check the PurifyPlus RealTime User Guide</p>

			and Reference Manual.
<b>3.1.1. Technology for C/C++ applications</b>			
<b>PurifyPlus for UNIX</b>	<b>PurifyPlus for Windows</b>	<b>PurifyPlus for Linux</b>	<b>PurifyPlus RealTime</b>
Instruments object files, requires linking. The whole PUT has to be instrumented, no selective instrumentation available.	Instruments executable files, doesn't require re-linking, or recompiling of the PUT. For performance profiling and code coverage instrumentation type can be changed per executable module (DLL, EXE, OCX, VBX). Selective instrumentation per executable module available for performance profiling and code coverage.	Instruments source files. Requires compiling and linking of the PUT. Selective instrumentation per source file.	Instruments source files. Requires compiling and linking of the PUT. Selective instrumentation per source file.
<i>Supported compilers:</i> On Sun: through Sun ONE Studio and Forte 7 (CC 5.4) and GCC 3.1 On HP-UX: cc/aCC 3.31 3.33 and 3.34, GCC through 3.1, GNUPro 98r2 On IRIX: SGI compiler through 7.3	<i>Supported compilers:</i> Visual C++ 6.0 Visual C++ 7.0	<i>Supported compilers:</i> GNU C/C++ 2.95 (SuSe), GNU C/C++ 2.96 (Red Hat)	<i>Supported compilers:</i> All ANSI-compliant C/C++ compilers
Full 64 bit support for memory corruption detection memory	Technology preview of PurifyPlus for 64-bit Windows available.	Not supported	Supports 64bit runtimes through custom TDPs

leak detection and performance profiling for Solaris 7, 8, 9 and HP-UX 11.0 and above. (Code coverage on Solaris 32bit only)			
<b>3.1.2. Technology for Java applications</b>			
<b>PurifyPlus for UNIX</b>	<b>PurifyPlus for Windows</b>	<b>PurifyPlus for Linux</b>	<b>PurifyPlus RealTime</b>
JVMPI and BCI based data collection. Pre-filtering per Java package, selective instrumentation in Beta. Supports J2SE and J2EE.	JVMPI and BCI based data collection. Pre-filtering per Java package, selective instrumentation Beta. Supports J2SE and J2EE.	Selective source code instrumentation for performance profiling, runtime tracing and code coverage analysis / JVMPI based data collection for memory profiling.	Selective source code instrumentation for performance profiling, runtime tracing and code coverage analysis / JVMPI based data collection for memory profiling.
<i>Supported JVMs:</i> <b>Java 2 compatible JVMs</b> <ul style="list-style-type: none"> <li>• J2SE SDK 1.3.1 and later</li> <li>• J2EE SDK 1.3.1 and later</li> </ul>	<i>Supported JVMs:</i> <b>Java 2 compatible JVMs</b> <ul style="list-style-type: none"> <li>• J2SE SDK 1.2.2 and later</li> <li>• J2EE SDK 1.2.2. and later</li> </ul>	<i>Supported JVMs:</i> <b>Java 2 compatible JVMs</b> <ul style="list-style-type: none"> <li>• J2SE SDK 1.3.1 and later</li> </ul>	<i>Supported JVMs:</i> <b>Java 2 compatible JVMs</b> <ul style="list-style-type: none"> <li>• J2ME SDK 1.3.1 and later</li> <li>• J2SE SDK 1.3.1 and later</li> </ul>
<i>Supported Java Application servers:</i> <ul style="list-style-type: none"> <li>• IBM WebSphere 4</li> <li>• BEA Web Logic 6.x and later</li> <li>• Apache Tomcat 4.x and later</li> </ul>	<i>Supported Java Application servers:</i> <ul style="list-style-type: none"> <li>• IBM WebSphere 4</li> <li>• BEA Web Logic 6.x and later</li> <li>• Apache Tomcat 4.x and later</li> </ul>	<ul style="list-style-type: none"> <li>• Not supported</li> </ul>	<ul style="list-style-type: none"> <li>• Not supported</li> </ul>
<b>3.1.3. Technology for Visual Basic applications</b>			
<b>PurifyPlus for UNIX</b>	<b>PurifyPlus for Windows</b>	<b>PurifyPlus for Linux</b>	<b>PurifyPlus RealTime</b>

Not applicable.	OCI for native compiled applications. Performance profiling and code coverage of p-code through the VB runtime profiling interface.	Not applicable.	Not applicable.
<b>3.1.4. Technology for .NET managed code applications</b>			
<b>PurifyPlus for UNIX</b>	<b>PurifyPlus for Windows</b>	<b>PurifyPlus for Linux</b>	<b>PurifyPlus RealTime</b>
Not applicable.	Data collection through .NET CLR profiling interfaces plus BCI. ASP.NET Profiling Agent for ASP.NET Web Services.	Not applicable.	Not applicable.
<b>3.1.5. User interface and product integrations</b>			
<b>PurifyPlus for UNIX</b>	<b>PurifyPlus for Windows</b>	<b>PurifyPlus for Linux</b>	<b>PurifyPlus RealTime</b>
Stand-alone GUI, command line.	Stand-alone GUI, command line. Full Visual Studio 6 and Visual Studio.NET integration.	Stand-alone GUI, command line.	Stand-alone GUI, command line. Various IDE integrations including Microsoft Visual Studio 6 and TI Code Composer Studio 2.0 and 2.1.
Code coverage data can be collected together with memory corruption and memory leak data	Code coverage can be collected together with memory corruption and memory leak data	Simultaneous data collection of memory leak, performance, code coverage and runtime trace data.	Simultaneous data collection of memory leak, performance, code coverage and runtime trace data.
Integrations: •Rational ClearCase •Rational ClearQuest	Integrations: •ClearCase •ClearQuest	Integrations: •ClearCase •ClearQuest	Integrations: •ClearCase •ClearQuest

	<ul style="list-style-type: none"> <li>•Rational Robot</li> <li>•Rational TestFactory</li> <li>•Rational VisualTest</li> <li>•Rational Rose RealTime</li> </ul>		<ul style="list-style-type: none"> <li>•Rose RealTime</li> <li>•TestManager</li> </ul>
Data files can be saved as proprietary binary formatted files, or ASCII files.	Data files can be saved as proprietary binary formatted files, or ASCII files.	Data files can be saved in HTML format.	Data files can be saved in HTML format.

### 3.2 [What are the differences in features between PurifyPlus Family members per product?](#)

Legend: + Feature available  
 - Feature not available  
 n/a - not applicable

		Memory corruption detection	Memory leak detection	Performance profiling	Thread analysis***	Runtime Tracing	Code coverage
PurifyPlus for Windows	<i>C/C++</i>	+	+	+	-	-	+
	<i>Java</i>	n/a	+	+	+	-	+
	<i>.NET</i>	n/a	+	+	+	-	+
	<i>VB6</i>	n/a	-	+	+	-	+
PurifyPlus for UNIX	<i>C/C++</i>	+	+	+	-	-	+
	<i>Java*</i>	n/a	+	+	-	-	+
PurifyPlus for Linux	<i>C/C++</i>	***	+	+	+	+	+
	<i>Java</i>	n/a	+	+	+	+	+
PurifyPlus RealTime	<i>C/C++</i>	***	+	+	+	+	+
	<i>Java</i>	n/a	+	+	+	+	+
Purify (Windows)	<i>C/C++</i>	+	+	-	-	-	-
	<i>Java</i>	n/a	+	-	+	-	-
	<i>.NET</i>	n/a	+	-	+	-	-
	<i>VB6</i>	n/a	-	-	-	-	-
Purify (UNIX)	<i>C/C++</i>	+	+	-	-	-	-

Quantify (Windows)	<i>C/C++</i>	-	-	+	+	-	-
	<i>Java</i>	-	-	+	+	-	-
	<i>.NET</i>	-	-	+	+	-	-
	<i>VB6</i>	-	-	+	+	-	-
Quantify (UNIX)	<i>C/C++</i>	-	-	+	-	-	-
	<i>Java*</i>	-	-	+	-	-	-
PureCoverage (Windows)	<i>C/C++</i>	-	-	-	-	-	+
	<i>Java</i>	-	-	-	-	-	+
	<i>.NET</i>	-	-	-	-	-	+
	<i>VB6</i>	-	-	-	-	-	+
PureCoverage (UNIX)	<i>C/C++</i>	-	-	-	-	-	+
	<i>Java*</i>	-	-	-	-	-	+

\* PurifyPlus for UNIX, Quantify (UNIX) and PureCoverage (UNIX) support Java on Sun Solaris only

\*\*PurifyPlus for Linux detects a subset of memory corruption errors detected by Purify (Windows), Purify (UNIX), PurifyPlus for Windows and PurifyPlus for UNIX

\*\*\*The entire PurifyPlus Family of products supports multithreaded applications. Thread profiling functionality refers to the ability to visualize running threads

### **3.3 What are the differences in licensing models between the members of PurifyPlus Family?**

Each PurifyPlus edition within the PurifyPlus Family comes with its own product license. There are four main products each coming with its own license. P+UNIX, P+Linux and P+RealTime are available with floating license, or named user license, P+Windows comes with floating license, or node locked license. The prices for the licenses defer differ between the main PurifyPlus editions:

Main product	Floating license	Named user license	Node locked license
Rational PurifyPlus for UNIX license	YES	YES	NO
Rational PurifyPlus for Windows license	YES	NO	YES
Rational PurifyPlus for Linux license	YES	YES	NO
Rational PurifyPlus RealTime license	YES	YES	NO

The constituent pPoint products are still available on UNIX and on Windows and they are available with named user license on UNIX and with node locked license on Windows.

Point product	Floating license	Named user license	Node locked license
Rational Purify for UNIX	NO	YES	NO
Rational Quantify for UNIX	NO	YES	NO
Rational PureCoverage for UNIX	NO	YES	NO
Rational Purify for Windows	NO	NO	YES
Rational Quantify for Windows	NO	NO	YES
Rational PureCoverage for Windows	NO	NO	YES

#### D) Summary:

Rational PurifyPlus Family is a set of tools used to perform runtime analysis. Each member of the Family has a unique set of specifications and strengths. There is one common line connecting all the Family members, however: on every platform, Rational PurifyPlus is the leading industrial-strength application for runtime analysis and we believe that the PurifyPlus package is stronger than any competitive product available out there. As this new release shows, we are continually working to evolve and grow our runtime analysis product line, producing a feature set that surpasses the competition for the capture and resolution best addresses development pains.

If you have more questions, or if you would just like to send us your feedback on this document or on the PurifyPlus Family in general, please send an e-mail to [support@rational.com](mailto:support@rational.com) and we will try to answer your question within 24 hours.

Thank you!

The Rational PurifyPlus team