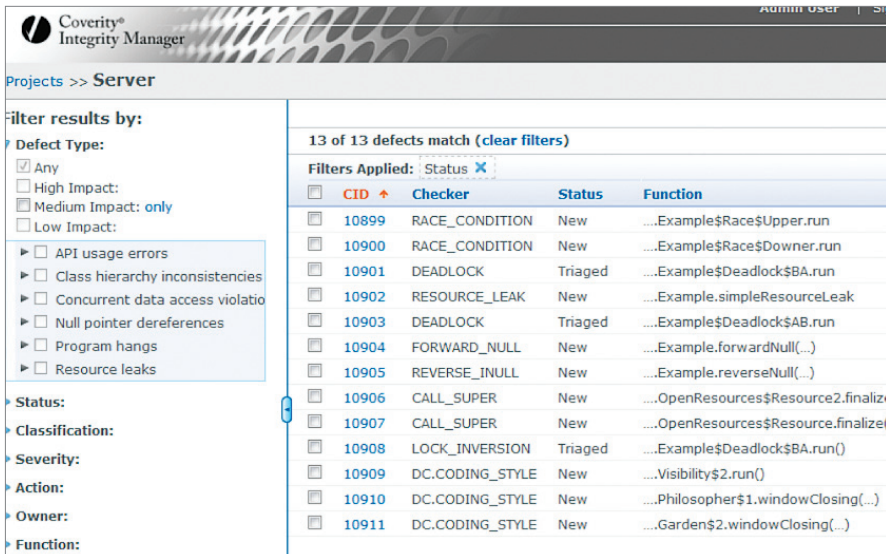


# Coverity® Dynamic Analysis

Coverity Dynamic Analysis helps developers, QA, and test engineers quickly identify hard to diagnose defects in multi-threaded Java applications. With minimal impact to your team or test environment, Coverity Dynamic Analysis automatically instruments Java programs and provides reliable, accurate, and reproducible detection of concurrency errors that could result in performance degradation, system crashes, or security vulnerabilities.



**Coverity® Integrity Manager**

Projects >> Server

**filter results by:**

**Defect Type:**

- Any
- High Impact:
- Medium Impact: only
- Low Impact:
- API usage errors
- Class hierarchy inconsistencies
- Concurrent data access violation
- Null pointer dereferences
- Program hangs
- Resource leaks

**Status:**

**Classification:**

**Severity:**

**Action:**

**Owner:**

**Function:**

13 of 13 defects match (clear filters)

Filters Applied: Status X

<input type="checkbox"/>	CID	Checker	Status	Function
<input type="checkbox"/>	10899	RACE_CONDITION	New	...Example\$Race\$Upper.run
<input type="checkbox"/>	10900	RACE_CONDITION	New	...Example\$Race\$Downer.run
<input type="checkbox"/>	10901	DEADLOCK	Triaged	...Example\$Deadlock\$BA.run
<input type="checkbox"/>	10902	RESOURCE_LEAK	New	...Example.simpleResourceLeak
<input type="checkbox"/>	10903	DEADLOCK	Triaged	...Example\$Deadlock\$AB.run
<input type="checkbox"/>	10904	FORWARD_NULL	New	...Example.forwardNull(...)
<input type="checkbox"/>	10905	REVERSE_INULL	New	...Example.reverseNull(...)
<input type="checkbox"/>	10906	CALL_SUPER	New	...OpenResources\$Resource2.finalize
<input type="checkbox"/>	10907	CALL_SUPER	New	...OpenResources\$Resource.finalize
<input type="checkbox"/>	10908	LOCK_INVERSION	Triaged	...Example\$Deadlock\$BA.run()
<input type="checkbox"/>	10909	DC.CODING_STYLE	New	...Visibility\$2.run()
<input type="checkbox"/>	10910	DC.CODING_STYLE	New	...Philosopher\$1.windowClosing(...)
<input type="checkbox"/>	10911	DC.CODING_STYLE	New	...Garden\$2.windowClosing(...)

Increase accuracy of analysis with a tightly-coupled integration of Dynamic Analysis with Static Analysis for your Java applications.

## Key Features

### Best of Breed Analysis

Coverity Dynamic Analysis analyzes your Java program while it runs to find concurrency defects such as race conditions, deadlocks, and resource leaks. These kinds of defects are difficult to find and time-consuming to reproduce using manual debugging.

### Integration with Coverity Static Analysis

By combining static analysis and dynamic analysis techniques, and providing the developers with an integrated view, increase the accuracy and speed of defect detection to achieve the most thorough analysis of race conditions, deadlocks, and resource leaks.

### Business Impact Mapping

Industry's first capability to automatically map the impact of a defect across the entire codebase, alerting you of the presence of a single defect in all projects and products that share code.

Developers can quickly identify the impact of a defect from one part of the code on the entire product portfolio. Development managers and executives have actionable information to make better fix/no fix decisions based upon impact to a single project, across all projects, across the product portfolio, and to the business, reducing the risk of schedule slips and quality issues across products.

## Types of Concurrency Defects Identified

**Race Conditions** that can cause incorrect application behavior and introduce security vulnerabilities in multi-threaded applications

**Deadlocks** that typically result when two Java threads wait for each other to release a lock, or more than two Java threads wait for locks in a circular chain

**Resource Leaks** that can result in unnecessary failures and bad performance

### Defect Understanding

An easy-to-understand defect description, a mapping to the Common Weakness Enumeration (CWE), intuitive and precise navigation and inline expansion of function calls for every defect allows improved developer efficiency when understanding the root issue of the error.

### Prioritization

Checker classification helps you easily prioritize defects by combining checkers into categories based upon how a defect manifests into issues. With information on the priority and type of error, developer efficiency is improved by spending less time on researching the defects and instead on fixing the critical priority defects first.

**Key Features** *continued*

**Code Components Mapping**

Break up large codebases into logical parts according to directories, libraries, developers and groups which allows automatic defect assignment and prioritization of code areas with high defect density.

**Defect Reporting**

Evaluating and measuring defect history and resolution status at the branch level, the project level, and across projects is critical to make better decisions and measure developer productivity and quality improvement over time. Reporting allows you to answer critical questions such as which defects have been fixed, have all critical defects been addressed and what is the trend in the software quality trend over time.

**Ease and Flexibility of Use**

Coverity Dynamic Analysis can be easily integrated into your existing development and testing environment, from a subset of code to the entire application, depending on usage. Developers can run ad hoc analysis on specific code changes and to debug concurrency issues. QA and Build Managers can run automated tests at the tail end of the build process to identify defects that may have gone undetected during development.

**Technical Specifications**

Coverity Dynamic Analysis			
Supported Platforms	System Requirements	Supported Java Environments	IDE Plug-In Support
<ul style="list-style-type: none"> <li>Windows XP, Vista, and Server</li> <li>Linux (32-bit and 64-bit)</li> <li>Mac OS X 10.5</li> </ul>	<ul style="list-style-type: none"> <li>1 GHz CPU (x86 or SPARC)</li> <li>2 GB of RAM minimum 4 GB recommended</li> <li>1 GB of free hard disk space</li> </ul>	<ul style="list-style-type: none"> <li>Sun JRE 1.5, 1.6</li> </ul>	<ul style="list-style-type: none"> <li>Eclipse 3.3 and later</li> </ul>
Coverity Integrity Manager Browser Support			
Browser	Flash		
<ul style="list-style-type: none"> <li>Internet Explorer 7, 8</li> <li>Firefox 3.5</li> </ul>	<ul style="list-style-type: none"> <li>Flash 10 with JavaScript enabled</li> </ul>		

**For More Information:**  
sales@coverity.com

**Coverity Inc. Headquarters**  
185 Berry Street, Suite 1600  
San Francisco, CA 94107 USA

U.S. Sales: (800) 873-8193  
International Sales: +1 (415) 321-5237  
[www.coverity.com](http://www.coverity.com)

