Logging Wrapper 1.4 Requirement Specification

1. Scope

1.1 Overview

The Logging Wrapper component provides a standard logging API with a pluggable back-end logging implementation. This update version includes some enhancements as well as the refactoring to remove the configuration management dependency.

1.2 Logic Requirements

1.2.1 Refactoring

1.2.1.1 To reduce dependencies among components, the component should be refactored to remove the dependency on Configuration Manager. Instead, object creation should be done programmatically.

1.2.2 Performance Enhancements

1.2.2.1 To improve logging performance, the component should provide additional APIs that allow the message formatting to be delayed to only when the message will be logged. In the current version (1.3), in many cases when the log(Level level, Object message) call is made, the message string is already formatted even though the message will end up not being logged at all. A solution to this is to provide a format message with a list of arguments, and format the message only when the message will be actually logged.

1.2.3 Exception Logging

1.2.3.1 The component should provide additional APIs to log exception stack traces, similar to the functionality provided in log4j and JDK logger.

1.3 Required Algorithms

None.

1.4 Example of the Software Usage

The Logging Wrapper component is used by other components. This allows the components to be plugged into an existing environment without requiring the additional configuration and implementation of a specific logging solution.

1.5 Future Component Direction

Additional logger implementations may be added in the future.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

None.

2.1.3 Environment Requirements

- Development language: Java 1.4
- Compile target: Java 1.4 and Java 1.5

2.1.4 Package Structure

com.topcoder.util.log

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

- None.

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

None.

3.2.2 TopCoder Software Component Dependencies:

**Please review the TopCoder Software component catalog for existing components that can be used in the design.

3.2.3 Third Party Component, Library, or Product Dependencies:

None

3.2.4 QA Environment:

- Solaris 9
- RedHat Linux Enterprise 4
- Windows XP, 2003

3.3 Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in the TopCoder Software Component Guidelines. Modifications to these guidelines for this component should be detailed below.

3.4 Required Documentation

3.4.1 Design Documentation

- Use-Case Diagram
- Class Diagram
- Sequence Diagram
- Component Specification

3.4.2 Help / User Documentation

- Design documents must clearly define intended component usage in the 'Documentation' tab of Poseidon.