Requirements Specification

1. Scope

1.1 Overview

The Executable Wrapper component provides the ability to execute programs and scripts outside the scope of the JVM. This functionality is available in Java but is often implemented incorrectly. This component masks the implementation details of executing external code while giving developers an increased level of control and details about those programs during execution.

1.2 Logic Requirements

1.2.1 Executable Control

- Provide a simplified mechanism for running executables outside of the JVM.
- Provide "administration" methods for controlling and obtaining details about the individual processes. Desired controls include start and stop. Details include program output and return code.

1.2.2 OS Support

- Define additional OS specific scripts that aide in executable control and obtaining process information.
- Require OS support includes:
  - Solaris 7
  - RedHat Linux 7.1
  - Windows 2000

1.3 Example of the Software Usage

Process Control:
Most applications have support scripts that are executed manually or by "cron" on a periodic basis. Providing a simplified executable wrapper will enable developers to easily integrate the execution of these scripts within the supported application.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

None Defined.

2.1.3 Environment Requirements

- Development language: Java1.4
- Compile targets: Java1.2, Java1.3 and Java1.4

2.1.4 Package Structure

com.topcoder.util.exec

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

None. If configuration parameters need to be read from a file, use the Configuration Manager component.

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

None.

3.2.2 Third Party Component, Library, or Product Dependencies:

None.

3.2.3 QA Environment:

- Solaris 7
- RedHat Linux 7.1
- Windows 2000

3.3 Design Constraints

3.3.1 Development Standards:

- It is required that all code be clearly commented.
- All code must adhere to javadoc standards, and, at minimum, include the @author, @param, @return, @throws and @version tags.
  - When populating the author tag, use your TopCoder member handle. In addition, please do not put in your email addresses.
  - Copyright tag: Copyright © 2002, TopCoder, Inc. All rights reserved
- For standardization purposes, code must use a 4-space (not tab) indentation.
- Do not use the ConfigurationManager inside of EJB's. The usage of the java.io.* package to read/write configuration files can potentially conflict with a restrictive security scheme inside the EJB container.
- All code must adhere to the Code Conventions outlined in the following: http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html

3.3.2 Database Standards:

- Table and column names should be as follows: table_name, not tableName.
- Every table must have a Primary Key.
- Always use Long (or database equivalent) for the Primary Key.
  For maximum portability, database objects (table names, indexes, etc) should be kept to 18 characters or less. Informix and DB2 have an 18-character limit.

3.4 Required Documentation

3.4.1 Design Documentation

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

3.4.2 Help / User Documentation

- Javadocs must provide sufficient information regarding component design and usage.