Auto Screening Tool Requirements Specification

1. Scope

1.1 Overview

Automated screening is a tool that screens submissions automatically. Submissions are put in a queue and will be picked up by standalone screening tools that could potentially run from multiple servers. Screening rules are configurable per project category. Once the submission is screened, the results will be logged.

This component provides the standalone screeners.

1.2 Logic Requirements

1.2.1 Overview

This release aims to integrate the existing automated screening with the new online review application. Compatibility is not required. Source codes for the existing system will be provided. Documentation should be produced according to the latest standards.

1.2.2 Automated Screener

Automated screeners can be run from multiple servers concurrently. Each individual screener will be configured with a different screener identifier. The component must ensure that no two screeners will pick the same task. When a screener starts, it should reset all the tasks for this screener with screening status to pending. These tasks might be left over from a server failure.

Only command line interface is required. The screener should use the Job Scheduler to configure the running interval.

1.2.3 Task Selection

The component will load all the pending tasks and use a pluggable mechanism to pick the task to be executed. The default implementation should prioritize the screening task with the earliest creation timestamp.

After a task is selected, the screener will update the task with its screener identifier and start timestamp. The status should be updated to screening. If at this point the task is picked by other screener, it should go back to select another task for execution.

If the task is successfully picked, it will be downloaded from the File System Server.

1.2.4 Screening Rules

Screening rules should be pluggable. Each submission will be screened by a set of rules according to the project category. This release should provide the rules from the previous version.

1.2.5 Screening Response

Screening responses should be logged with configurable mechanism. The default implementation should log each response into database. The ID's should be generated with ID Generator.

1.2.6 Screening Result

After all the responses are produced, the screening status will be changed to either failed, passed or passed with warning.

1.2.7 Logging

Additional logging must be performed through the Logging Wrapper. The minimum requirement includes the task selected, the task finished (with the final status) and each error the screener

encounters. The screener will not halt on errors from the screening rules.

1.2.8 Persistence

Persistence needs to be pluggable. For this release an Informix plug-in will be developed. The SQL scripts will be provided.

1.2.8.1 Persistence Implementation

The persistence implementation needs to be designed in this component, but will be separated into a second development project. Please put all persistence implementation related information into a separate sub-package and clearly mark the responsibilities of the two development projects.

1.3 Required Algorithms

No specific algorithms are required.

1.4 Example of the Software Usage

The screeners will be run in order to support the Online Review application. Once the screening tasks are initiated, they will be picked up by the screener so that the user will be able to view the screening details when it's complete.

1.5 Future Component Direction

None.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

None.

2.1.3 Environment Requirements

- Development language: Java1.4
- Compile target: Java1.4

2.1.4 Package Structure

com.cronos.onlinereview.autoscreening.tool

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

- Database connection
- Screener identifier
- Task selection mechanism
- Screening rules per project category

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

JDBC.

3.2.2 TopCoder Software Component Dependencies:

- Configuration Manager
- DB Connection Factory
- ID Generator
- Job Scheduler

- User Project Data Store
- Compression Utility
- Data Validation
- Directory Validation
- Executable Wrapper
- File Class
- Generic Event Manager
- Magic Numbers
- XMI Parser
- Logging Wrapper

**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 7
- RedHat Linux 7.1
- Windows 2000
- Windows 2003
- Informix 10.0

3.3 Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in the TopCoder Software Component Guidelines.

3.3.1 Database Connections

Database connections must not be cached within the component. Connections should be created for each operation and closed afterwards.

3.3.2 Component Scalability

The component needs to be scalable. Running multiple instances in the same JVM or in multiple JVM's concurrently should not cause any problem.

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.