Requirements Specification

1. Scope

1.1 Overview

This component will provide compatibility with configuration manager persistence files (XML and properties) via the new preferred Configuration API. It is expected that only the application will need to directly access this component, and that components used by the application will be configured using instances of the Configuration Object interface defined by the Configuration API component.

1.2 Logic Requirements

1.2.1 Read and Write Configuration Manager XML Files

The component will be able to read XML files and convert the namespaces, properties, and values into Configuration API Configuration Object instances. Likewise the instance should persist to these files when needed.

1.2.2 Read and Write Configuration Manager Property Files

Same requirement as 1.2.1, except for CM property files.

1.2.3 File Path or Classpath Loading of Configuration Files

A configuration file can be found via file path or via classpath. File paths are specified relative to the application installation to prevent accessing restricted files or violating J2EE container rules.

1.2.4 Default Constructor

The default constructor should read it's own file via the classpath. Other configuration files can be read in from the configuration file, but this functionality does not have to work exactly like CM's related functionality.

1.2.5 Alternative Constructor

Alternatively an application can pass in a Configuration Object instance or configure this component programmatically by passing in an argument list.

1.2.6 Multiple Instances

Since this component is intended to be managed by the application and not other components, the singleton pattern will not be followed. The component should provide an intuitive manner for handling multiple instances colliding on the same file.

1.3 Required Algorithms

None

1.4 Example of the Software Usage

An existing application uses the CM component and it (along with its components) is being upgraded to use the Configuration API approach instead. This component will be used so that the configuration files will not require a complete rewriting. Instead a single configuration file for this component will be created that is used for configuring this component, and loading all the component and application configuration files. The application will pass in the path to the configuration file (or the configuration file will be in the default location) to instantiate an instance of this component. At that point it can access all configuration data via the Configuration Object instances and pass those to the correct components.

1.5 Future Component Direction

In the future extension to the XML or properties format may be made to take further advantage of the Configuration APIs capabilities. It is also possible that a future version supports other persistence formats, but it's probably just as likely other persistence formats get their own component.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None

2.1.2 External Interfaces

2.1.2.1 Configuration Manager's XML files

2.1.2.2 Configuration Manager's property files

2.1.3 Environment Requirements

- Development language: Java1.4
- Compile target: Java1.4, 1.5

2.1.4 Package Structure

com.topcoder.configuration.persistence

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

- Location of the component's configuration file.
- Other configuration files that need to be read.

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

XML, Schema/DTD, XPath

3.2.2 TopCoder Software Component Dependencies:

Configuration API 1.0
Configuration Manager – only for the file formats, do not use the CM directly

3.2.3 Third Party Component, Library, or Product Dependencies:

Xerces

3.2.4 QA Environment:

- Solaris 7
- RedHat Linux 7.1
- Windows 2000
- Windows 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
- **XML Schema's for the XML files – these may be enhanced from the CM version, as long as CM version XML files are still valid**

3.4.2 Help / User Documentation

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