File Upload 2.0 Requirements Specification

1. Scope

1.1 Overview

The File Upload component supports web-based file uploads. Files are received via HTTP requests. Applications implementing the component have the ability to store uploaded files on the file system.

Storing the uploaded files to the file system may not work in a clustered environment. The file will not be available if user's subsequent requests are served from another node. This enhancement of the component provides the option to make the uploaded file available from the File System Server.

1.2 Logic Requirements

1.2.1 Compatibility

Compatibility with version 1.0 is not required.

1.2.2 File System Server Integration

The component should provide an option to place the uploaded files on the File System Server. An identifier should be provided for each uploaded file. The identifier is used to retrieve the file from File System Server.

The component can also instantiate the uploaded file from a given identifier and handle the downloading from the File System Server transparently. Component users do not need to interact with File System Server.

The solution should allow uploads with the same filename.

1.2.3 Uploaded File Serializable

The entity that represents the uploaded file should be serializable so that it can be stored as session data in applicable scenarios.

1.2.4 Configuration Manager Dependency

The previous version of the component depends on Configuration Manager 1.2. Version 2.0 should depend on Configuration Manager 2.1.4.

1.2.5 Documentation Update

Documentation of the previous version should be updated according to the latest standards.

1.3 Required Algorithms

No specific algorithms are required.

1.4 Example of the Software Usage

A distributed web application needs to register through several steps. User can upload their resume and the uploaded file will be sent to the File System Server. The representation of the file is stored in session until the complete registration form is committed. If the node serving that user fails the user's session will be migrated to a new node in the cluster. The uploaded file will still be available from the new node.

1.5 Future Component Direction

In the future different approaches could be developed to handle uploaded files with the same filename.

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.topcoder.servlet.request

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

- Upload target (file system, or file system server)
- Temporarily directory on the file system
- Connection information for the file system server

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

HTTP 1.1

3.2.2 TopCoder Software Component Dependencies:

- File System Server
- Configuration Manager

**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

3.3 Design Constraints

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

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.