[TopCoder]

Software Documentation : Java Generic File System Server

This page last changed on Oct 17, 2008 by axiom.

New Version 1.1 Changes

File System Server 1.1 adds support for the use of network file systems for the storage of files, and improves the client to allow direct uploading of data from an InputStream, avoiding the use of intermediate file storage.

Changes from the original 1.0 requirement specification are highlighted in **blue**.

File System Server Requirements Specification

Scope

1.1 Overview

The File System Server and Client component extends the respective IP Server and Client components to provide a mechanism for sending, storing, managing, and retrieving files and groups of files \from multiple clients to a server over a TCP/IP socket connection. The component shall never reject any file for any reason (including duplicate file names) other than hardware limitations. The component will handle any file-naming conflicts internally.

1.1.1 Version

1.1

Changes in version 1.1:

1. Addition of requirement 1.2.7 - Support for uploading from an InputStream
2. Addition of requirement 1.2.8 - Support for downloading to an InputStream
3. Addition of requirement 1.2.9 - Support for network file system persistence
4. Addition of note regarding use of Java 1.4
5. Addition of QA environments (Windows XP, RedHat Enterprise Linux 5.1)
6. Addition of the full list of component dependencies from version 1.0
7. Addition of documentation note about conversion of zuml to tcuml and clean-up as required.
8. Upgrade component dependencies to latest versions (Data Validation only).

Version 1.1 is backwardly compatible with version 1.0

1.2 Logic Requirements

1.2.1 File Submission

- Accept any file submitted from client, regardless of type, operating system, or size.
- Must refuse to accept file if accepting it would exceed hardware limitations and gracefully inform client of the situation. This must be done prior to the file being uploaded.
- Must appropriately handle, transparently to user, any operating system limitations or conflicts on file names.

1.2.2 Storage

- Store all files in the Server local file systems while avoiding name collisions.
- Unless client specifically requests overwrite of an existing file, the server shall ensure that duplicate file names do not overwrite each other.

- The Server must uniquely identify each uploaded file with a unique ID to allow duplicate file names, and provide that unique ID to the sending client. The server may rename the uploaded file as part of the unique identification process.

1.2.3 Acknowledgment

- Clients must be sent an upload-complete acknowledgment.

1.2.4 File Management

- Client must be able to upload, rename, delete, and retrieve any file, using the server based unique identification mechanism.

1.2.5 Searching

- Client must be able to search for files and/or groups (see 1.2.6) using simple masks and/or regular expressions based on the file and/or group name; or search for a specific ID number. The search will return a list of files and/or groups that meet the search criteria.
- The client may search for files within a specified Group.

1.2.6 Groups

- Component must provide ability to group files together and to manage such groups (create, update, retrieve, delete). Grouping files is the ability to create a Group Name and assign/remove files to that group.
- The group may be dissolved without affecting the member files, or it may be deleted with all of its member files.
- Files may be individually added or removed from the Group at any time. Retrieving a group will retrieve all files associated with that group. Group names must be unique.
- A file may belong to multiple groups simultaneously; however, the file would still be unique. Deleting the file (either individually or through a group deletion) would remove the file from all groups it belongs to.

1.2.7 Upload from InputStream

The updated 1.1 component will enhance the existing FileSystemClient or provide a new FileSystemClient that accepts InputStream objects instead of physical file locations. Naming of the files must be optional but still supported. Intermediate local file system storage is not permitted.

1.2.8 Retrieve to InputStream

The updated 1.1 component will enhance the existing FileSystemClient or provide a new FileSystemClient that returns an InputStream from which the caller may obtain file data. Intermediate local file system storage is not permitted.

1.2.9 Network file system persistence

The updated 1.1 component will enhance the server to support writing to a network file system. The network file systems supported must at least be the version of NFS in RedHat Enterprise Linux 5.1. There may be multiple File System Server instances running on independent physical servers accessing this network file system. However, with the possible exception of backup software, there will be no other clients directly modifying the files being maintained by this component.

1.3 Required Algorithms

None

1.4 Example of the Software Usage

- The component may be used in a web-based forum where users may upload multiple files (for instance, graphic files) with the same file name. The system must be able to uniquely identify each upload in order to ensure the correct retrieval of a particular file.
- An uploaded file may subsequently be reference in a page link for retrieval. An example would be a user posting to a forum and including a graphic file as part of the post. The file would be stored on the server (possibly rename or with the unique ID embedded in the name) and the user's post would include a hyperlink back to the uploaded file.
- Uploaded files may then be assigned to groups, based on content, or source, through the client interface.
- The component may be used on a server to permit centralized high volume and high speed file manipulation services.
- In the case of a web based forum, the client component would be embedded in the forum page while the server component would be a back end to the forum.
- In a client-server application, the client component would reside on the client system and the server component would reside on the server.
- A server may have connections from clients located / embedded in a variety of web-based systems or application systems.

1.5 Future Component Direction

The components will be used in applications requiring very fast, high volume data and file transactions from clients to server and back; and as a replacement for the Java performance-constrained Java serialization mechanism.

Support for persistence to alternate networked file systems such as Windows network drives, Samba, or Xdrive.

2 Interface Requirements

2.1.1 Graphical User Interface Requirements

None

2.1.2 External Interfaces

None

2.1.3 Environment Requirements

Java 1.5 / Java 1.6 should be not be used since it may break source-level backward compatibility:

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

2.1.4 Package Structure

com.topcoder.file.transfer

3 Software Requirements

3.1 Administration Requirements

3.1.1What elements of the application need to be configurable?

Server location and connection information, configuration file location, file storage location.
Client configuration and connection information.

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

None

3.2.2 TopCoder Software Component Dependencies:

IP Server version 2.0

The full list of dependencies in version 1.0:
* Base Exception 2.0.0
* Configuration Manager 2.1.5
* Data Validation 1.0
* GUID Generator 1.0
* ID Generator 3.0
* IP Server 2.0.1
* Type Safe Enum 1.1.0
For this version of the component we will make the following version upgrades:
* Data Validation 1.1
The component will not utilize APIs specific to higher versions of the components unless such a change is necessary for the new version to function.

**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
  * RedHat Enterprise Linux 5.1
  * Windows XP

3.2.5 Client / Server communications methods

- Must not use RMI.
- Must support synchronous and asynchronous requests.
- The IP Server component provides a communication mechanism that should be extended and expanded for this component.

3.2.6 Backup

- The component needs to be designed with the knowledge that the server-stored user data files may be backed up by an external application and as such should not impede those potential backups in any way.

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 TC UML.

The designer must convert the Poseidon zuml to TC UML and clean up the converted TC UML as required.