Requirements Specification

1. Scope

1.1 Overview

This upgrade of the Authentication Factory is intended to make the component truly pluggable. Although the current design applies Factory pattern, it's heavily bound with HTTP authentication backend. The new design will make the component independent of underlying authentication mechanism.

1.2 Logic Requirements

1.2.1 Independent of Authentication Backend

The component will not be required to know its underlying backend to be functional. To which backend the component is connected will be configured through Configuration Manager, and specific authentication will be invoked based on login context at run time.

1.2.2 Support Multiple Means of Authentication

Though user name and password scheme is most used to authenticate a subject, it should not be assumed. Other means of authentication, e.g., fingerprint, could be used.

1.2.3 A Common Interface for Authentication Return Value

The returned object from authentication needs to have a common interface, which is truly backend independent.

1.2.4 Support Current Web/HTTP Based Authentication

Current Web/HTTP based authentication needs to be continually supported. But retrieving contents, cookies, source URL, and HTTP headers will be through a generic interface outlined in 1.2.3.

1.2.5 Separate Web/HTTP Authentication Plug-in From Component Archive

Backend authentication plug-ins needs to be put into different archives (.jar files), and supplied to client on needed base.

1.2.6 An Optional Caching Mechanism

In some case, backend services may be too heavy to invoke on every authentication. For such cases, a simple cache will be used. Enable the caching or not will be determined by Configuration Manager. For this release, a time-based cache will be implemented.

1.3 Required Algorithms

None.

1.4 Example of the Software Usage

An application may authenticate its users in different ways. With this component, the application will be able to check different login against different mechanisms. For example, using LDAP to authenticate an internal user and using HTTP to authenticate a web user.

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: Java1.3, Java1.4

2.1.4 Package Structure

com.topcoder.security.authenticationfactory

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

Supported authentication plug-ins.

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

None.

3.2.2 TopCoder Software Component Dependencies:

Base Exception
Simple Cache

**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. 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
- Test Plan

3.4.2 Help / User Documentation

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