Distributed Protocol Factory 2.0c Requirements Specification

1. Scope

1.1 Overview

This enhancement adds a level of security to the distributed protocol. Authentication will be added to the protocol to verify the messages are received from approved peers. It is safe to assume that each node will have access to the same authentication store. This eliminates the need to worry about federated identities.

1.2 Logic Requirements

1.2.1 Authentication

Extend the protocol to accept an authentication ticket and the message it is sending. Each node will verify the message comes from an authenticated source. The TopCoder authentication component will be used to accomplish this task. It is assumed that each node will have access to the same authentication server.

If the authentication fails the message is ignored and an authentication error is returned otherwise the message behaves as normally.

It is ok that the authentication components use the configuration manager.

1.2.2 State

Once two peers have been authenticated, it is safe to assume they will be authenticated indefinitely. Subsequent authentication checks from the same peer are not necessary.

1.2.3 Authentication Ticket

The authentication ticket will be pluggable. Potentially different forms of authentication will be required such as private keys or different types of tickets.

1.2.4 Authorization

The authorization component will be used to ensure only approved systems may use the protocol. Once the message has been authenticated, the protocol will verify the authorized “user” has access to the protocol.

1.3 Required Algorithms

None.

1.4 Example of the Software Usage

A configuration manager is using the protocol to sync itself between other machines. Some clients will not be allowed to update information. The protocol will verify the person sending the message came from within the group and can send the information.

1.5 Future Component Direction

None identified.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

None.

2.1.3 Environment Requirements

- Development language: C#
- Compile target: .NET 1.1

2.1.4 Namespace Structure

TopCoder.Network.Synchronization.Secure

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the component need to be configurable?

The protocol implementation should be configurable programmatically.

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

Authentication source and authorization source

3.2.2 TopCoder Software Component Dependencies:

None.

3.2.3 Third Party Component, Library, or Product Dependencies:

None.

3.2.4 QA Environment:

- 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

3.4.2 Help / User Documentation

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