Common Component Standards Proposal

Date	8.20.2015
Spec Title	Security Token Service
Author	Michael Jacobs
Status	New
Version	1.0

Executive Summary

This document contains a proposal to the Common Component Standards (CCS) Steering Committee to form a new project team whose goal it is to produce a specification for a *Security Token Service,* which defines a component interface capable of satisfying the client authentication requirements of an OpenEdge distributed computing architecture.

The following sections of the document describe in general terms what the component is, the benefits it provides to application developers, and what the team's deliverables would be.

Component Description

Every business application operates on the basic premise that some portion of its operations and/or data requires restricted access according to an individual's job requirements. The strength and integrity of the component doing the validation of a user credentials (i.e. authentication) and managing their login-session is foundational to the effectiveness of the authorization to sensitive data and the audit trail that tracks it.

The role of a *Security Token Service* (*STS*) component in modern OpenEdge client-server and/or n-tier distributed application architectures focused on being that highly secured and trusted component for all of the application's services.

While a *STS* component may exist as an embedded component within a client-server type architecture, but its security & integrity is diminished. The *STS* component provides the greatest value to the OpenEdge business application when it exists as a separate entity from the business application clients, servers, and daemon services.

The *STS* component is always physically and architecturally a separate entity from the business logic's authorization services who authorize access to data and

operations, regardless of the business application's architecture. The *STS* generates and provides a Client-Principal security-token that satisfies the business logic's authorization requirements without any need of further user account access. In this way the *STS* component may operate as an embedded or distributed component, without a significant change in how the business application uses it.

In some cases the lifecycle of a *STS* issued security token is one operation. In other cases the security-token's lifecycle spans many operations in a statefull client login-session. The *STS* component handles both types according to its configuration policy, where the policy applies controls for longevity, concurrency, and other attributes.

As a separate entity a *STS* may be required to operate in a high availability (HA) and scalable architecture in order to meet the needs of all the business application's distributed components.

The *STS* component interfaces defined by the CCS specification will expose a set of public and protected APIs that offer the services:

1. A Single point of user authentication services across a distributed application
2. Client authentication and login-session policy enforcement
3. The issuing, validating, and lifecycle management of distributable security-tokens
4. Exchange of security-tokens of one client's native form for an equivalent of another client's native form

The APIs exposing an *STS* component's functionality will be defined in such a way that they may be applied to a number of business application deployment architectures that range from direct client and server access up to and including that of the role of a Federated authenticator service.

Benefits and Use Cases

The security model supported by an *STS* component has a number of very important improvements over the older flawed method of embedding full authentication and authorization service into every distributed application component.

Simplifies the Security Model in the Business Application

By removing the direct-login services from the business application's logic, the *STS* APIs effectively isolates it from the complexities and requirements for operating a secure authentication process regardless of where the business application logic runs.

Extensibility to Incorporate Strong User Account Systems

By abstracting the authentication process and user account access from the business application logic, the STS component's APIs provides a single programming method to obtain a security-token regardless of where the user account data is stored, how it is accessed, and how it is applied to the authentication policy rules.

Enforces a Single Authentication/Login-Session Policy

Using the *STS* APIs promotes the practice of enforcing a single authentication, security token management, and login session policy adherence across the application distributed services.

Enhanced Authentication Security

Using the *STS* APIs enhances overall application security as an independent component because it, and its policy, configuration, and any account data can exist in a highly secured environment that would not be consistent with that normally found in an application's business server.

Does Not Expose Client Credentials Over the Network

As an independent service the *STS* APIs can be used by secure network connections between itself and its clients, which allows the client's business data and operations to operate faster over non-secure connections.

Single Point of Authentication Across the Business Application

A *STS* component, acting as an isolated distributed service, effectively becomes a single point where all aspects of user authentication and login-session share the same policy enforcement via a common set APIs.

Promotes Single Sign-On Across the Business Application

An *STS* based authentication architecture, with its APIs support for single point of security token management, eases the business application's migration to using SSO between its components. This type of architecture promotes a single direct-login to any one of the business application's services who can then SSO and use any other service without the requirement of needing the original credentials to effect secondary direct-logins.

Interoperability with Federated Systems

All business applications will eventually interact with other business applications either as a client or as a server. When an external business application acts as a

client and uses a native form of security-token incompatible with the business application, the *STS's* APIs can bridge the gap by converting the foreign client's security token from its native form into one the native form used by the business application. Likewike, when the business application acts as a client to an external service that uses a different form of native security token, the STS's APIs can bridge the gap by turning its native security token into a form native to the external service.

Related / Dependent Common Component Specifications

Any *Security Token Service* implementation is a highly configurable component who's very nature is one where all of its run-time policy data must be stored and managed in a secure location. Therefore, using a common configuration component capable of supporting secure policy data in different forms/location is indicated.

The *Security Token Service* implementation will also require logging services for debugging and tracking operations that are capable of being secured against the mining of user account and session data. Therefore, having a common logging component capable of supplying a higher level of data storage suitable for auditing purposes is indicated.

Project Team Requirements

The *STS* project team's formation will commence upon acceptance of this standards specification proposal by the CCS Steering Committee. The composition of the team will be supplied to the CCS Steering Committee within 30 calendar days after acceptance.

The *STS* project team will provide regular updates whereby the CCS Steering Committee may gauge their progress, offer guidance, or require additional information.

The team's goal would be to produce a draft *STS* component specification within a period of 90 days after the team's formation was complete. Upon initial acceptance by the CCS Steering Committee for meeting specification requirements, the specification would be published to the CCS community for comments for a period of 90 days. During the 90 day community comments period the team is responsible for answering all comments and requests for clarification. At the completion of the 90 day comments period a second version of the *STS* specification will be produced that incorporates those public comments accepted by the team. The second version of the *STS* specification will then be ready for a formal review by the CCS Steering Committee for acceptance, cancelation, postponement, or denial.

An essential part of the project team's delivery is that of an operating reference sample that may assist is the community's in gaining a better viewpoint of how a *STS* component may be used, rather than just reading a specification. This sample would also serve as a physical illustration of the originating team's initial vision of the standard.

Upon acceptance of the *STS* team will have one last task, and that is to publicly publish the standard, sample, and any other artifacts generated as part of the project.

The team would remain until one of those three outcomes is reached: acceptance, cancelation or postponement. If the CCS Steering Committee postpones the project the team will be disbanded and should the project be resumed at a later date a new team would be formed.