caGrid Web Single Sign On

Design Document

Version No: 1.0

Last Modified: 11/09/07

Author : Kunal Modi

Team : WebSSO

Client : National Cancer Institute - Center for Bioinformatics,

National Institutes of Health,

US Department of Health and Human Services

Document History

Document Location

The most current version of this document is located in CVS under c3pr/docs/designs.

Revision History

| **Version Number** | **Revision Date** | **Author** | **Summary of Changes** |
| --- | --- | --- | --- |
| 0.1 | 10/10/2007 | Kunal Modi | Initial Version |
| 1.0 | 11/09/2007 | Kunal Modi | Baselined |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Review

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Team/Role** | **Version** | **Date Reviewed** | **Reviewer Comments** |
| Scott Oster | caGrid Chief Arch | 0.1 | 10/26/2007 |  |
| Josh Phillips | caGrid Lead | 0.1 | 10/26/2007 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Related Documents

More information can be found in the following related CSM documents:

|  |
| --- |
| **Document Name** |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

Table of Contents

[1. Introduction 4](#_Toc183220553)

[1.1 Overview 4](#_Toc183220554)

[1.2 Problem Scenario 4](#_Toc183220555)

[1.3 Scope 4](#_Toc183220556)

[2. Requirement Analysis 5](#_Toc183220557)

[2.1 Summary of Requirements 5](#_Toc183220558)

[2.1.1 Business Requirements 5](#_Toc183220559)

[2.1.2 Technical Requirements 5](#_Toc183220560)

[2.2 Assumptions 6](#_Toc183220561)

[2.3 Dependencies 6](#_Toc183220562)

[2.4 Known Issues or Future Considerations 7](#_Toc183220563)

[2.5 Technical Environment 7](#_Toc183220564)

[3. Detail Design 8](#_Toc183220565)

[3.1 Out of Box SSO Component 8](#_Toc183220566)

[3.2 JA-SIG’s Central Authentication Service (CAS) 8](#_Toc183220567)

[3.3 Overview 10](#_Toc183220568)

[3.3.1 Component of WebSSO Framework 10](#_Toc183220569)

[3.3.2 Single Sign in work flow 12](#_Toc183220570)

[3.3.3 Inter Component Transport Security 13](#_Toc183220571)

[3.4 Interaction between SSO Server and GAARDS 14](#_Toc183220572)

[3.5 Interaction between SSO Server and Delegation Service 15](#_Toc183220573)

[3.6 Interaction between SSO Agent and SSO Server 16](#_Toc183220574)

[3.7 Interaction between SSO Agent and Delegation Service 17](#_Toc183220575)

[3.8 Interaction between SSO Agent and Target Web Application 18](#_Toc183220576)

[3.9 Single Sign Out design 19](#_Toc183220577)

[3.10 Web / Grid Session Sign out 20](#_Toc183220578)

[3.11 HTTPUnits/ JUnit Test Cases 21](#_Toc183220579)

[3.12 Test Case Scenarios 21](#_Toc183220580)

[4. Configuration/Deployment Considerations 22](#_Toc183220581)

[4.1 Property/Configuration Files 22](#_Toc183220582)

[4.2 Deployment Considerations 22](#_Toc183220583)

# Introduction

## Overview

The WebSSO framework is chartered to provide a comprehensive solution for a Single Sign On framework for participating web application using caGrid’s GAARDS Framework. WebSSO framework provides a mechanism for users to navigate between different applications without being challenged to provide credentials to login when they switch application. Also, this framework will help the applications establish a grid session for the signed in user. This way the user can access grid services without being challenged for credentials

## Problem Scenario

Current approaches to address Single Sign On in web applications using caGrid are not sufficient to meet security and policies requirements set by the caBIG Security Working Group. Additionally, with bundling of applications in the form of suites and distributing them collectively amongst the caBIG community, there is a requirement to provide a Single Sign On solution which these teams can leverage easily.

Hence the caGrid Infrastructure team proposes to develop a Single Sign On product which can be easily integrated into the CCTS Suite as well as by other application that wants to leverage the Single Sign On capabilities. This solution will leverage the underlying caGrid Security Infrastructure and use it to establish a user’s identity and maintain his session across applications and grid services.

## Scope

This document defines the detailed design for Web Single Sign On Project. It provides detailed design for the providing a Single Sign On solution using the caGrid’s GAARDS framework including the new Delegation Service. It also shows how target application can leverage the solution. However it doesn’t give any design details about the Delegation Service. A separate document will address that which will be prepared by caGrid Team. Also this design document currently only provides the details for the caGrid WebSSO v0.5 Release.

# Requirement Analysis

## Summary of Requirements

### Business Requirements

1. **Web based Single Sign On**

There is a requirement to allow a user to provide his credentials only once while logging into an application. However when the user navigates to another related application either by clicking a link or entering the URL in the browser then he should not be challenged to provide his credentials again. He should automatically be logged into that other application.

1. **Grid Single Sign On**

There is a requirement to allow a user to access a Grid service once he has established a single sign on session without providing his credentials again. Also when the user navigates to another web application under the single sign on umbrella, then his grid identity should be effectively delegated to that other application such as he should be able to access a grid service seamlessly as he did from the first application.

1. **Parameter Passing**

There is a requirement for the WebSSO framework to allow application pass user’s specific data elements between each other as part of the URL which links various applications. Currently CCTS applications uses only GET method to pass parameters.

1. **Single Sign Out**

There is a requirement to log the user out of all the applications which he has visited under the single sign on session whenever the user presses logout on any one of the application. Also at the same time his grid session should also be terminated. Due to time restriction the out of box capability provided by the selected product will be sufficient.

### Technical Requirements

1. **Single Identity Provider**

As the entire CCTS Suite is to be deployed at a single Institution the 0.5 release of WebSSO should support a framework where all the applications that are part of the WebSSO Trust Fabric use a Single Identity Provider (IdP) to validate the user’s credentials against it. Also all these applications will use only a single Dorian server to obtain the grid identity for the authenticated user.

1. **Look and Feel**

The Web UI component of the WebSSO project should be configurable to allow different applications to plug in their look and feel templates. CCTS has a commonly developed look and feel the WebSSO application should be able to adapt to this look and feel.

1. **Session Timeouts**

WebSSO Component should provide a configuration parameter to configure the SSO session time out for the user. Also this session timeout should be linked with the user’s grid session.

1. **Browser Support**

WebSSO project should support at least Internet Explorer v6.0 and above as the end user browser. However if possible the WebSSO project should also try to avoid browser specific coding.

## Assumptions

1. **Applications to use Grid Identity**

All the participating applications of the CCTS will use the Grid Identity to identify the authenticated user. This is would ensure a common identity across the entire suite and aid in the single sign on capabilities.

1. **Applications to use e-Authentication Level 2 credentials, if LOA 2 is required.**

All the participating applications of in the bundle will use e-Authentication Level 2 credentials if required for user authentication. This means that the user will provide a username/ password pair to authenticate themselves to the WebSSO framework.

1. **Single Identity Provider for the WebSSO v0.5 (CCTS Release)**

For the 0.5 Release of WebSSO project, this design assumes that all the applications in the Suite will use a single Identity Provider for user credentials.

1. **Interface Stability**

Since WebSSO Release v0.5 is an interim release just to support the immediate release of the CCTS v1.0 there is not guarantee for interface stability between it and the final WebSSO caGrid Release v1.0.

1. **User Attributes**

The WebSSO project assumes that the following user attribute obtained from the SAML Assertion are okay to be returned to a target web application.

* User’s First Name
* User’s Last Name
* User’s Grid Identity

## Dependencies

1. **Delegation Service**

There is an external dependency on delegation service. WebSSO uses Delegation Service to park user’s grid credentials temporarily for the target web applications to retrieve them later. This requires the delegation service to be completed before WebSSO project so that WebSSO project can use it during its construction phase.

1. **CCTS Look and Feel**

Since WebSSO Login page has to provide the same common look and feel as the CCTS Bundle, there is a dependency on the CCTS look and feel. This look and feel template should be made available to WebSSO team during the construction phase

## Known Issues or Future Considerations

1. **Sign Out across all application**

Current solution provides an out of box implementation for Single Sign Out capabilities provided by the product used. However as a long term solution a more elaborate Single Sign Out Capability is desired which signs out the user from all the applications as well as destroy his grid session.

1. **Supporting Multiple Identity Provider**

Currently WebSSO solution is designed to support only Single Identity Provider which would be used for CCTS implementation. However in future there would be a need to support multiple identity providers. In this case there should a mechanism devised which would obtain the list of available Identity Providers and provide users an option to select the Identity Provider it wants to use to authenticate against.

## Technical Environment

* Client Interface
  + Internet Explorer 6.0 and above
  + Mozilla v1.5.0.3 and above (if possible)
* Application Server
  + Apache Tomcat 5.0 and above
* Operating system
  + Windows 2000, XP
  + Linux/Unix
* Java 1.5, Globus Toolkit 4.0.3
* SSO Framework
  + JA-SIG CAS Server 3.1
  + JA-SIG CAS Client for Java v3.0
* caGrid Release v1.1 + Delegation Service

# Detail Design

## Out of Box SSO Component

The WebSSO Team along with the caGrid Architecture team evaluated various open source technologies for selecting an out of box standard Web Single Sign On Product. Following are some of their findings

1. The Team looked at the following open source products in details –
   1. Internet 2’s Shibboleth
   2. JA-SIG CAS (Central Authentication Service)
   3. JOSSO (Java Open Single Sign On)
   4. Sun’s OpenSSO (Open Single Sign On)
2. The team decided to drop both Shibboleth and OpenSSO as they are too complex and overkill for our simple requirement of providing a simple SSO solution. Also the deployment of both Shibboleth and OpenSSO was elaborate needing various components to be installed and configured even if they weren’t used.
3. Both JOSSO and CAS are simple framework which met all out business needs. JOSSO uses web service as its underlying protocol for communication between the web agent and the center sign on server. However this caused an issue with the Axis version used by Globus. As a result of which there were compilation errors in the original JOSSO source code which would have needed elaborate recoding of the JOSSO product itself. Due to this technical difficulty JOSSO was dropped
4. Finally the team evaluated JA-SIG’s CAS. It seems to be a perfect fit for our requirement. Following are some of the important features which helped us finalized on it:
   1. Provides a simple spring based pluggable approach to extend and customize CAS to add functionality
   2. Simple non intrusive filter based integration with the target webapps (other ways possible too)
   3. Extendable Client Server Protocol. Provides capability of returning signed SAML assertions back to client from the server, hence providing a higher level of security
   4. Extensive documentation, user base and a bigger development team.

## JA-SIG’s Central Authentication Service (CAS)

The caGrid’s WebSSO Project uses CAS as the core component for providing the Single Sign On Framework. CAS is an authentication system originally created by Yale University to provide a trusted way for an application to authenticate a user. CAS became a JA-SIG project in December 2004.

CAS will be enhanced to use the caGrid’s GAARDS framework in backend to authenticate the user. Also this framework will be enhanced to obtain’s user’s grid credentials on the client side and provide it to the Target Web Application.

More details on CAS can be obtained from their website at: <http://www.ja-sig.org/products/cas/>.

It provides details about its inner architecture, protocols and available features and how to use them.

For caGrid WebSSO Project we are going to the following components from the JA-SIG CAS.

CAS Server v3.1 Stable Release: This is the server component of the SSO which facilitates authentication of user’s credentials and establishment of the Single Sign On Session and granting the SSO Session Tickets.

JA-SIG’s CAS Client for Java v3.0: This is client side web agent that includes classes for ticket validation, proxy ticket acquisition, servlets and filters for implementing the client portion of the CAS protocol and Assertion Java object for representing the results of a validation attempt. This library is usable for implementing custom CAS functionality, and can be used stand-alone to CASify existing applications.

WebSSO leverages the CAS Client for Java in such a way that it doesn’t prevent other existing CAS clients from being used, since it doesn’t modify the underlying client server protocol or modifying the client code in any way.

## Overview

The following diagram gives the over view of the WebSSO Project. It showcases all the components that are involved in the WebSSO framework. It also shows how these components interact amongst themselves. These components and their interactions are explained below.

Figure Overview of WebSSO Framework

### Component of WebSSO Framework

Following are minimum set of components which will be involved in the WebSSO Framework. The diagram above depicts a single target application which is protected using WebSSO. If there are more than one application then there will corresponding numbers of CAS Agents deployed along with them.

1. **Client Browser**

The Client is the end user trying to access a Target Web Application using a browser. Hence the browser is generally referred to as the user agent or just the client in most of the cases. The client browser is also useful for persisting the Single Sign On session in form of a cookie there by allowing the SSO Server to identify the user and not challenge him for credentials when he is redirected from another application for validation.

1. **Target Web Application**

This is the central authentication server which controls the entire Single Sign On framework. This component is provided by CAS. It also provides a login page which will be configured to adapt the look and feel of the CCTS Suite. It will be enhanced to communication with caGrid’s GAARDS

1. **CAS Single Sign On Server**

This is the central authentication server which controls the entire Single Sign On framework. This component is provided by CAS. It also provides a login page which will be configured to adapt the look and feel of the CCTS Suite. It will be enhanced to communication with caGrid’s GAARDS framework to authenticate the user using his local credentials. In return this server will obtain his grid credentials and pass them to the Delegation service and publish a delegation policy as in who all can access the grid credentials. Once use is authenticated, it generates a service ticket which is transferred back to the client. It then uses this ticket to confirm that the SSO Session has been established and retrieve user’s attributes from the SSO Server.

1. **CAS Agent**

This is the client component of the CAS Single Sign On framework which will be deployed and configured as part of the target web application. It is a standard client which all application can just integrate through the web.xml It intercepts all the client’s request to check if the single sign on session is established or not. It not it routes the user’s request to the Central Sign On Server allowing the user to authenticate himself and establish the SSO Session. On return from the Central Server the client intercepts the user’s request and validates the user’s session against the server. On success it also interacts with the Delegation Service to obtain user’s Grid Credential. Then CAS Agent forwards the user request along with the user’s attributes and grid credentials to the target application.

1. **Authentication Service**

In order to achieve federation, WebSSO allows user to log in using his local credentials. The CAS Single Sign On Server will be configured to point to an Authentication Service which will be standing in front of user’s Identity Provider. This Authentication Service accepts the user credentials in form of Username/ Password and authenticates the user’s credentials against the configured Identity Provider. If successful, it retrieved the user attributes from the Identity Provider and formulates a SAML Assertion which is sent back to the Central Server.

1. **Dorian**

Once user’s is authenticated from the Authentication service, the Central Server forwards the SAML Assertion to Dorian. The Central Server is configured to point to a single Dorian Server which acts as Identity Federation Service for the v0.5 Release. It generates the grid credentials which is nothing but a short lived proxy certificate and returns the same to the Central Server.

1. **Delegation Service**

The Delegation service is being built by the caGrid team which will be used to store the User’s Grid Credential along with a delegation policy. This delegation policy determines which applications have access to user’s Grid Credentials. On successful obtaining the User’s Grid Credential, the SSO Server publishes a delegation policy along with the User grid credentials to the Delegation Service. On the client side the CAS Agent connect to the Delegation Service using Host Credentials and passed the user’s grid identity to retrieve User grid credentials. If it is permitted then the Delegation Service returns the user’s grid credentials back to the CAS Agent, which is then forwarded the Target Application for it to use to access grid services.

1. **Sync GTS**

Sync GTS will be installed at all the participating applications as well as the central CAS Server. This daemon will keep the application in sync with the Grid Trust Fabric and update the CRLs accordingly. Once the CAS Server obtains a user’s Grid Credentials it will validate it against the GTS to make sure that the certificate is still valid and has not be revoked.

### Single Sign in work flow

Following is the workflow of interactions between the components mentioned above to establish a single sign on session and allow a user to access a protected web application.

1. User initiates a call to the target application by typing the URL to the target application in the browser which is protected by the SSO Framework.This user request is intercepted by the CAS Agent which searches for an established session. Since it can’t find it, it redirects the user’s request to the CAS Server providing the target application’s URL as return point.
2. The CAS Server displays a login page to the user. The user provides his local credentials in form username/ password pair on this login page for authentication.
3. The CAS server provides these credentials to the Authentication Service for the purpose of authentication.
4. If the credentials are valid, the Authentication Service returns a signed SAML Assertion back to the CAS Server.
5. The CAS server now passes this signed SAML assertion to Dorian.
6. Dorian makes sure that the SAML is signed by a registered Authentication Service and return a short term Grid Credentials (Grid Proxy) for the user.
7. They CAS server validates the Grid Proxy obtained from Dorian against the GTS to make sure that it is still valid and the CA has not be revoked.
8. Once validated, the CAS server obtains the list of Host Identities from the configuration files and formulates a delegation policy from it. It then publishes this delegation policy passing the user’s Grid Credentials to the Delegation Service
9. Now it formulates a service ticket and attaches it as part of the URL as a GET parameter. It redirects the user back to the application using the return URL provided
10. The CAS Agent sitting on the application again intercepts the request and retrieves the Service Ticket It now validates this Ticket against the CAS Server to make sure that the User is authenticated recently.
11. CAS Server validates the ticket and returns an assertion object back to the client which contains the user’s details. The CAS Agent retrieves these information and attaches them to User Request/Session
12. Now the CAS Agent connects to the Delegation Service using the Host Credentials of the Target Web Application to retrieve the User’s Grid Credentials (Grid Proxy) by passing the user’s Grid Identity
13. The Delegation Server retrieves the Host Identity from the call and checks it against the policy published for that user. If the application has been given the delegation rights, it returns the Grid Proxy for that user back to the CAS Agent.
14. The CAS Agent attaches the Grid Proxy as an attribute to the Session and forwards the request to the target application
15. The Target Application can retrieve all user attributes from the Request/Session including the user’s Grid Credentials. It then can use this Grid Credential to access a grid service on user’s behalf.

### Inter Component Transport Security

All the interaction mentioned above between various components of the WebSSO Framework will be secured. Following are the details various security mechanisms used between these components

|  |  |
| --- | --- |
| Interaction between Component | Transport Security Mechanism |
| Client Browser to Target Application | Can be configure to use SSL |
| Client Browser to CAS SSO Server | Using SSL |
| CAS SSO Server to Authentication Service | One Way Authentication |
| CAS SSO Server to Dorian | One Way Authentication |
| CAS SSO Server to Delegation Service | Mutual Authentication using User’s Grid Proxy |
| CAS Agent to CAS SSO Server | Mutual Authentication using Host Credentials |
| CAS Agent to Delegation Service | Mutual Authentication using Host Credentials |
| Target Web Application to Grid Service | Mutual Authentication using User’s Grid Proxy |

## Interaction between SSO Server and GAARDS

The WebSSO project leverages GAARDS framework provided by the caGrid project for the purpose of authenticating a user and obtaining his Grid Credentials. It uses the following components of the GAARDS framework

1. **Authentication Service**

For CCTS, the entire suite is going to use only a single Identity Provider which will store the user credentials. In order to access and validate these credentials, an Authentication Service will be hosted in front of this Identity Provider. The URL for this Authentication Service will be provided as a deployment time configuration parameter to the CAS Server Component. There is no additional security requirement or configuration needed for the CAS Server to connect and talk to the Authentication Service. Since there is only a single authentication point, there is no need to have a WAFY (Where Are You From) framework in the CAS Server to allow user to select where he is from.

1. **Dorian**

For CCTS, there will be a single Dorian server deployed. This Dorain will act as an Identity Federation Service which will provide a Grid Credentials for the user which is authenticated using the Authentication Service. The URL for this Dorian Service will be provided as a deployment time parameter to the CAS Server Component. There is not additional security requirement or configuration needed for the CAS Server to connect and talk to the Dorian. However the Authentication Service has to be registered with Dorian for Dorian to honor the SAML Assertions signed by the Authentication Service.

1. **Sync GTS**

This daemon will be running on the CAS Server to sync it with the Grid Trust Fabric. Once in sync it will update the CAs and CRLs by querying the GTS. CAS Server uses these CRLs to validate that the Grid Credentials it obtained from the Dorain to make sure that the issuing CA is trusted and that the particular identity is not on the revoke list.

## Interaction between SSO Server and Delegation Service

The WebSSO Project will use the newly developed Delegation Service for the purpose of delegating a user’s credentials. This is done for the reason that we cannot transmit the user’s grid credentials over a wire back from the SSO Server to the SSO Agent/Target Web App as part of the request. As a result there is a need to store these credentials, which are obtained at the time of user login, temporarily and publishing a policy which would allow the Target Web Application to retrieve these credentials later on to use them to access the Grid. At the same time when a user’s logs out this delegation policy must be canceled barring any applications from obtaining the user’s grid credential as user’s has official signed out. Since there will be a single Delegation Service hosted for the grid, its URL will be provided as a deployment time parameter for the SSO Server

Following are the two interactions between the SSO Server and the Delegation Service. The SSO Server is initiator of both of these interactions.

1. **Publish Delegation Policy At the time of User’s Login**

Once a user has been successfully authenticated and his Grid Credentials obtained it will be published on the Delegation Service using the following

* 1. User’s Grid Credentials
  2. List of Host Identities to which the credentials are to be delegated
  3. SSO Server will add itself to the list of Host Identities so that it can obtain the Grid Credentials at the time of logging the user out.
  4. Other delegation attributes like delegation path, delegation length etc.

The SSO server uses the User’s Grid Credential to connect to the Delegation Service and publish the policy for him.

1. **Destroy Delegation Policy At the time of User’s Logout**

At the time when user’s selects a logout on any of the application it will result in destruction of the user’s Single Sign On Session. This case the SSO Server will issue a call to the Delegation Service. Since the SSO Server won’t have a handle to User’s Grid Credentials, it would have to obtain them from the Delegation Service first. In order to facilitate this, the user credentials will always have to be delegated to the SSO Server. Once it has handle to the User’s Grid Credentials, it then goes ahead and issues a destroy call asking the Delegation Service to destroy the publish delegation policy for that user.

## Interaction between SSO Agent and SSO Server

WebSSO Project leverages CAS client server protocol to transfer user’s data from server back to the client. Once the user has been authenticated successfully by the server, the request is routed back to the target application with a authentication ticket embedded in the URL. The SSO Agent again intercepts the request and realizes that the user has been successfully authenticated by the CAS Server from the ticket which is attached by the server to the request. It now uses this ticket to query the CAS Server in order to retrieve the user attribute from the server. The server recognizes the ticket and pushes an assertion object which contains the user details back to the SSO Agent. This assertion object contains user details like his first name, last name, email id etc. It also contains the End Point Reference (EPR) to the Delegation Service.

The SSO Agent parses this assertion object to retrieve the user details. It provides a default implementation for parsing the assertion object via spring, allowing developers to provide their own parsers if they have their own custom assertions coming from the server.

## Interaction between SSO Agent and Delegation Service

Once the policy has been published by the server, the Target Web Application can retrieve the same by connecting to the Delegation Service using its host credentials. However in order to facilitate this, the SSO Agent deployed at the target web application provides a pluggable additional filter which can be used to connect to the Delegation Service on behalf of the Target Application and retrieve the delegated credentials. This filter is configured through the web.xml and would be chained after the CAS Client. This filter reads the User’s Grid Identity from the assertion object as well as the EPR to the published policy in the Delegation Server. Now using the host credentials of the target application for security, this filter queries the Delegation Service using the EPR retrieved. It passes the user’s Grid Identity to the Delegation Service and retrieves the stored Grid Credential for the user. This Grid Credential retrieved is attached to the session and passed on for the target application to use.

This way the Target Web Application is total agnostic of any of the caGrid GAARDS Infrastructure components. Also since the SSO Agent has to connect to the Delegation Service on behalf of the Target Web Application it would need access to its Host Credentials.

## Interaction between SSO Agent and Target Web Application

The SSO Agent (CAS Client + Delegation Lookup Filter) which gets deployed at a Target Web Application is actually a Servlet filter. This filter has to be configured using the web.xml file. Once configure, this agent filters all the HTTP Requests to make sure that the user’s is authenticated and a SSO Session has been established for that user. If not it handles the entire routing of the user to the SSO Server to establish the SSO Session. It can also be configured additionally to act on behalf of the Target Web Application to retrieve the user’s delegated credentials. All this information is passed from the SSO Agent to the Target Web Application using the Session attributes. It would read the assertion object which is retrieved from the SSO Server at the time of initial login.

Following is the List of User information which will be provided from SSO Agent to the Target Web Application along with the Attribute name to be used for them.

|  |  |
| --- | --- |
| **Session Attribute Name** | **Information** |
| CAGRID\_SSO\_GRID\_IDENTITY | User’s Grid Identity |
| CAGRID\_SSO\_FIRST\_NAME | User’s First Name |
| CAGRID\_SSO\_LAST\_NAME | User’s Last Name |
| CAGRID\_SSO\_EMAIL\_ID | User’s Email Id |
| CAGRID\_SSO\_DELEGATION\_SERVICE\_EPR | Delegation Service’s End Point Reference |
| CAGRID\_SSO\_GRID\_CREDENTIAL | User’s Grid Credential |

.

The Target Web Application can retrieve these attributes from the Session anytime it needs to.

**Note:** User’s Grid Identity and not his Local Identity are provided back to the Target Web Application.

## Single Sign Out design

For the v0.5 Release, WebSSO team plans to provide whatever Sign Out Capabilities are provided by CAS. However these will be enhanced to extend the web sign out over grid too. The SSO Server component which handles the logout function of destroying the SSO Session will be extended to now call the Delegation Service to issue a destroy policy request as mentioned above.

Following is the detailed description of the sign out functionality to be provided as part of v0.5 Release

* A user comes to Application A through web browser. Application A challenges the user to provide credentials.
* Once the credentials are validated a single sign on session should be established using the underlying Grid Framework
* Now user clicks a link to Application B from Application A or types the URL for Application B in the same browser session
* Application B should be able to acknowledge the Single Sign On session and not challenge the User to provide credentials again. It should automatically log in the user
* Now user click log out on Application B. He would be logged out of Application B locally. Also his Global Single Sign On Session would be destroyed. This is done by routing the user back to the CAS Server and destroying the login session.
* During destruction of the login session, the CAS Server would connect to the Delegation Service to obtain the User’s Grid Credential. Now using this credential it would issue a Destroy Delegation Policy to the Delegation Service. This effectively end user’s Grid Session as a new application cannot obtain the user’s grid credentials anymore,
* However the user would still remain logged in locally on Application A.
* Now user clicks a link to Application C from Application A (since he is still logged in there) or types the URL for Application C in the same browser session
* Application C will challenge the User to provide credentials again as it cannot find a Single Sign On session. A new Single Sign On as well as Grid Session is established for the user.
* Alternatively Application A and B which already have handle to user’s Grid Credentials can continue to use them to access the grid service until they time out (Grid Proxy Timeout)

## Web / Grid Session Sign out

WebSSO will have a configurable SSO Session time out which will be provided as a configuration entry at the time of SSO Server Deployment. Individual Application will control their session outs. However the general rule is that these sessions cannot extend beyond the Grid Proxy time out.

If the individual application’s session times out, the SSO Agent can’t find the session object and hence can’t retrieve the stored Assertion object. It assumes that the User’s is not authenticated and forwards the user to the SSO Server. The SSO Server checks to see if the user’s still has a valid SSO Session if so it generates a Service Ticket again and redirects the user back to the Target Web Application. The SSO Agent validates the ticket and reestablishes the local session. This way the user is logged back into the application without providing his credentials again.

However if the user’s SSO Session has expired then he would be have to provide his credentials again on the login screen to reestablish is SSO Session.

The SSO Agent can also check if the User’s Grid Credentials (Grid Proxy) have timed out or not on each request. If the proxy has timed out then SSO Agent can issue a log out (or renew call) to the SSO Server forcing the user to re-login. This way the user obtains a new Grid Proxy which then can be used to access Grid.Unit Testing

## HTTPUnits/ JUnit Test Cases

This solution requires integration with an end user application. Also the testing of these is dependant on many components to be deployed up and running. As a result it is not possible to formulate automated HTTPUnits/ JUnits to test these features.

Hence the team is planning to perform integration tests.

## Test Case Scenarios

The test case scenarios will be developed in conjunction with the QA Team. Based on the initial design the overall test scenarios are as mentioned below. Note that based on details each of these scenarios can have multiple test cases

* 1. Testing Web Single Sign On
     1. Signing into One application and then in the same browser session accessing another application without being challenged for credentials
     2. Signing into One application and then clicking a link on that application which forwards your to another application accessing another application without being challenged for credentials
  2. Testing calling a Grid Service once the Single Sign On Session is established
  3. Testing Parameter Passing when a hot link is clicked to another application
  4. Testing the Sign Out feature
  5. Testing the Web Session / Grid Session Time Out
  6. Testing the common look and feel
  7. Testing for browser compatibility

# Configuration/Deployment Considerations

## Property/Configuration Files

1. WebSSO Sign Server would need the following configuration which needs to be provided at the time of deployment
2. URL for the Authentication Service which fronts the single IdP which will be used by the CCTS Suite
3. URL of the Dorian Server which would provide the Grid Credentials
4. URL of the Delegation Service which would be used to publish and store the delegation policy
5. List of Host Identities of all the applications in the CCTS Suite to which the user’s credential will be delegated
6. Single Sign On Session time out entry
7. WebSSO Client Agent would need the following configuration which needs to be provided at the time of deployment
8. URL of the Delegation Service from where the credentials are needed to be obtained.

## Deployment Considerations

1. The WebSSO Sign On Server needs to be deployed with Host credentials so as to be able to post to the Delegation Service.
2. The WebSSO Client Agent will be deployed along with the target application in its container. It would use the target application’s Host Credentials to access the Delegation Service.
3. The WebSSO Sign On Server and the WebSSO Client Agent + Target Application will have to install and run Sync GTS to establish the Grid trust fabric.