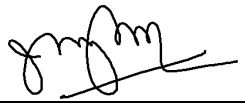
Document Information:

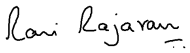
Sponsor/Owner	**NCI**
Protocol/Project	**CTSU**
Function/System	**CTSU Single Sign-On (Java) Software Framework**
Document	**CTSU SSO (Java) Design V1.0**

Approvals:



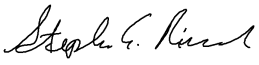
2011-10-17 11:21 AM

IT Manager / Jayan Nair — Date



Ravi Rajaram
I am approving this document- Signing for Manisha Rakheja
2011-10-17 08:11 PM

Assistant Project Director / Ravi Rajaram — Date



Stephen E. Riordan
2011-10-18 08:58 AM

Project Director / Steve Riordan — Date

Table of Contents

1. References

#	Name	Location (Intranet)	Location (Internet)
1.	CTSU SSO Java Starter Kit Installation and Integration Document (CTSU_SSO_Java_StarterKit_Installation_V1.0.docx)	\\westat.com\dfs\CTSU8339\Tasks\8339_14_CDMS\07_IT\Releases\CTSU_SSO\Java\1.0	https://www.ctsu.org/ctsusso |---JAVA |---VER_1.0 |---Documents
2.	CTSU Rave Requirements Document (CTSU-RAVE_Integraion_Requirements.docx)	\\westat.com\dfs\CTSU8339\Tasks\8339_14_CDMS\07_IT\Requirement	

2. Definitions, Acronyms, and Abbreviations

#	Abbreviation	Expansion	Description
1.	API	Application Programming Interface	A set of declarations of the functions (or procedures) that an operating system, library or service provides to support requests made by computer programs.
2.	CTEP-IAM	Identity and Access Management	Unique identity management system provided by CTEP
3.	IdP	Identity Provider	It is a centralized system that creates, maintains, and manages identity information for principals (users) and provides primary authentication to other service providers within a federation.
4.	SAML	Security Assertion Markup Language	XML-based open standard for exchanging authentication and authorization data between security domains, that is, between an identity provider (a producer of assertions) and a service provider (a consumer of assertions).
5.	HTTP	Hypertext Transfer Protocol	A communications protocol for the transfer of information on the internet
6.	HTTPS	Hypertext Transfer Protocol over Secure Socket Layer	An URI scheme used to indicate a secure HTTP connection
7.	SOAP	Simple Object Access Protocol	A protocol for exchanging XML-based messages over computer networks, normally using HTTP/HTTPS
8.	SP	Service Provider	A system that provides services (such as patient enrollment service or clinical data collection service) to the end users.
9.	SSO	Single Sign-On	Ability to login in once and access multiple systems if connected within the federation

3. Introduction

As part of the integration of Rave as the CDMS (Clinical Data Management System) for the Cooperative Groups and other lead organizations, a CTEP-IAM based Single Sign-On (SSO) authentication system is planned to be used for authenticating users by the existing web based systems at the Cooperative Groups. The CTSU SSO authenticator implements Security Assertion Markup Language (SAML) for login authentication.

To enable the incorporation of this SSO federated authentication system to existing Cooperative Group applications, CTSU created a SSO software framework and a starter kit to integrate CTEP IAM SSO authenticator to existing JAVA and .NET applications of the cooperative groups.

This document details the installation and usage instructions for the CTSU SSO Framework for Java. A similar document will be available for .NET developers for using the CTSU SSO Framework for .NET.

The CTSU SSO framework provides the following advantages:

1. Object oriented approach by providing a class based implementation to hide the SAML complexity.
2. Simple APIs to construct SAML request.
3. Simple APIs to extract SAML response.
4. Ability to verify the digital signature of the Identity Provider.
5. Optional roster data integration for authorization.
6. Ability to extend the framework to work with other Identity Providers other than NCI CTEP-IAM.
7. Two options for integrating SSO to the existing web applications
    a. iFrame Based Approach: In this approach the developers can embed the CTEP IAM login content to the existing login page. The displayed IAM login content can be customized by using a CSS file integrated with the implementation.
    b. Page Redirection Based Approach: In this approach when the users login to the existing login page, they will be redirected the URL of the Identity Provider (CTEP-IAM) and after they successfully login the user will be sent back to the welcome page of the original system they tried to access. Ability to replace the CTEP IAM image header with cooperative group's specific image header is also incorporated with this implementation.

4. Scope and Purpose

The CTSU SSO Framework and starter kit are not products by themselves. This document specifies a sample implementation and a recommended framework that the cooperative groups and other organizations can use to integrate to their existing web based systems to use the NCI CTEP SAML authentication scheme.

The requirements for this development are captured as part of the CTSU-Rave integration effort and are listed in the references section. The testing details are available in the JIRA issue tracking system.

This document specifies the design of the CTSU SSO API framework and their architectural details. It includes the most important details that will enable the developer to sufficiently understand the software architecture to enable its integration with their existing systems.

The UML modeling diagrams used in this design document are described below.

1. Use Case Diagram: A use case diagram is a type of behavioral diagram. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals—represented as use cases—and any dependencies between those use cases.

2. Class Diagram: A class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes.
3. Sequence Diagram: A sequence diagram is used primarily to show the interactions between objects in the sequential order that those interactions occur.
4. Activity Diagram: An activity diagram represents the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

4.1 Target Audience

This document is targeted at the stakeholder – management, architect, developer and 3rd party implementer – to aid in developing a high-level familiarity with the CTSU SSO API architecture as well as a more detailed understanding to support implementation. The SSO API framework starter kit provides a set of UI tools as well as documentation for the users which will complement this document.

5. Requirements

Please refer to the CTSU Rave Integration Requirements document listed in the references section for the full listing of the requirements gathered during the initial KA (Knowledge Acquisition) sessions.

6. Development Information

6.1 System Information

Software	Version
Java	JDK 1.5
MVC Framework	Spring MVC 2.5.6
View Framework	JSP, JSP tag libraries, AJAX, Scriptaculous, JMESA library
ORM	Hibernate 3.2.0
Transactions, Inversion of Control	Spring 2.5.6
Application Server	JBoss 5.1.0
Database	Not applicable
Development Tools	JDeveloper 10.1.3, Eclipse, Enterprise Architect
Build	JDeveloper / Ant
Source Control	VSS
JDBC Driver	Oracle JDBC 10.2.0.3.0 (ojdbc14.jar)
JRE	jdk1.6.0_25
caDSR API	4.0

7. SAML/SSO Overview

Security Assertion Markup Language (SAML) developed by the Security Services Technical Committee of the Organization for the Advancement of Structured Information Standards (OASIS) is an XML-based framework for communicating user authentication, entitlement, and attribute information. SAML is a flexible and extensible protocol. As its name suggests, SAML allows business entities to make assertions regarding the identity, attributes, and entitlements of a subject (an entity that is often a human user) to other entities, such as a partner company or another enterprise application. SAML protocol is designed to be used – and customized if necessary – by other standards. SAML has emerged as the gold standard

for federated identity. By defining standardized mechanisms for the communication of security and identity information between business partners, SAML makes federated identity, and the cross domain transactions that it enables a reality.

SAML can act to push responsibility for proper management of identities to the identity provider, which is more often compatible with its business model than that of a service provider. Using SAML to "reuse" a single act of authentication (such as logging in with a username and password) multiple times across multiple services can reduce the cost of maintaining account information by transferring the burden to a single centralized entity - the identity provider.

An *assertion* is a package of information that supplies one or more statements made by a SAML authority. SAML defines three different kinds of assertion statement that can be created by a SAML authority.

- **Authentication:** The specified subject was authenticated by a particular means at a particular time. This kind of statement is typically generated by a SAML authority called an identity provider, which is in charge of authenticating users and keeping track of other information about them.
- **Attribute:** The specified subject is associated with the supplied attributes.
- **Authorization Decision:** A request to allow the specified subject to access the specified resource has been granted or denied.

Web based SSO, is a single sign-on service where the user is authenticated.

7.1 Benefits

1. Centralized Storage of User Information
2. More secure than other custom SSO methods
3. Industry standard
4. Can be used in a federation for federated authentication
5. User & Password issues are managed centrally
6. Platform and Vendor Neutrality

8. Core Design

We made the class hierarchy as simple as possible. The CTSU SSO API framework just consists of four main classes.

1. SAMLRequest
2. SAMLResponse
3. SSOAuthenticator
4. SSOUser

All the classes are declared as abstract classes to provide member variables and methods that are wholly shared by all IdP specific subclasses which are extended from these parent classes.

8.1 Interface Class Diagram

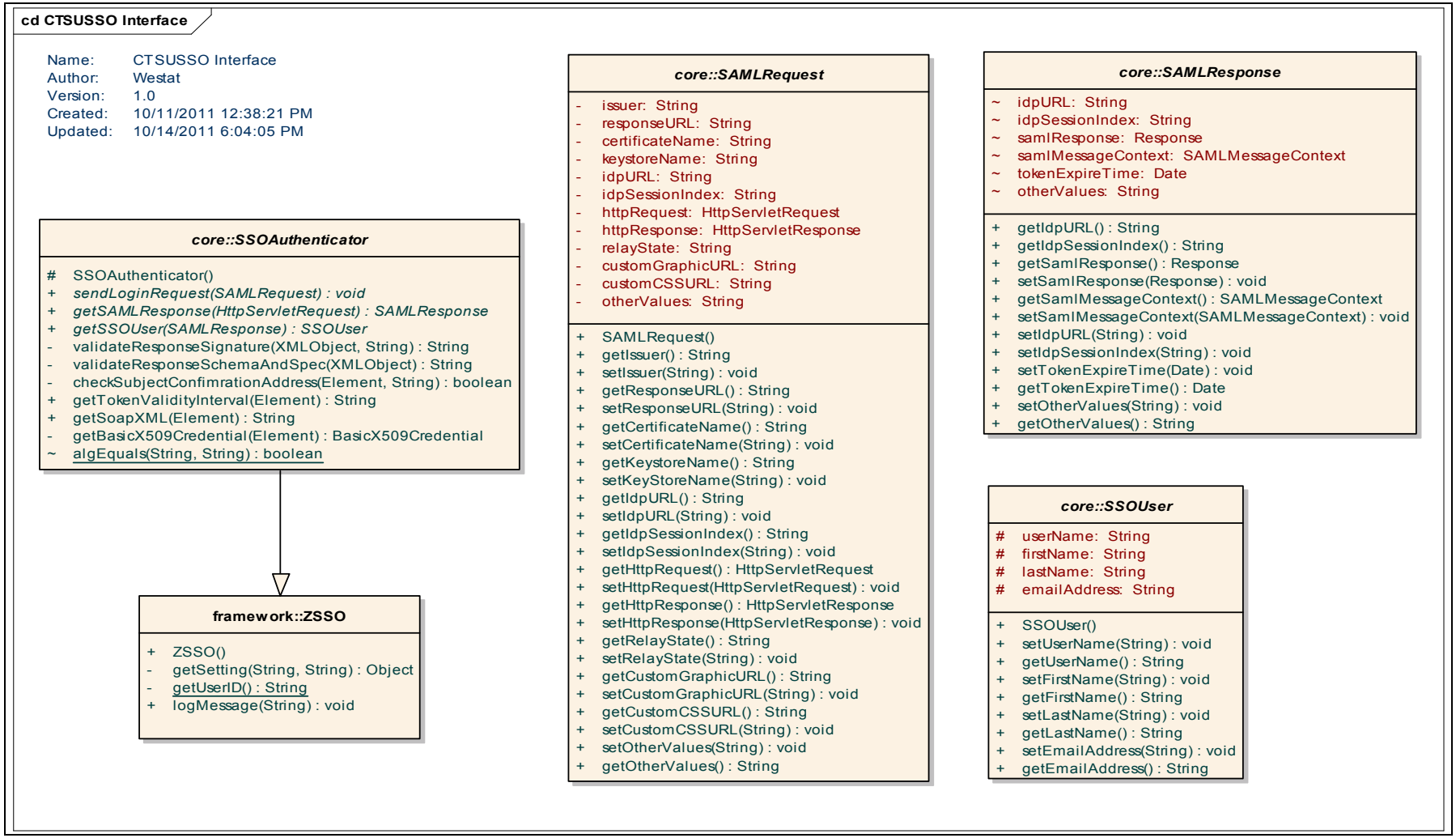


Figure 1: Interface Class Diagram

8.2 ZSSO

This is the parent framework class. As of now apart from the logging mechanism no additional features/methods have been added. But going forward once the framework matures we expect some additional method to be included.

8.3 SAMLRequest

The SAMLRequest is the class definition for the main request object which holds key information related to the SAML authentication request. It has all the attributes that are required by the IdP to process the user authentication. It also holds key information about the certificate key store location also.

8.3.1 Attributes

Attribute	Type	Description
issuer	String	The system which initiated the request. This should be the SP name
responseURL	String	The landing page upon successful authentication. This is the page where IdP will send back the SAML response. SP->IdP, this is the issuer URL and for SP1->SP2, this is the SP2 URL.
certificateName	String	Certificate use to verify the IdP signature
keystoreName	String	Where the certificate can be located
idpURL	String	Address of the IdP provider
idpSessionIndex	String	This holds the session index. Required for logout request and SP1 to SP2 request
httpRequest	HttpServletRequest	The basic request object
httpResponse	HttpServletResponse	The basic response object
relayState	String	This is for deep linking into RAVE
customGraphicURL	String	This is to specify to use the custom graphic/logo URL
customCSSURL	String	This is to specify to use the custom CSS for iFrame URL
needPersonRoster	boolean	This is to set to true if the person roster information is needed
otherValues	String	Future placeholder

8.3.2 Operations

Corresponding getters and setters are available for the above attributes.

8.4 SAMLResponse

The SAMLResponse is the main response object which holds key information from the IdP once the authentication has been process by the IdP. It has all the attributes that are required by the SP to process the authenticated user into their system. It also holds key assertion related to the authentication along with the IdP session index which is basically the confirmation key. This is the index key that is tied to the session expiry and also when jumping from one SP to the other within the federation.

8.4.1 Attributes

Attribute	Type	Description
idpURL	String	The address of the IdP provider that provided the authentication
idpSessionIndex	String	This holds the authenticated session index. Also needed for logout request and SP1 to SP2 request
samlResponse	Response	SAML 2 response object
samlMessageContext	SAMLMessageContext	SAML message context extracted from SAML response

Attribute	Type	Description
tokenExpireTime	Date	IdP token expiry timestamp
otherValues	String	Future placeholder

8.4.2 Operations

Corresponding getters and setters are available for the above attributes.

8.5 SSOAuthenticator

The SSOAuthenticator is the main class that sits between the SAMLRequest and SAMLResponse classes. It's main job is to package the objects to/from the IdP.

8.5.1 Attributes

No private attributes defined.

8.5.2 Operations

Method	Input	Return	Description
sendLoginRequest (abstract)	SAMLRequest	void	Main function of the method is to construct the SAML request based on the inputs and package it to the IdP for authentication
getSAMLResponse (abstract)	HttpServletRequest	SAMLResponse	Gets the SAML response object for the request input. Will contain all the attributes returned by the IdP
getSSOUser (abstract)	SAMLResponse	SSOUser	Method to extract the authenticated user information as sent in by the IdP
getTokenValidityInterval	Element	String	Will return the token expiry as a string from the SAML DOM.
getSoapXML	Element	String	For future use

8.6 SSOUser

As the class name suggests, the SSOUser class holds all the user information from the IdP upon successful authentication.

8.6.1 Attributes

Attribute	Type	Description
userName	String	The user name of the authenticated user
firstName	String	The first name of the authenticated user
lastName	String	The last name of the authenticated user
emailAddress	String	The email address of the authenticated user

8.6.1 Operations

Corresponding getters and setters are implemented for the above attributes.

9. Implementation Design

This design talks about a single IdP implementation. In this case, the single IdP would be the CTEP-IAM IdP. Based out of the core APIs defined, this implementation adds the business functionality to the abstract classes.

The implementation design consists of the following CTEP subclasses.

1. CTEPSAMLRequest
2. CTEPSAMLResponse
3. CTEPSSOAuthenticator
4. CTEPSingleSignOn
5. CTEPSSOUser
6. PersonRoster

9.1 Interface Class Diagram

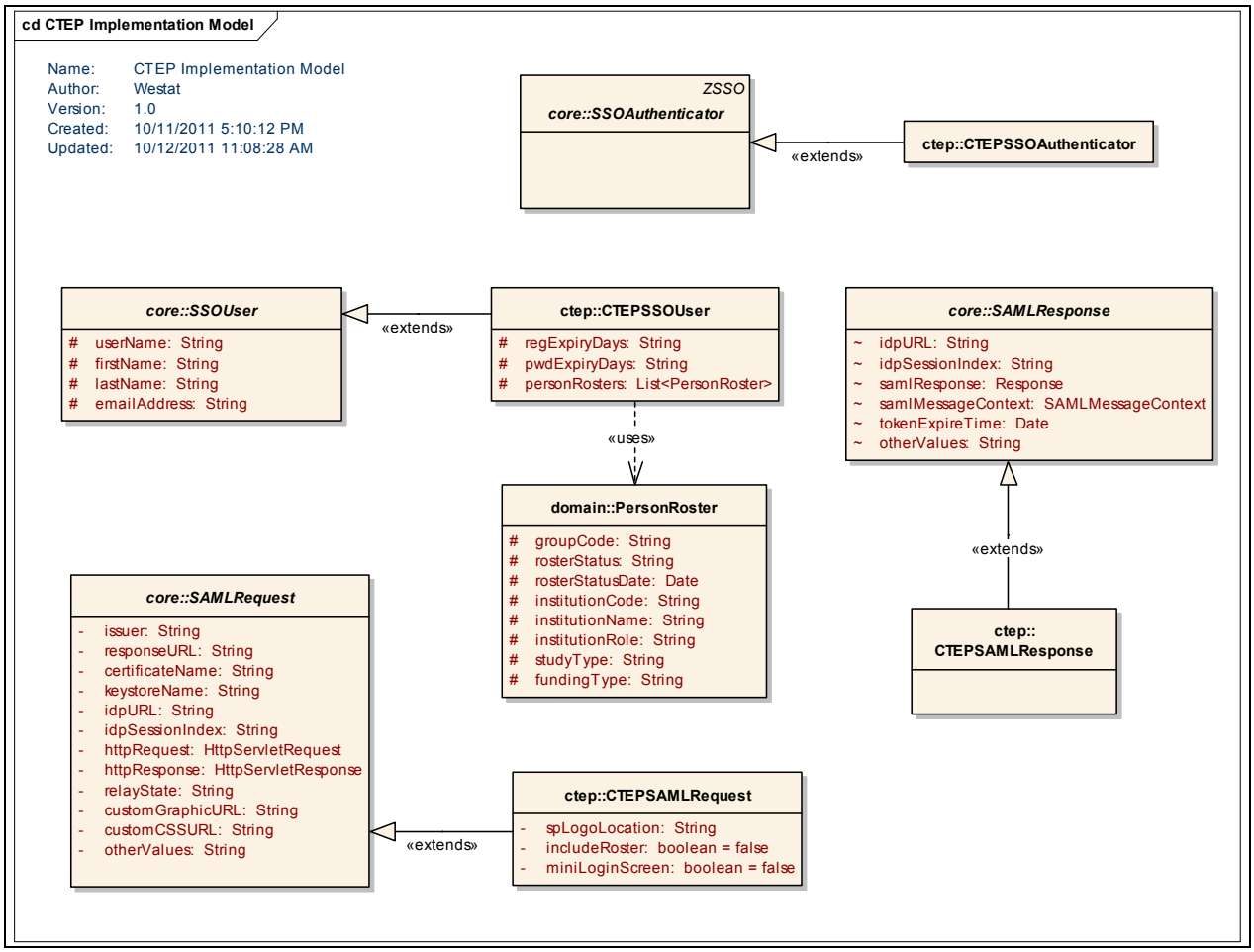


Figure 2: CTEP-IAM Interface Class Diagram

9.2 CTEPSAMLRequest

This subclass extends the SAMLRequest abstract class. It has the following additional attributes specific to CTEP-IAM IdP.

9.2.1 Attributes

Attribute	Type	Description
spLogoLocation	String	URL of the public SP logo

Attribute	Type	Description
includeRoster	boolean	Whether to include roster information from CEWS
miniLoginScreen	boolean	Whether to get a smaller login interface for embedding in an iFrame

9.2.2 Operations

Corresponding getters and setters are available for the above attributes.

9.3 CTEPSAMLResponse

This subclass extends the parent SAMLResponse class. To start with, it is an empty class. This class is included to show how Groups can extend the current implementation for using with a different IdP.

9.4 CTEPSingleSignOn

This is the main CTEP-IAM web SSO processing class. It extends the parent framework class to inherit all the core functionality like logging. All the logic for processing the authentication resides in this class.

9.5 CTEPSSOAuthenticator

This subclass extends the SSOAuthenticator. It has the implementation to all the abstract methods of the parent class. The implementation instantiates the CTEPSingleSignOn class to set and get the SAML request/response.

9.6 PersonRoster

The CTEP-IAM has the functionality to invoke the CEWS (CTSU Enterprise Web Service) to obtain the roster information for the user. All the attributes exposed by CEWS are available here.

For more information please refer to the CEWS guide.

9.6.1 Attributes

Attribute	Type	Description
groupCode	String	The Group's CTEP code
rosterStatus	String	The roster's status
rosterStatusDate	Date	The roster status date
institutionCode	String	The CTEP institution code
institutionName	String	The institution name
institutionRole	String	The role of the institution
studyType	String	The type of study
fundingType	String	The funding type

9.6.2 Operations

Corresponding getters and setters are available for the above attributes.

9.7 CTEPSSOUser

This subclass extends the SSOUser abstract class. It has the following additional attributes specific to CTEP-IAM IdP.

9.7.1 Attributes

Attribute	Type	Description
regExpiryDays	String	The no. of days before the CTEP registration expires
pwdExpiryDays	String	The no. of days before the CTEP password expires

Attribute	Type	Description
personRosters	List<PersonRoster>	The list of Person Rosters

9.7.2 Operations

Corresponding getters and setters are available for the above attributes.

10. CTSUSSO.jar/dll

The ctsusso.jar for JAVA and ctsusso.dll for .NET comes pre-configured to work with CTEP-IAM as the Identity Provider.

As part of the starter kit distribution, the following packages are included with the ctsusso.jar/dll

- com.westat.ctsu.sso.framework – this is a framework suggested by the CTSU on how to design the SAML web SSO. This framework will provide a clean separation between the web input and the business logic.
- com.westat.ctsu.sso.core – this package contains all the core SAML parent class with abstract methods/attributes for implementation by the subclasses.
- com.westat.ctsu.sso.ctep – this package contains all the CTEP-IAM implementation subclasses that extend the parent core classes.
- com.westat.ctsu.sso.domain – this package contains all domain objects pertaining to the IdP.

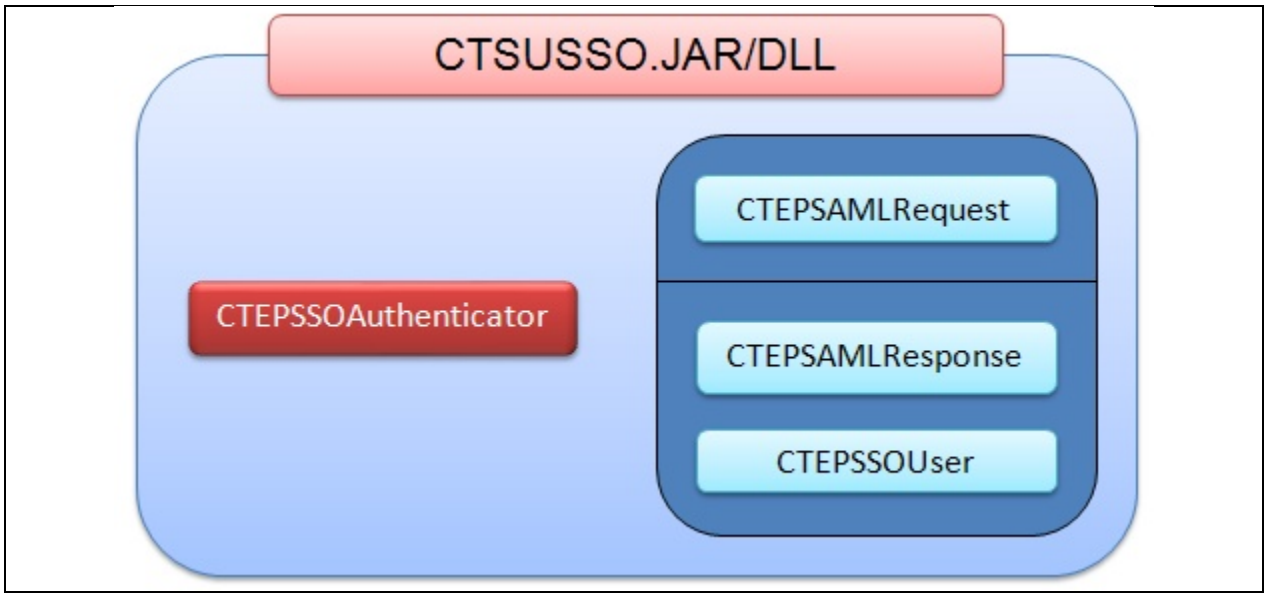


Figure 3: CTEP-IAM CTSUSSO jar/dll

11. SAML Authentication Workflow

Here's a high level overview of the how the whole SAML workflow takes place.

1. Any SP that will trust a 3rd party IDP Authority will need a copy of its certificate
2. A client connects to an Authority and authenticated with local security information (username/password) and supplies a certificate (optional) that it will use to prove its identity.

3. The Authority authenticates the client and verifies the supplied certificate against Certificate Authority's (CA) certificate (if available).
4. Authority creates assertion with client roles/attributes, identity, supplied certificate, signed assertion and returns to client.
5. The SOAP message contains the SAML assertion in the header plus a signature of the SOAP message. Signature is created with private key corresponding to certificate.
6. The client verifies the message signature, validates its SAML compliance, validates the subject confirmation data and it also checks the SAML signature against the Authorities certificate that is has locally.

11.1 Authentication Sequence Diagram

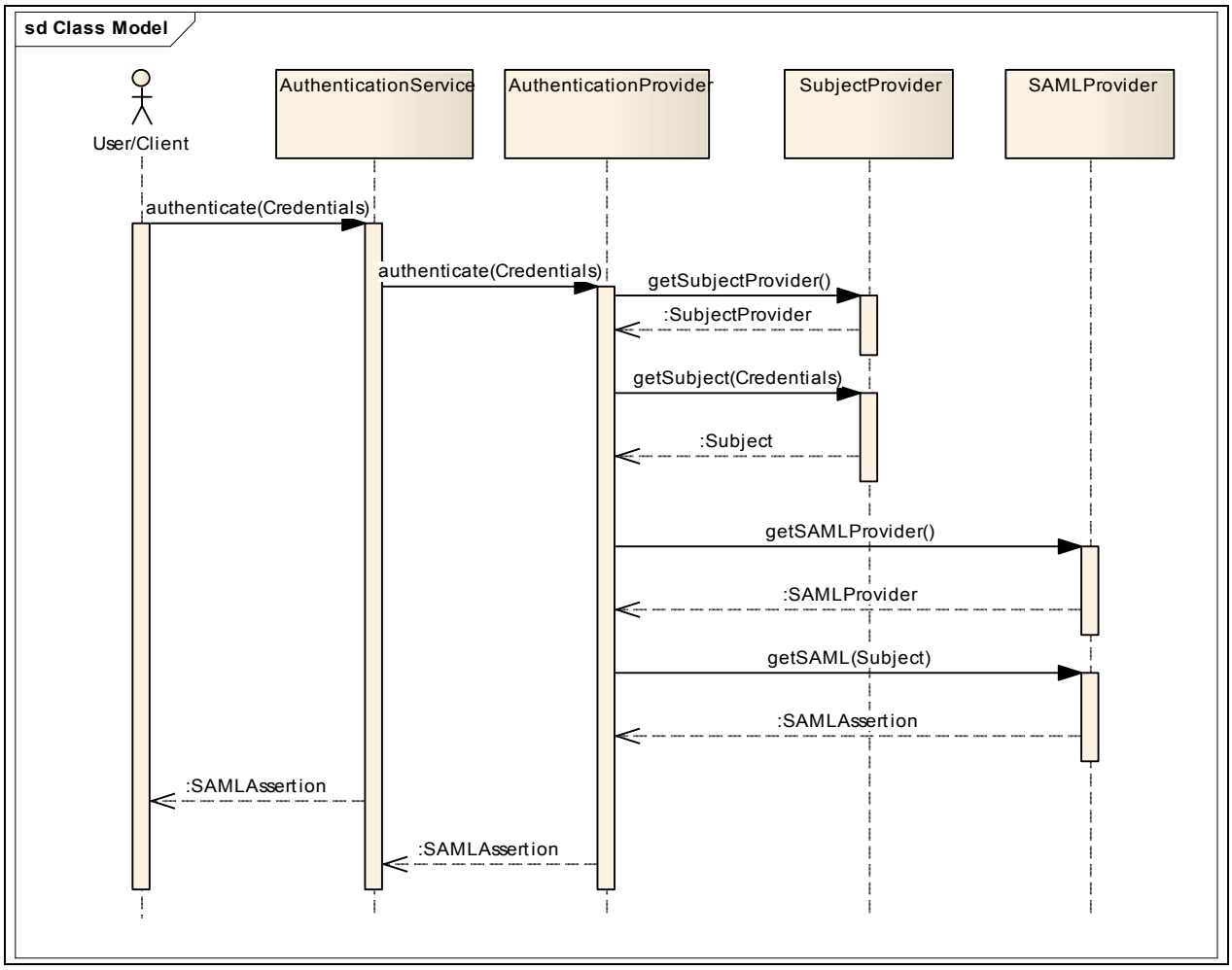


Figure 4: Authentication Sequence Diagram

11.2 Authentication Workflow

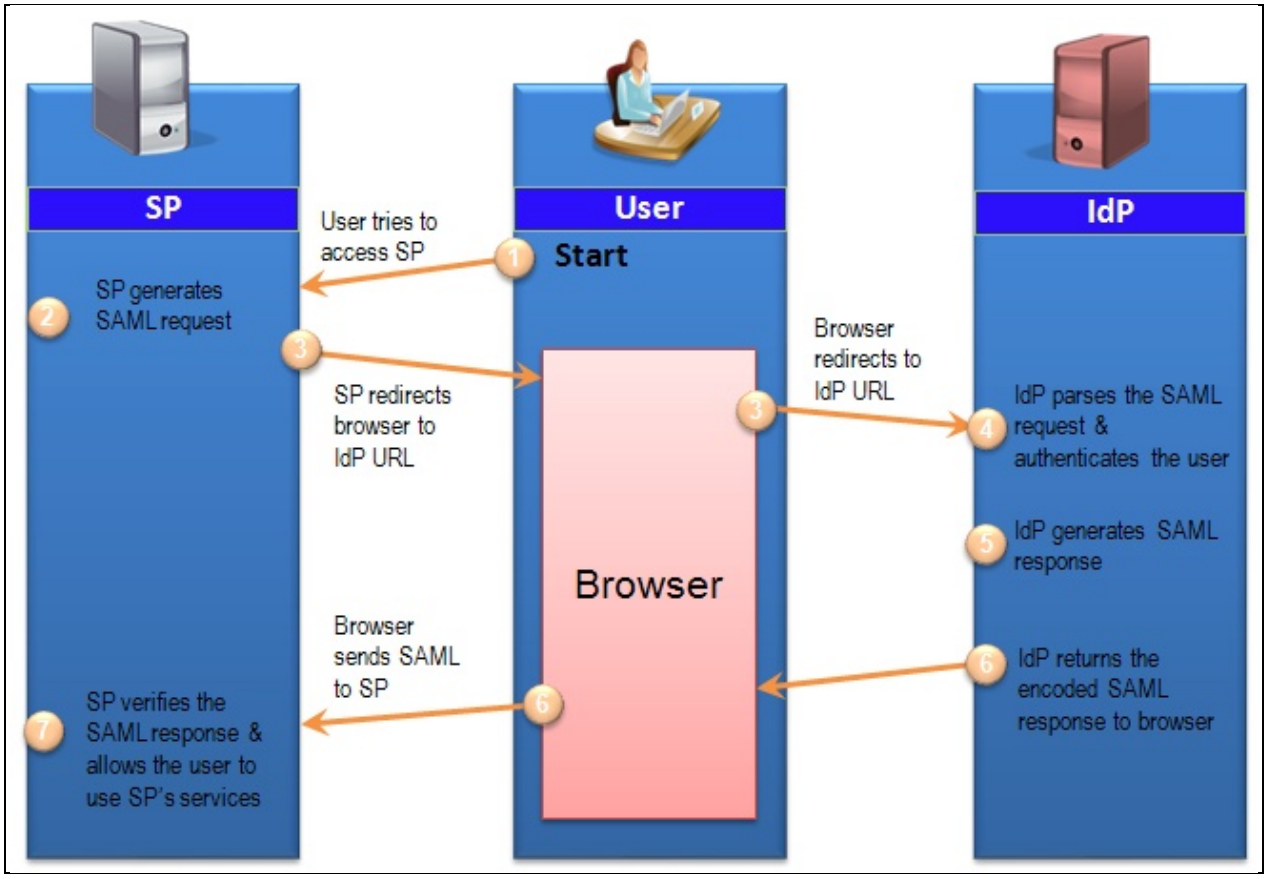


Figure 5: SAML SSO Workflow