**Faculty of Engineering of University of Porto**


# Implementation of a Single Sign On solution using Security Assertion Markup Language

**Filipa Alexandra Santos Cerdeira Mendes Moura**

Temporary Version

Project's Report
Integrated Masters in Informatics Engineering and Computing

Supervisor: Prof. Raul Moreira Vidal

June 2009

# Implementation of a Single Sign On solution using Security Assertion Markup Language

**Filipa Alexandra Santos Cerdeira Mendes Moura**

Project's Report
Integrated Masters in Informatics Engineering and Computing

Approved in oral examination by the committee:

Chair: _____

_____

External Examiner: _____

Internal Examiner: _____

June 2009

# Legal Terms

# Abstract

Today's dynamic market longs for innovative solutions that, not only add value for the business, but also improve the user experience and satisfaction. Single Sign On can be seen as a major step towards this direction.

This project's goal was to design a Single Sign On solution that was compliant with the Security Assertion Markup Language (SAML) standards.

Single Sign On allows an user to log in once to one application and gain access to multiple applications without any further actions, while SAML is an XML-based standard whose major goal is to solve the Web Single Sign On problem.

Adopting standards is an advantage and simplifies software interoperability: whether or not the software is made by the same vendor, implementing SAML standards makes it virtually possible to communicate with all the existing software that conforms to the same standard.

Research done about Single Sign On and SAML lead to an architecture description of how the solution should be structured in order to achieve all the requirements initially defined. Since the project's goal matched ALERT's objectives, a proof of concept was developed over ALERT® software.

After defining the architecture, state of the art technologies were studied to evaluate the software that best fitted the project's requirements. The main problem of evaluating the software available for SAML solutions was the complexity of the installation and testing processes. On the contrary of what was expected, local deployment and simple testing was harder and, in some situations, unachievable on the time that was available for the project.

After choosing OpenSSO Enterprise as the software to be used, a prototype was developed. OpenSSO implements SAML and is presented as the single solution for Web access management, federation and Web services security offered by Sun Microsystems.

This prototype was a huge stride to the project's success because it allowed a clearer understanding of SAML message flow and the configurations needed as well as it permitted to discover how to overcome OpenSSO restrictions.

Since the prototype covered all interactions and flows on a Single Sign On solution, the process of integrating the software with MyALERT® and ALERT® Online was simplified. These are the two ALERT applications that were used because they are online applications that offer a single point of entrance, presenting separated flows.

The tests performed were all successful and the conducted evaluation concluded that the framework is stable and that it could be easily extended and integrated with other applications because of its modular architecture.

The fact that the most relevant requirements were implemented, accomplishing a rather complete deployment of Single Sign On, but opens the door to the usage of SAML for authorization purposes. The versatility of SAML and the modularity of the

solution offer innumerous options to achieve a better user experience with simpler interactions and greater software interoperability.

# Resumo

O mercado actual aspira por soluções inovadoras que não só criem valor para o negócio mas, acima de tudo, que melhorem a satisfação e a experiência da utilização. O mecanismo *Single Sign On* é considerado como um grande passo para atingir os referidos objectivos.

O propósito deste projecto era implementar uma solução de *Single Sign On* que estivesse de acordo com os standards da *Security Assertion Markup Language* (SAML).

O *Single Sign On* permite ao utilizador autenticar-se numa aplicação e ganhar acesso a diversas aplicações sem necessitar de qualquer acção extra, enquanto que o SAML é um standard XML cujo maior objectivo é o de solucionar a problemática do *Single Sign On* em ambientes *Web*.

A adopção de *standards* é vantajosa e permite melhorar a interoperabilidade entre diferentes aplicações: mesmo que a aplicação não tenha sido desenvolvida pela mesma empresa, o facto de implementar SAML torna possível a comunicação entre todas as aplicações que estejam de acordo com o mesmo *standard*.

Foram realizados estudos sobre *Single Sign On* e SAML que permitiram elaborar a descrição da arquitectura da solução final, para que se atingissem todos os requisitos propostos. Dado que o propósito do projecto em causa está de acordo com os objectivos da ALERT, foi desenvolvida uma prova de conceito sobre *software* da ALERT.

Após a arquitectura ter sido definida, realizou-se uma análise às tecnologias actuais de forma a que se pudesse avaliar qual o *software* que melhor se enquadra nos requisitos do projecto. A maior dificuldade sentida no que toca à referida avaliação relaciona-se com a complexidade dos processos inerentes à instalação, configuração e testes do *software* avaliado. Ao contrário do que seria de esperar, a implementação local e a execução de testes às aplicações foi mais difícil do que o previsto e, em alguns casos, não atingível no tempo disponível para a realização do projecto em causa.

Da referida análise surgiu como escolha o OpenSSO Enterprise. O OpenSSO é a solução apresentada pela Sun Microsystems para gestão de acessos e segurança de serviços Web e federação, implementando SAML. Posteriormente foi desenvolvido um protótipo com base na supracitada decisão.

O protótipo assumiu um papel relevante para o sucesso do projecto dado ter permitido uma compreensão mais clara do fluxo de mensagens SAML e das configurações necessárias para a aplicação correcta do mesmo, mas também porque possibilitou descobrir como ultrapassar restrições do OpenSSO.

Como o protótipo foi extenso e se debruçou sobre todas as interacções e fluxos presentes numa solução de *Single Sign On*, o processo de integração da referida solução nas aplicações ALERT Online® e MyALERT® foi simplificado. Estas aplicações fazem parte das já aplicações que a ALERT oferece e foram escolhidas devido a serem aplicações online, com um único ponto de entrada, que apresentam fluxos separados.

Todos testes realizados apresentaram resultados positivos e a avaliação feita à referida *framework* conclui que a mesma se encontra estável e que pode ser facilmente expansível e integrada com outras aplicações devido à sua arquitectura modular.

Dado que os requisitos mais relevantes foram implementados, tendo sido atingida uma implementação extensa de *Single Sign On*, a solução permite que novos desenvolvimentos sejam feitos que podem incluir a utilização do SAML em acções de autenticação. A versatilidade inerente ao SAML e a modularidade da solução oferece inúmeras opções que permitem oferecer uma melhor experiência ao utilizador e maior interoperabilidade com outras aplicações.

# Acknowledgments

First and foremost I would like to thank everyone at ALERT for making this project possible: from the interview until today, I can say nothing but a big thank you for welcoming me as you did and making me feel at "home" since my first day.

Filipe Pereira, or *my boss* as I like to call him, was definitely one of the most valuable persons I met at ALERT: it provided me guidance when needed but also left me to discover the solutions alone, forcing me to grow up from a student to a company worker.

From my team, Technical Architecture Security, I have to also thank André Tavares for being able to put up with me during all this time but also for always being available to provide a helping hand when needed, and Joana Costa, that proved that two women on a manly company can still be friends and help each other.

Since this project involved many areas, I also have to refer Pedro Vilaça and Nuno Guerreiro from the Technical Java Architecture team, for helping me understand the Java layer and being patient with all my doubts and questions and Ricardo Pereira, from Technical Flash Architecture, that taught me Flash basics. André Cova and Alexandre Albuquerque were absolutely indispensable to helping me learn how to run and configure Apache Servers.

Vitor Monteiro and Nuno Martins are the project managers for MyALERT® and ALERT® Online, and their help for understanding their software's flow was also very important.

I can't end my list of friends at ALERT without thanking Diogo Coelho, Rui Marante, Mauro Martins, Nuno Freitas, Pedro Albuquerque, Rafael Santos, João Loureiro and João Ribeiro for making me feel welcome and for all the funny lunches and nights out.

Referring now to my home institution, FEUP, I have to thank to my responsible Raul Moreira Vidal, for helping and supporting me, not only during this project's course, but all through my university career.

Also, to all the OpenSSO support team from Sun, especially Bruno Bonfils, Pat Patterson and Emily Xu: their support was indispensable for learning how to configure OpenSSO.

Not directly related to this internship, but also worth mentioning, is my friend Tiago Nunes that provided me with wise advices on how to survive a working life with happiness and motivation.

To all my other friends, who walked my horse when I was working late or were patient enough to cope with my lack of time available: thank you! They know who they are so I won't mention any names here.

Last, but not least, my family and my housekeeper: thank you for all your support. And Mom, thank you for all the times you woke up early to prepare breakfast for me!

Filipa Moura

# Content

# List of Figures

# List of Tables

# Abbreviations

| | |
|---|---|
| AOL | ALERT® Online |
| HL7 | Health Level 7 |
| HTTP | Hypertext Transfer Protocol |
| HTTPS | Hypertext Transfer Protocol Secure |
| ICD9 | International Statistical Classification of Diseases and Related Health Problems |
| ICPC2 | International Classification of Primary Care |
| IdP | Identity Provider |
| JAAS | Java Authentication and Authorization Service |
| JVM | Java Virtual Machine |
| LDAP | Lightweight Directory Access Protocol |
| NHS | National Health Service |
| OASIS | Organization for the Advancement of Structure Information Standards |
| PHR | Personal Health Record |
| POC | Proof of Concept |
| RMI | Remote Method Invocation |
| SAML | Security Assertion Markup Language |
| SDK | Software Development Kit |
| SLO | Single Logout |
| SP | Service Provider |
| SSL | Secure Socket Layer |
| SSO | Single Sign On |
| TLS | Transport Layer Security |
| URL | Uniform Resource Locator |
| XACML | eXtensible Access Control Markup Language |
| XML | eXtensible Markup Language |

# 1. Introduction

This thesis presents the result of a project developed during four months, whose major goal was the implementation of a Single Sign On solution using Security Assertion Markup Language (SAML). The deployment and integration of such solution within all the existing frameworks was considered as the foremost factor to measure its success.

In order to thrive, the knowledge learnt during the Master studies, had to be applied for the resolution of the complex engineering problem. It was a multifarious project that required extensive research in order to offer an innovative solution that can replace the similar frameworks that already exist on the software.

Since the project presented many facets, it involved the collaboration of innumerous teams to approve the architecture and design which lead to an easier integration.

This project was developed for ALERT (more information about ALERT on Appendix A).

## 1.1. Context

Today's market is innovative and competitive. Traditional solutions are no longer feasible if applications are to thrive and conquer user satisfaction.

The amount of time spent every day on login actions by each user can no longer be seen as acceptable, changes have to be done to prevent such repetitive tasks.

The ability of run multiple applications at the same time, has led users to multitasking but having to login in each and every single application consumes time. Users usually tend to sign on into all applications at the same time, so they can later interact with them without having to be worried with access restrictions. Assuming that all applications are run on a browser and are available on the Internet means that the login actions include the time to load the welcome page, time for the user to enter his credentials (if the login form is on the welcome page, if not, more extra time to load the login form), time for the credentials to be validated and then time to be redirected to the restricted access area.

The average time spent on login actions is 20 seconds[1]. If a user wants to access his personal email, company email and a sports website, he will spend about one minute or more only in login actions. Also, because when a browser is closed, most of the user sessions are terminated, the user will have to login again to every single application. At the end of the day, the time spent on such tasks is no longer affordable. Let alone to consider the case when a user fails to remember his website credentials and has to ask for a password reset and wait for the email response.

A method called Single Sign On is the perfect solution for the above problem: as long as the providers agree on certain aspects, the user will only need to login once and he will automatically gain access to all the providers.

## 1.2. Motivation

The healthcare market, which was considered to be one of the most innovative and competitive markets [War88] and where time is critical, demands the best out of each action on the application in terms of efficiency and user experience. Given that the project was developed for ALERT, which is a company that focuses on offering software solutions for the healthcare market, it must be taken into account specific market concerns.

Currently ALERT presents an innovative suite of clinical documentation that represents the workflow of all the different staff that works on the healthcare industry. ALERT® Software displays and stores patient information and data through a modern view that respects international medicine standards as ICD9 or ICPC2. It also uses communication standards (such as HL7) to enhance its interoperability with multiple software applications.

The diversity and multiplicity of software that ALERT provides already offers some web-browser applications that can be accessed through the Internet. Alas, users' authentication in several applications, whether or not provided only by ALERT, is a complex and time-consuming task that could be overcome by using the current technologies for Single Sign On. According to the company's strategy, ALERT aims to achieve Single Sign On through the usage of standards that are currently in vogue on the healthcare market, which will allow great interaction with other applications. At the moment, SAML is the most used standard on clinical applications therefore it will be the adopted standard.

An XML based standard entitled Security Assertion Markup Language (SAML), offers a standard solution for Single Sign On that enables applications from different providers to settle on an agreement that allows the user to be logged in to one application and be recognized on the others providers website, whether or not they share the same credentials.

The use of Single Sign On relates to two important factors: time and money. It will reduce the time spent on sign on actions but also the costs of helpdesk calls related with sign on operations (such as password loss). It is also relevant the improvement it will bring to the user experience, thus increasing user's satisfaction.

The project should take into consideration the multiplicity of environments where ALERT software executes and interacts with: local applications, browsers (on a local network) and online (on the Internet). It should support the maximum browsers possible and should be developed in Java to be easily integrated with ALERT® software tier model.

[1] Time measured based on a sample of different login actions done by 10 different users.

## 1.3. Project

Since the market commands a better user experience to be offered, ALERT has directed its efforts to include a Single Sign On solution.

Something as simple as logging in to an application can turn into a hard task as users have different usernames/passwords for every application. The complexity increases as users need to keep track of each and every credential.

If time is money, let's picture a scenario where a doctor working at a hospital forgets one of his passwords and has to call the Helpdesk to recover it. Since he will not be able to access the information system the hospital provides for outpatient records he will have to wait to get the password back before he can take notes about the patient he is observing, which leads to time being spent on unproductive tasks.

The aim of this project is to reduce the time consumed by login tasks, and Single Sign On is the solution. Single Sign On (SSO) is a method of access control that allows a user to log in once and gain access to the resources of multiple software systems, without being prompted to log in again.

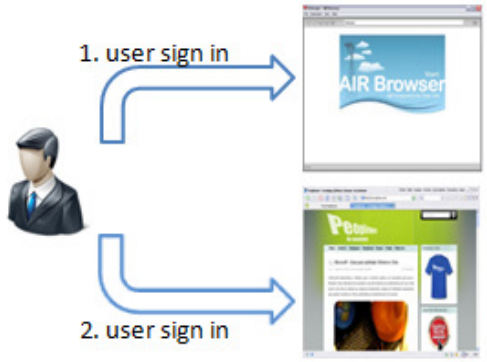


Figure 1: Regular sign on options on separate systems.

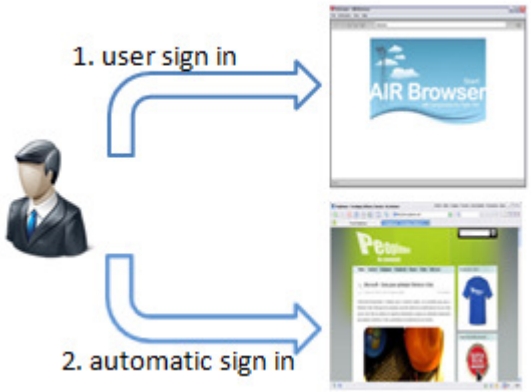


Figure 2: Single Sign On example.

The basic scenario of Single Sign On usage is demonstrated next. The MD John Doe wanted to log on into application A and afterwards to B:

- without SSO, MD John Doe has to log in into application A and then log in again into application B (Figure 1);
- with SSO, MD John Doe logs into application A and when he accesses application B he is automatically recognized and logged in (Figure 2).

Now, if instead of two applications, four or five were to be used at the same time (such as applications for outpatient, pharmacy prescriptions, lab requests and

scheduling), John Doe would eventually spend a lot of time logging into all applications. With SSO this is not necessary, therefore John's interaction with all applications becomes simpler and faster, which is exactly what the market aims for.

ALERT® software is getting broader and interacts with multiple applications from different providers. The goal of this project is to adopt standard protocols that support SSO, making interaction simpler not only for ALERT® applications, but with all of the other applications ALERT® software has to interact with.

### 1.3.1. Goals

The major goal of this project is to implement a Single Sign On solution that complies to the SAML standard. This solution should be as interoperable as possible and should be easily scalable. It should be presented as a framework that could be easily integrated with ALERT® software.

This framework should be as independent as possible from SAML, so that if in the future it is chosen to adopt another standard or another SSO technology, this can be easily done with no additional effort.

Even though, the framework is the final and most important output, there are other goals to accomplish so that the ultimate result is as successful as expected. These objectives include:

- Requirements analysis for a Single Sign On solution for various ALERT® applications;
- Document containing the Architecture description of the solution that will be implemented;
- Document containing an analysis of solutions already offered by companies, including system and extra requirements offered, as well as, the requirements that could not be achieved;
- Software library that achieves the following generic requirements:
  - Support for the various ALERT® applications as well as non-ALERT applications;
  - Support for Windows platforms (XP/VISTA/Others), Linux/Unix, and, if possible, Macintosh platforms;
  - Support for multiple authentication methods as biometrics, smartcard and password;
  - Provide a standard API that offers a transparent interface for ALERT® software;
- Library integration on various ALERT® applications;
- User documentation.

The conjunction of all the goals detailed above will follow the development cycle of functionality for critical environments, including functional analysis, prototyping, technical analysis, development, documentation and testing.

### 1.3.2. Schedule and Deliverables

According to the goals described in 1.3.1, the deliverables are:

- architecture description document;
- software provider comparison document;
- prototype;
- software library;
- and user documentation.

This project was developed during 20 weeks, which also contained training in ALERT® software and company procedures, as seen on Table 1.

Table 1: Project's tasks breakdown.

| Description | Week(s) | Days |
|---|---|---|
| Getting familiar with the company and its products | 1 | 5 |
| Project Requirement analysis, debate with the concerned areas and creation of the architecture description | 2 and 3 | 10 |
| Analysis of the existing solutions in the market and creation of a document for solution comparison | 4 and 5 | 10 |
| Discussion and decision of the solutions to acquire and what should be developed internally | 6 | 4 |
| Software project design | 7 | 5 |
| Software implementation and discussion of what requirements will be accomplished | 8 to 14 | 28 |
| Project documentation for the Master thesis | 15 and 16 | 8 |
| Integration of the solution with the ALERT® software and testing | 17 and 18 | 9 |
| Documentation of the developed functionalities | 19 and 20 | 10 |

The project's Gantt Diagram is included in Appendix B.

## 1.4. Report Overview

This dissertation is composed by six chapters, starting with an overview of the project. It is presented the motivation and context for the project as well as its goals and its schedule.

It is followed by the state of the art analysis, containing the solution specification that presents a Single Sign On description as well as a description of the SAML standards.

Chapter four contains the initially defined architecture for the solution, being composed by all relevant architectural views.

On the fifth chapter, state of the art technologies, such as software applications or libraries that implement SAML are analyzed in order to evaluate the one that best fits the previously defined architecture. This chapter also includes the specification of the prototype that was developed.

The sixth chapter presents the proof of concept. It starts with a brief explanation of the applications it involves and goes to the details of the adapted architecture. It provides the implementation's fine points and the output of the tests that were conducted. The chapter ends with a critical evaluation of the solution.

The final chapter presents a review of the project with all its positive and negative factors. It also announces perspectives for future developments.

# 2. Technical Overview

Many authors consider authentication as one of the key aspects of cryptography and network security. Single Sign On is an authentication process that permits a user to enter its credentials once and gain access to multiple applications.

However, it should be understood that SSO is not making a stand only in the healthcare industry:

- Windows Live ID was developed and provided by Microsoft to allow users to sign on to many websites using only one account [Wik091];
- Microsoft also offers SAML support on its Federated Identity and Identity Metasystem "Geneva" [Mic09];
- Google provides its own SSO service that allows partner companies to access Google's web-applications like Gmail or Google Calendar [Goo09], through the utilization of SAML;
- and on the economical sector SSO is also being implemented: on [Sah08] a case study for a SSO solution that provides a single authentication interface to all end users on the International Bank of India is presented.

The idea is spreading and it is starting to be implemented by every company that wishes to prevail and thrive in the competitive market. Recent debates among numerous Chief Information Officers show that Single Sign On is something that is on their radar screens as an important or high priority requirement [Sah09]. A survey conducted by the Healthcare Information and Management Systems Society (HIMSS) Analytics showed that 79% of IT executives ranked SSO as the highest priority for the next two years [Dav09].

The implementation of a Single Sign On solution is evaluated on the case study for a dynamic healthcare portal site for the National Taiwan University Hospital. Even though there were many positive feedbacks, users were unsatisfied with the solution offered. The fact that it presents a list with over 250 function links that allows users to access other sites it originated many users complains due to the delay injected by the time spent on finding and choosing the desired site from the broad list [Yun07]. This proves the point that Single Sign On should be achieved in a simple way that does not require linkage between all sites that implement it.

Sentillion, a company focused on Identity and Access Management for healthcare, has already created an SSO Solution that was developed exclusively for healthcare and it is being adopted by hospitals. Nevertheless, it does not use standards which reduces

(and might annul) its interoperability with other software applications that do not use the same Sentillion solution.

The Security Assertion Markup Language is a well known standard among the healthcare industry and its utilization has lead to successful cases such as the NHS Spine and the Mayo Collaborative Services:

- The United Kingdom uses NHS Spine, a national network that stores patient information while also offering interfaces with all the local IT Systems within the National Program but, most importantly, Spine uses SAML for SSO operations;
- The famous Mayo Clinic in the United States, through the Mayo Collaborative Services, also supports SAML for access to critical information resources and content.

Supporting SAML will open the system for federation scenarios aside from providing interoperability with systems which implement this industry standard. Even though not related to the healthcare industry, SAML is already being used by the Citizen and Agency e-government services in Portugal.

## 2.1. Single Sign On

### 2.1.1. Description

Single Sign On is a mechanism whereby a single authentication action grants access to all other systems where the user has access permission.

Since information systems start to proliferate to support business processes, users start to face manifold sign on actions to several systems, where the user credentials are usually different.

For the following example, it will be assumed that the user credentials for each system are the same.

On legacy systems, even though they may work under the same enterprise, they both have their own authentication framework as well as user information management and database: the user signs in into the primary domain and is required to sign on to every other domain he wishes to access. User information is not correlated between domains and usually for each domain the user has its own set of credentials.

As seen on Figure 3, every domain has its own Authentication Framework and User Account Manager. The Authentication Framework is responsible for validating the users' credentials and the User Account Manager will handle its sessions. The Domain and Resources represent specific application data and logic.

For each domain, the user has a correspondent set of credentials. Therefore, for each login action on a separated domain, the user has to provide the correlated set of credentials.

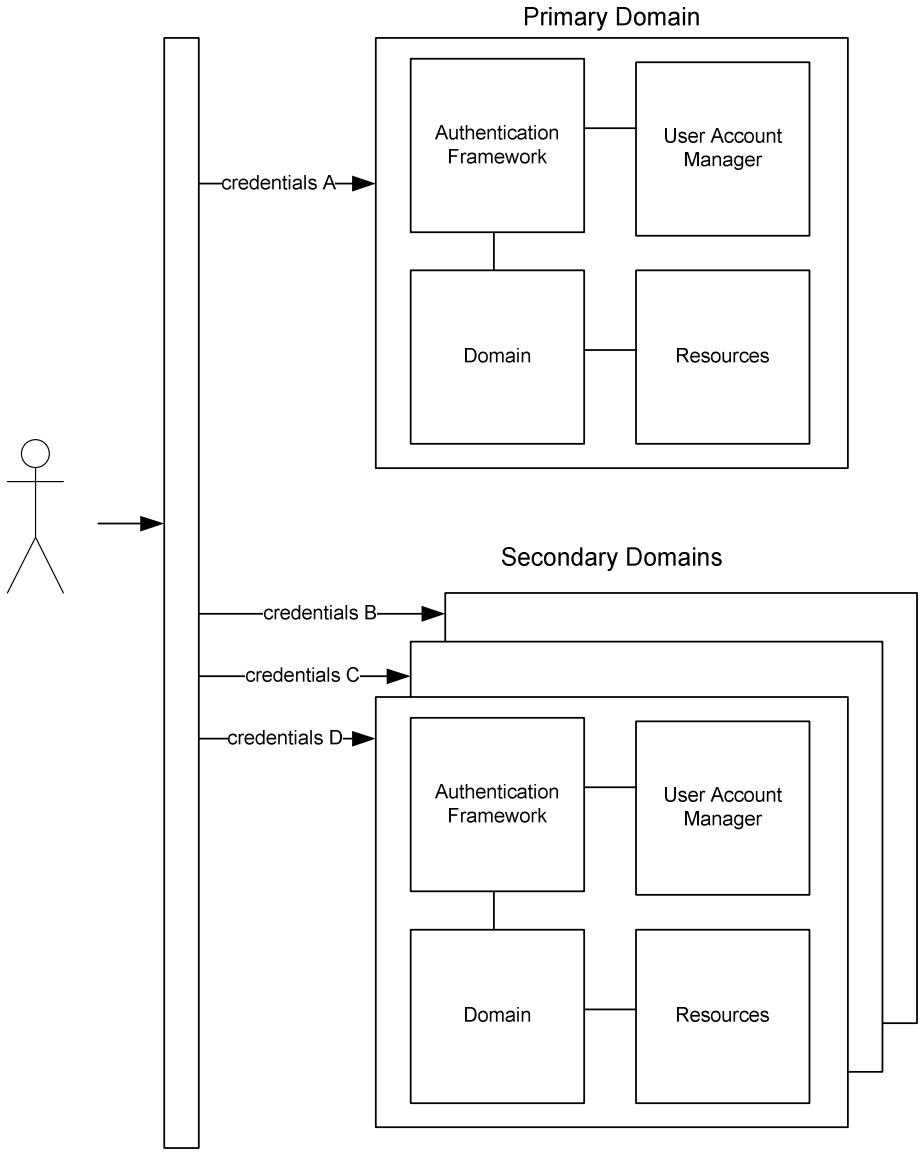


Figure 3: Legacy systems approach to sign on actions.

On current systems, this approach is no longer feasible: if a user usually works under a usual set of domains, the amount of time spent on sign on activities has to be cut down to the minimum. Single Sign On achieves time reduction on sign-on operations to individual domains which also includes the reduction of the probability of such sign on operations failing.

On a system that provides SSO there is a primary entity for user account management (Figure 4). After a user is logged in into a primary domain, this information is communicated to the user account management which will then handle all other secondary sign on operations. This approach eliminates the need for the user to provide a set of credentials for each domain, providing them to the primary domain is enough.

The User Account Manager is responsible for communicating with the secondary domains and manage the user sessions on each one of them. This is possible due to the existence of a trust relationship between all of the domains.

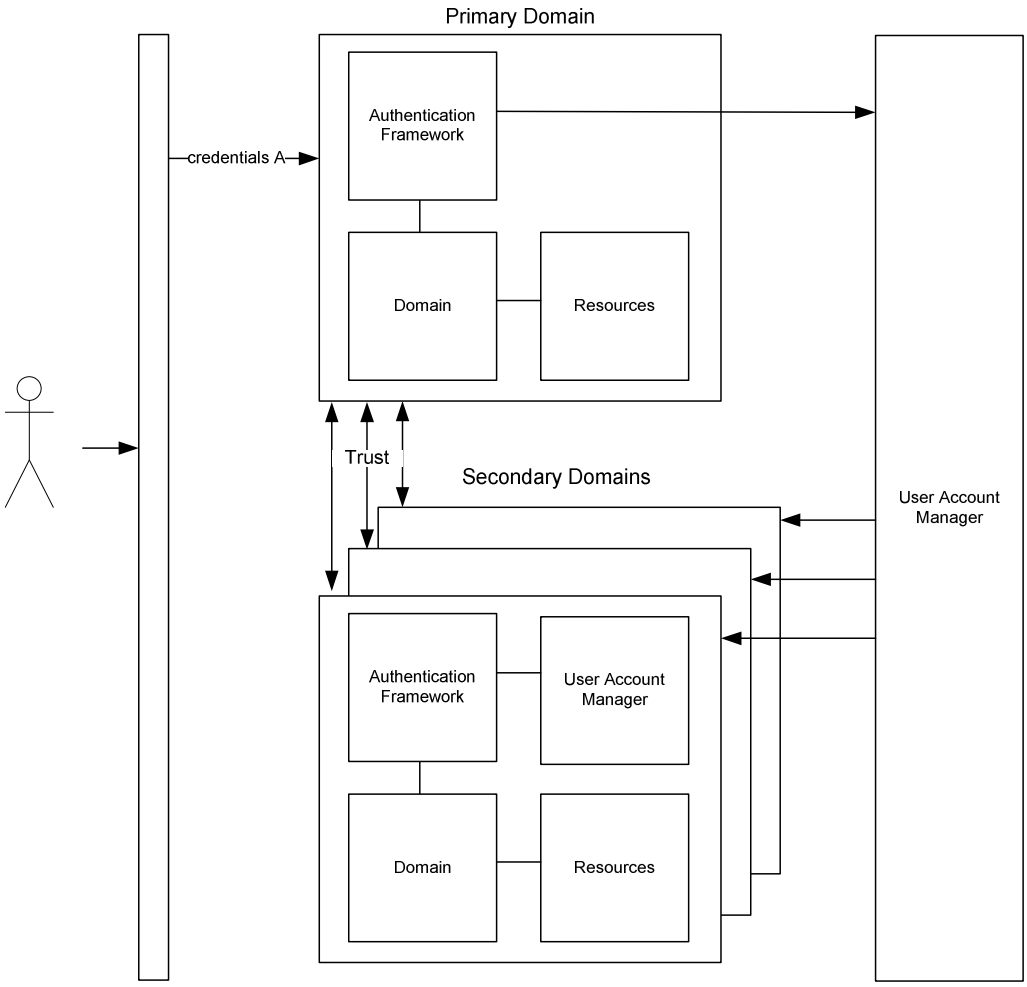


Figure 4: Systems sharing an User Account Manager for Single Sign On.

However, these actions are not quite as trivial as they may appear, since the information exchange can occur in different ways and this needs to be defined according to the company's needs and security measures. But on most of the actual scenarios, the user does not have only one type of credentials for all the systems.

Therefore, the next step is to consider a scenario where the credentials for each service are unique, not only in content but also in type (Figure 5).

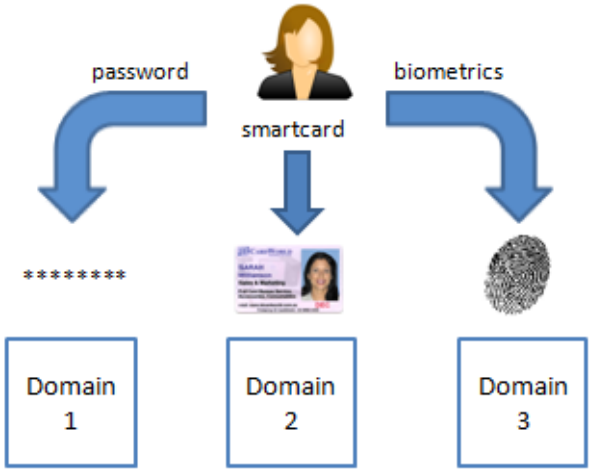


Figure 5: Multiplicity of authentication methods and domains.

The complexity of SSO arises as the need to find a solution for such diverse authentication systems increases and the answer resides on Identity Federation. Identity Federation has been named "the *linchpin of digital convergence*" and "*one of the most important technologies of the modern era*" [Sun09].

Every person has their own separated digital identity for every service they need to interact with (such as their bank or their phone company) but, in some cases, the companies may decide to offer a new service that has to encapsulate multiple services from various providers.

However, situations where each company has its own set of customer data and decide to share it are not viable options either for business or legal reasons. And in case companies do choose to share it, they will face synchronization problems.

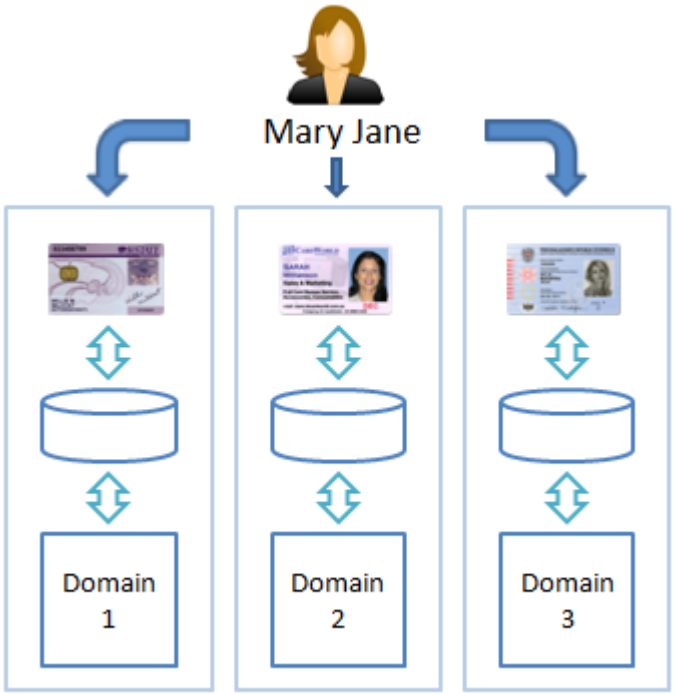


Figure 6: One single user (Mary Jane) has different user identities for each domain.

On Figure 6 the same user, Mary Jane is registered on three different domains and, for each, she has a specific profile (represented by the distinct identity cards). In this case, sharing specific profile details is not considered acceptable.

Identity Federation means that an agreement between the providers exists and that the user identity will be referred by a set of identifiers [OAS09]. The utilization of identifiers eliminates the need to share specific profile details.

The identifier could be something as simple as the customer's email or address. This approach annuls the need for the providers to share the customer information, yet making it possible to map the customer across different providers.

Federation is a very important part of SSO since it allows accomplishing SSO in a manner that is transparent to the user, even though a user may have different credentials for each provider.

The advantages that result from using SSO should now be clear and straightforward to understand and include:

- reduction of the time spent on log in actions;
- reduction of security complexity related to sign on operations;

- reduction of password fatigue that arises from different user name and password combinations;
- reduction of IT costs due to decrease of calls to the Helpdesk regarding password loss;
- strengthening and centralization of user access control;
- improvement of report and monitoring for regulatory compliance.

## 2.2. Single Logout

After SSO has been performed it should be possible to execute a single logout. Single Logout (SLO) permits near real-time session logout from all active sessions associated with a user.

When a user performs SSO, at each request he is logged in to *one* extra application. On the other hand, Single Logout allows multiple logout from *all* applications with one simple action.

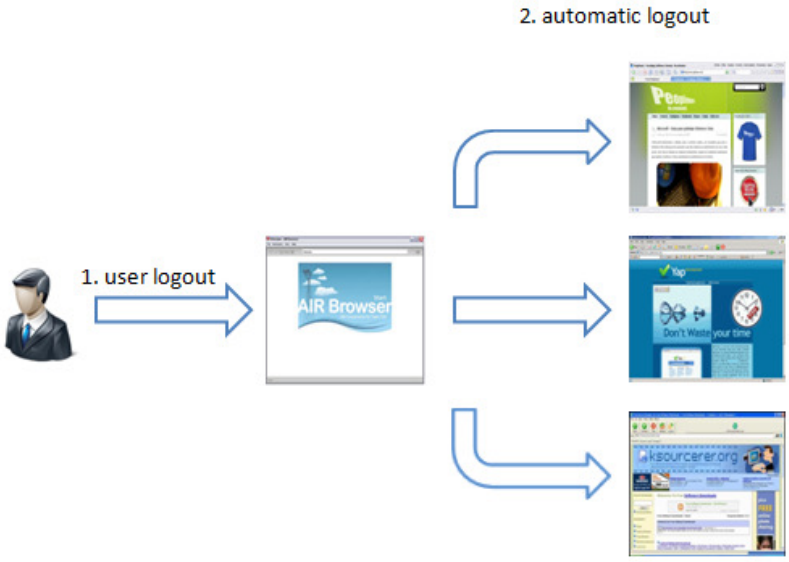


Figure 7: Single Logout overview.

On Figure 7, the user requests logout from one application and is automatically logged out from all other applications where he shares a SSO session.

Besides allowing a faster logout for the user, thus simplifying its experience and improving its satisfaction, SLO benefits go further [FEI09]:

- since SLO destroys all SSO sessions, they will be no longer available for hijacking attempts;
- because service may close down in a controlled manner, the risks of lockups or problems due to instability are minimized;
- cleanup is done and resources can be release sooner.

## 2.3. Scenarios

Now that the aim of the solution has been defined and the concept is becoming clearer, there are some entities that need to be understood before proceeding to a more detailed and technical view. There are three major intervenients that will participate in a SSO/SLO scenario are:

- Identity Provider – manages credentials of the individual end users and verifies if they are valid. Rather than being just a simple database where the user credentials are stored, it also presents logic that allows to authenticate and access information about users;
- Service Provider – entity that provides services. It stores the resources that the user wishes to access thus needs to know whether or not the user is authenticated;
- User Agent – Represents the application the user wishes to interact with and is responsible for providing the requested resources to the user and managing the exchange of messages between Identity and Service Provider.

A simple and usual interaction could be exemplified by the situation when the user wants to access protected resources from another system. The user would have to authenticate himself at the Identity Provider, firstly, and then request access the resource stored on the Service Provider.

In order for the Service Provider to allow the user to access the resource, it needs to know the user with whom it is interacting. Therefore, it will need to communicate with the Identity Provider to become acquainted with the user's identity.

Figure 8 intents to clarify the interactions that occur on the system. The user interacts with the user agent who in turn will interact with both Service and Identity Provider. It is important to refer that the Service and Identity Provider will never communicate directly, but always through the User Agent.

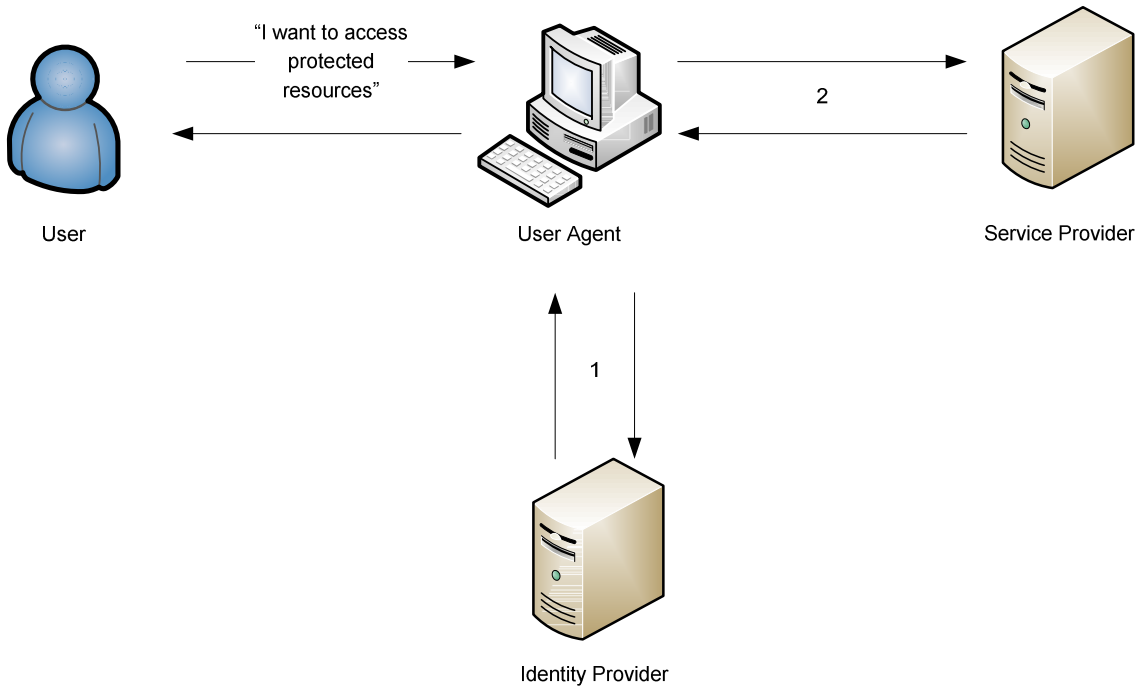


Figure 8: Context view for Single Sign On.

It must also be referred that there will always be one and only one Identity Provider but there can be multiple Service Providers. A practical example can be found on Google: while Google plays the Identity Provider role, Gmail and Greader can be seen as different Service Providers.

It should also be noticed that these interactions are possible because there is a trust relationship between Identity and Service Providers that allows them to confide in the information the other provider transmits.

Real life examples are probably the best way to understand how SSO actions will be handled by the Providers therefore in the next sections, simulations of different user interactions relative to sign on operations will be presented.

### 2.3.1. Single Sign On

The first scenario for SSO at a Service Provider assumes that the user has already authenticated himself on the Identity Provider. It is the simplest scenario because no user credentials are involved, they were already handled during the sign on at the Identity Provider.

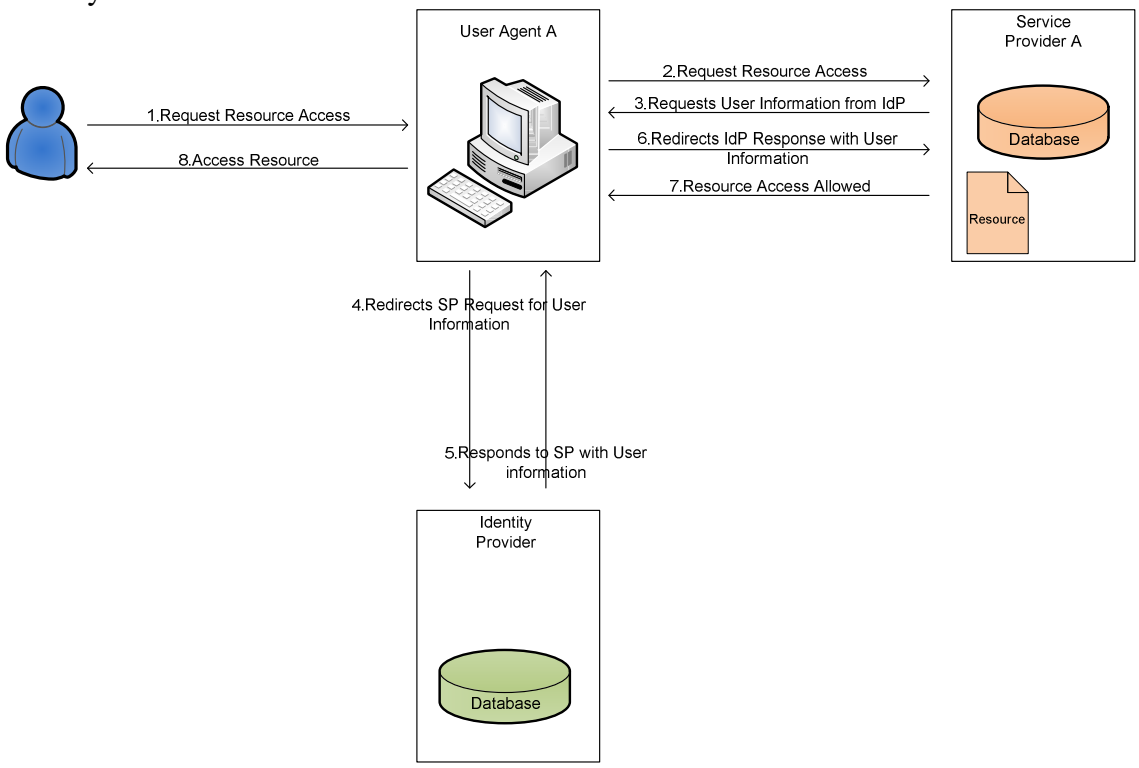


Figure 9: Single Sign On actions flow.

It starts with a user that wishes to access Service Provider A, to a specific resource, say a user that wants to read his company's email (step 1 and 2). Since this resource is only available for registered users, it will need to contact the Identity Provider to become aware of whether the user is authenticated or not, so it can propagate its session to the Service Provider (step 3 and 4).

Since the email reader will only grant access to authenticated users, and only to the receiver of such emails, it will interrogate the Identity Provider if a valid user session exists and waits for the response. In this case, the Identity Provider can be represented by the company's website,

The "user information" referred on the diagram is simply a general designation that implies that some type of information, related to the user, was shared. It allows the user to be identified and could be, in this case, the user's email. Nevertheless, the details about this information will be discussed later.

After contacting the Identity Provider, and in case the user information that the Service Provider A received is valid (step 5 and 6), the user will be automatically signed in to Service Provider A and will be granted access to the requested resource (step 7 and 8). Meaning, if there is really a user with the given email, the email reader will grant the user access to his private emails.

It should also be referred the role that the User Agent plays on these operations, which is redirecting the requests from the Service Provider to the Identity Provider, and the responses from the Identity Provider to the Service Provider.

### 2.3.2. Identity Federation

On the latter scenario, it was mentioned that some type of user information was sent from the Identity Provider to the Service Provider. This is related to Federated Identity and to the different mechanisms supported by SAML for establishing and managing federated identities. The method that will be used is entitled "Federation via Transient Pseudonym Identifiers".

The mechanism forces user information to be sent and, the main reason for it to be called transient, is because a temporary identifier between the Identity Provider and the Service Provider is set during the user's SSO session.

The motive for using this type of federation is that it avoids having to manage user IDs at the Service Provider, supporting a truly anonymous service.

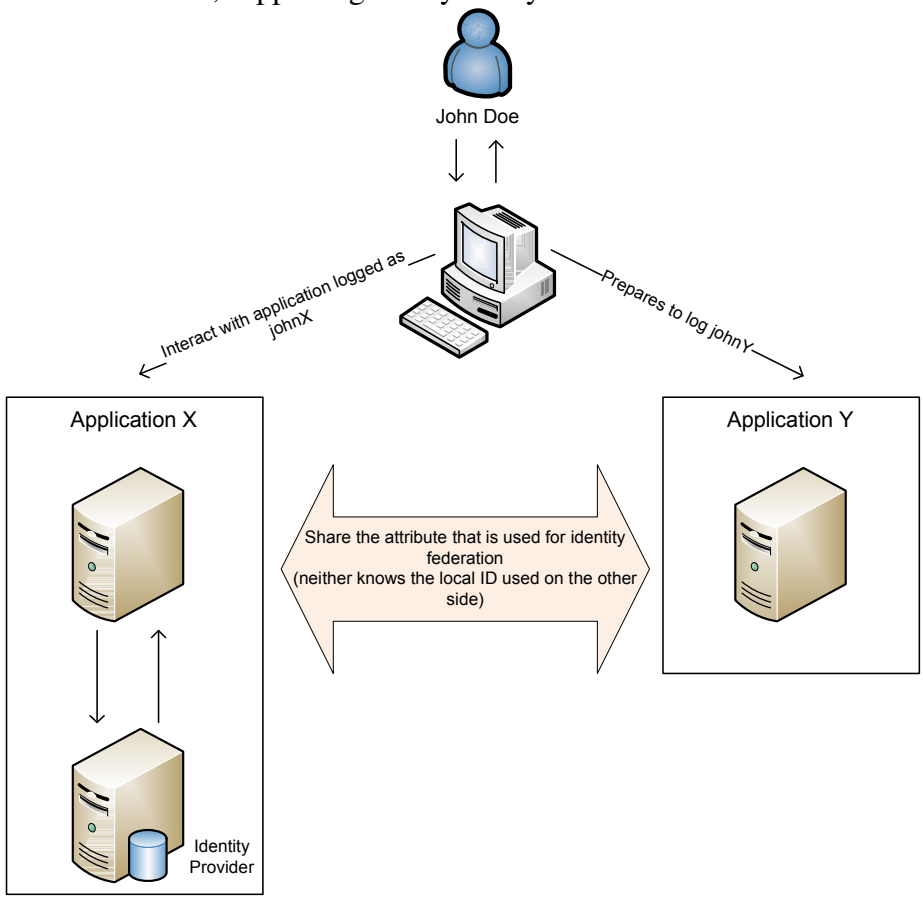


Figure 10: Identity Federation.

As an example, the user John Doe has logged in onto application X and now he wishes to access application Y. His username on X is *johnX*, and on application Y is *johnY* but he has registered with the same email on both. Application Y will perform a request for Single Sign On and wait for application X answer. Then it will check the attribute that it has received, trying to map it to a user on its own database. In this case, the attribute used could be the email. If this operation is accomplished, SSO is successful.

### 2.3.3. Single Logout

The following scenario assumes that the user has already authenticated himself at the Identity Provider and performed SSO to Service Provider A and B. Thus, at the moment, the user has three active sessions: at the Identity Provider, Service Provider A and Service Provider B.

Considering the first described scenario, the user has currently an active account at its company's website, on the email reader but also at the website for project management. He shares a workstation and wishes to sign off every application with one single action therefore he requests logout, which the company decided to implement as Single Logout.

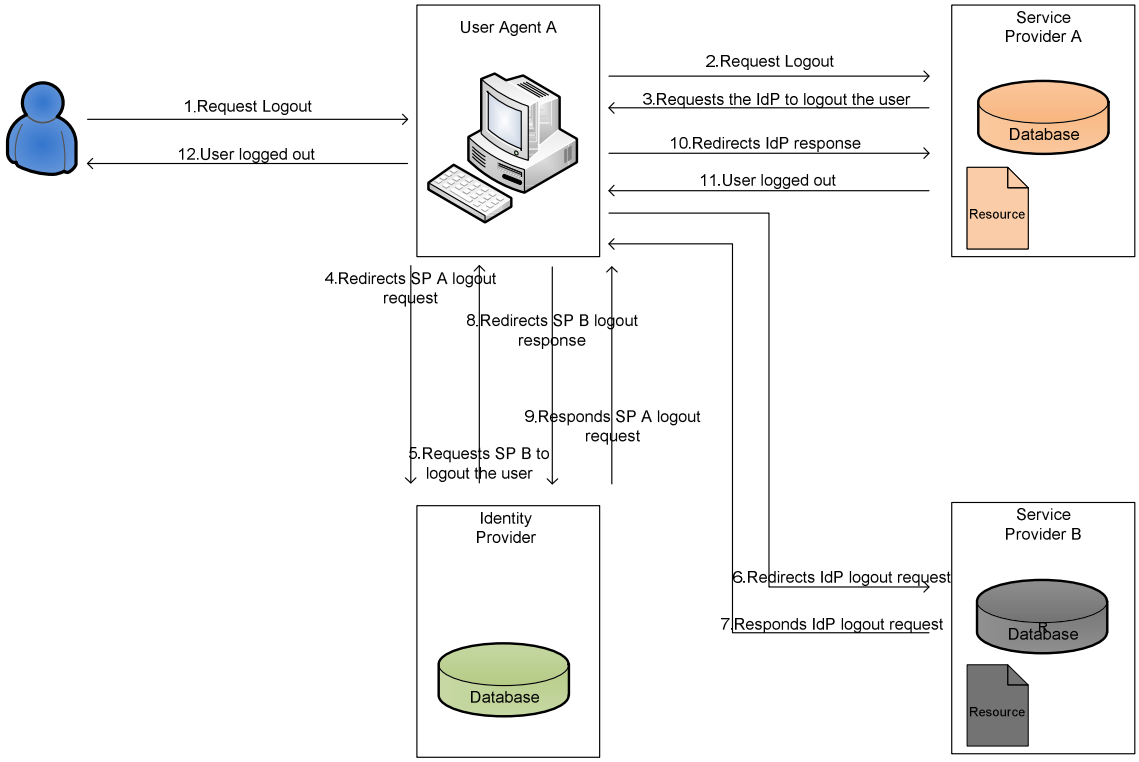


Figure 11: Single Logout actions flow.

The user requests SLO to Service Provider A, the email reader. SP A will communicate with the Identity Provider which is responsible for propagating the decision to all the other Service Providers where the user has an active session, to the company's project management website in this case (steps 5 to 8). In Figure 11 the Identity Provider communicates the user logout request to Service Provider B, and afterwards replies to Service Provider A's initial request (step 9 and 10).

Again, the User Agent also only plays the role of message redirecting.

## 2.4. Security Assertion Markup Language

Now that the interactions between the providers are clear, it is time to understand how SAML will be used to achieve SSO. But first, it is important to clarify the value of standards.

Technical standards are established norms or requirements [Wik092] and businesses can benefit by using them as strategic market instruments. Standards present

innumerous benefits, but the focus will remain on the ones that the planned solution will benefit from.

Standards are mostly related with the need of interoperable software, since companies prefer not to be locked to specific technologies or vendors. Since standards are usually open, it is usual that other providers will implement the same solution according to standards. Therefore it will be easier for systems from different parties to interoperate and communicate with one another, improving data exchange. This is the major reason why standards should be used: to ensure better and easier interoperability and eliminate the risk that the software developed is dependent on some kind of vendor software.

Also, the utilization of standards simplifies product development and reduces non-value-adding costs thereby increasing the ability to compare competing products. It promotes effective research and development which can, and usually leads to, the making of products that are easier to use.

Having clarified the magnitude that applying standards can conquer, the details of SAML will be described next. After having a basic understanding of the SAML structure, it will be demonstrated how to take advantage of it for the scenarios previously described.

### 2.4.1. Specification

Security Assertion Markup Language is an XML-based standard developed by the Security Services Technical Committee of the Organization for the Advancement of Structured Information Standards (OASIS). Its major goal is to solve the Web Single Sign On problem [Eve09].

As a framework for communicating user authentication, entitlement and attribute information, SAML allows business entities to make assertions regarding the identity, attributes, and entitlements of a subject to other entities, such as a partner company or another enterprise application, therefore allowing the exchange of security information between on-line business partners.

SAML is a flexible and extensible protocol that was designed to be used by other standards. SAML v1.0 became an OASIS Standard in November 2002. Shortly after, version 1.1 was released and accomplished great success, gaining momentum in important industry segments like education, government and financial services. It was implemented by all major Web access management vendors. SAML v2.0 followed the successful path.

It was designed to be interoperable with other standards such as the Liberty Alliance Project, the Shibboleth project and the OASIS Web Services Security standard.

SAML v2.0 was the triumphant feature upgrade from v1.1. It presents a critical step towards full convergence for federated identity standards since it unifies disparate federated identity building blocks from v1.1 taking into account Liberty's Identity Federation Framework (ID-FF), capabilities present in the Internet2's Shibboleth architecture and enhancement requests resulting from experience from SAML v1.x deployments in the industry [OAS92]. Figure 12 shows the evolution path that lead to SAML v2.0.

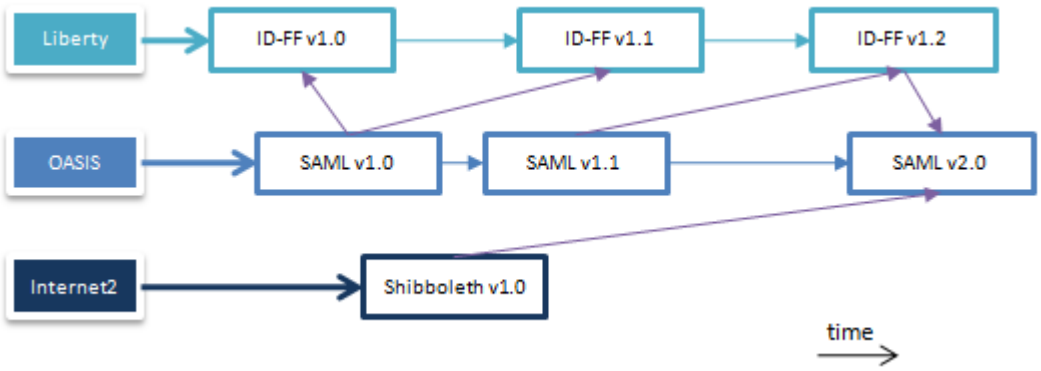


Figure 12: Relationship between SAML, ID-FF and Shibboleth.

Nowadays v1.0 is considered to be obsolete, while v1.1 is still used and v2.0 is the latest. Alas, on-the-wire representations of SAML v2.0 assertions and protocol messages are incompatible with v1.x processors thus, from here on, when SAML is mentioned it refers to v2.0 except when specifically mentioned otherwise.

Several reasons lead to adoption of SAML standard for the exchange of security information, including:

- Single Sign On – even though various products have claimed to provide web-based SSO, they relied only on browser cookies that do not allow that the authentication state information is available to other domain. Therefore complex mechanisms had to be developed which did not work on heterogeneous environments;
- Federated Identity – SAML has implemented protocols that simplify the creation of federated identity;
- Web Services and other standards – SAML allows that the security assertion format does not have to comply with the protocol context also, its modularity simplifies industry efforts to address authorization services [OAS091].

Because it is a standard, SAML promotes interoperability by providing a set of standard interfaces which allows for faster, cheaper and more reliable integration between systems.

More concrete benefits that derive from SAML are related to [Car09]:

- Platform neutrality – it abstracts the security framework from vendor implementations and architectures, allowing dynamic integration of existing security infrastructures;
- Loose coupling – information does not need to be synchronized between directories or in identity information systems;
- Improved online experience for end users – enables users to authenticate at the Identity Provider and then access Service Providers without additional authentication;
- Reduced administrative costs for Service Providers – reduces the cost of maintaining account information since it is all stored on the identity provider;
- Risk transference – can push responsibility of the identities management to the Identity Provider, which is more suitable with its business model.

SAML is composed by four different types of components as seen on Figure 13.

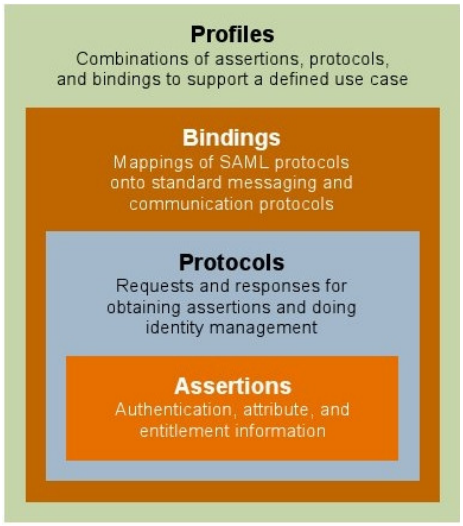


Figure 13: SAML components [OAS09].

*Assertions*

An assertion is a package of security information that provides statements about the subject [OAS098].

There are three kinds of assertion statements:

- Authentication – typically generated by a SAML authority that successfully authenticated a user and should at least describe the particular means used to authenticate the user and the specific time at which the authentication took place;
- Attribute – contains specific attributes of the specified subject;
- Authorization Decision – related to the actions the subject is entitled to do.

*Protocols*

SAML supports multiple protocols but the most used for SSO are [OAS099]:

- Authentication Request Protocol – allows a principal to request assertions containing authentication statements and, optionally, attribute statements;
- Single Logout Protocol – provides a mechanism that permits near-simultaneous logout of all active sessions that are associated with a principal.

*Bindings*

SAML Protocol Bindings detail how to map SAML protocol messages exchange into transport protocols. There are bindings defined for HTTP as well as for SOAP.

*Profiles*

A profile defines how to combine SAML components in order to enhance interoperability in particular usage scenarios. By defining constraints and extensions it can remove some of the flexibility inevitable in a general-use standard.

One of the profiles is the Web Browser SSO Profile that details how to use the Authentication Request/Response Protocol in conjunction with different bindings so as to accomplish SSO with standard web browsers.

## Profiles

Since SAML profiles are the major intervenients on the SSO and SLO events, the profiles related to these actions will now be described more thoroughly.

### *Web Browser SSO Profile*

The Web Browser SSO Profile provides innumerous options within two dimensions of choice: who initiated the message flow (either the Identity or the Service Provider) and which bindings are used to deliver the messages.

During the implementation, SSO will always be SP-initiated. On the most common scenario the user decides, at some point after having authenticated himself, to access a resource on the SP. Since the user is not logged in at the SP, it will be redirected to the IdP to authenticate. The IdP will create an assertion that represents the user's authentication and afterwards it will be sent back to the SP which processes it and determines whether or not to grant the user access to the resource.

When it comes to the bindings, there are many combinations of message flows and bindings that are possible, but the major concern is on the Authentication Request Message and its Response.

The profile defines two SAML messages to be used:

- Authentication Request Protocol <*AuthnRequest*> - represents the SSO request;
- Response <*Response*> - contains the response to the SSO request.

### *Single Logout Profile*

Single Logout permits near real-time session logout of a user from all participants in a session. The Single Logout profile allows reversing all the sign-on process of all the providers at once.

A user that visits an SP decides to log out of its web SSO session and issues a request to the IdP. The IdP will process the request and destroy the local session information about the user. Afterwards, it determines all other service providers where the user also has a valid session and will request them to logout the user. After this has been accomplished, it will respond to the initial Service Provider with the status of the action.

The profile defines two SAML messages to be used:

- Single Logout Protocol <*LogoutRequest*> - represents the SLO request;
- Response <*Response*> - contains the response to the SLO request.

## 2.4.2. Exchange

Now that the flow of the actions has been defined, it is time to become acquainted with the part that SAML plays during SSO and SLO. Nevertheless, before presenting SAML samples of messages exchange, some security issues SAML facesare presented next as well as possible solutions to overcome them.

Alas proving simple SAML assertions may not be adequate to ensure a secure system. In order to prevent "man-in-the-middle" attacks (such as spoofing), SAML specifications define a number of security mechanism.

The specifications also suggest that the trust relationship between user agent and identity provider should already be established and rely on a Public Key Infrastructure.

When it comes to the exchange of SAML:

- For transport-level security, one should use HTTP over SSL 3.0 or TLS 1.0;

- When a message containing an assertion is delivered to a relying party via a user's web browser, to ensure message integrity, it is mandated that the response message be digitally signed using XML Signature.

**Transport Level Security**

For Single Sign On it is important to secure the message transport, and since the user agents will be running on a browser, HTTPS will be used since it refers to the combination of a normal HTTP interaction over an encrypted SSL or TLS connection.

**XML Signature**

XML Signature is a W3C recommendation that defines XML syntax for digital signatures. Since response containing assertions will be sent, and it is important to ensure message integrity, XML Signatures will be used on responses.

On the following sub-sections SAML exchange will be detailed through SAML code samples. These samples should be seen as mere examples, since the messages can contain other XML elements and attributes, as long as they comply to specification.

The scenarios are equal to the ones detailed earlier: the focus here will be only on the exchanged SAML messages.

## User Single Sign On

For Single Sign On, the “Web Browser SSO Profile” will be used. On this section the focus will be on the “Authentication Request Protocol” used.

The following diagram presents a scenario in which a user wishes to access a resource on Service Provider A. It is assumed that the user has a valid authentication context at the Identity Provider.

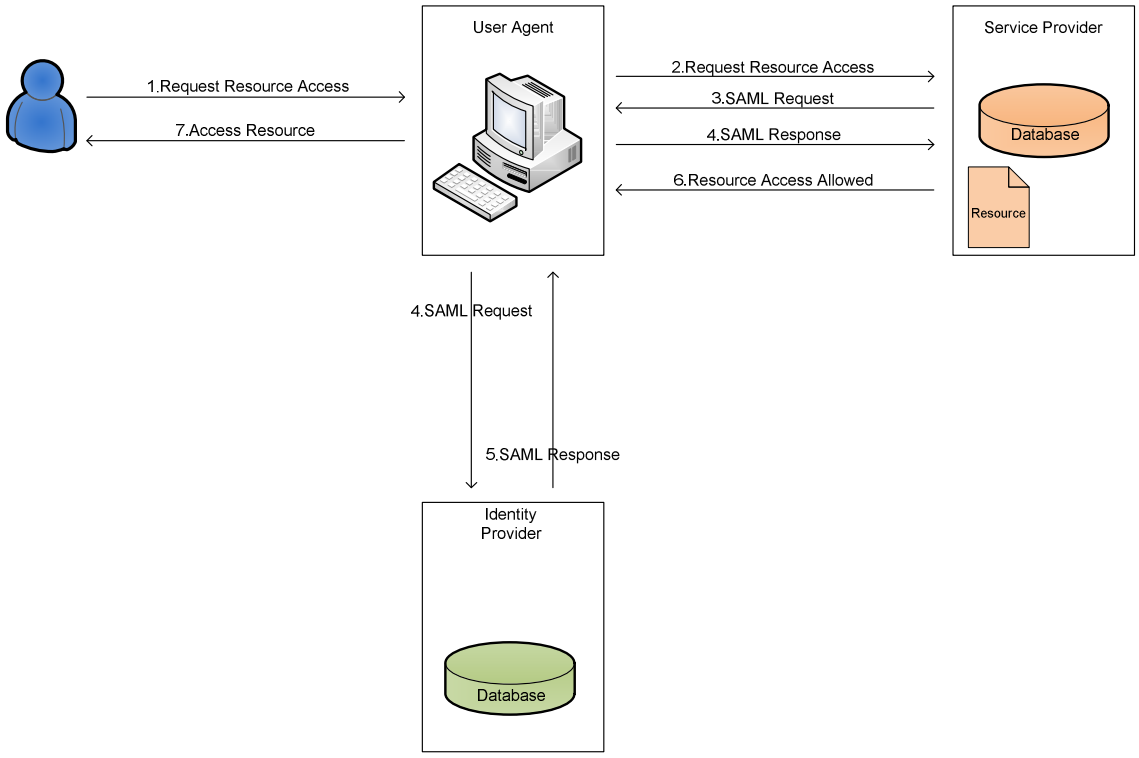


Figure 14: Single Sign On SAML exchange.

On step 3 the Service Provider A asks the Identity Provider if the user has a valid authentication context (step 3). This "question" is encapsulated as an *AuthNRequest* (Figure 15).

As already mentioned, because the Service and Identity Provider do not communicate directly, Service Provider will send the request to the User Agent A which will forward it to the Identity Provider (step 4).

```
<samlp:AuthnRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
AssertionConsumerServiceURL="http://www.sp.example.com/SSO"
AttributeConsumingServiceIndex="0" ProviderName="acompany.com" ID="abe567de6"
Version="2.0" IssueInstant="2005-01-31T12:00:00Z"
Destination="http://www.idp.example.com/"/>
```

Figure 15: Single Sign On authentication Request example.

This request includes:

- the namespace;
- the URL for the Assertion Consumer Service (the module that processes assertions), in this case http://www.sp.example.com/SSO;
- the provider name, in this case "acompany.com";
- the id of the request;
- the SAML version being used, in this case 2.0;
- the issue instant at which the request was made, in this case *31T12:00:00Z*;
- the destination, in this case http://www.idp.example.com.

The Identity Provider will then have to process the request and issue a response to the Service Provider.

```
<samlp:Response xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
mlns:samlp="urn:oasis:names:tc:SAML:2.0:assertion" Version="2.0" ID="abe567de6"
InResponseTo=" abe567de6" IssueInstant="2005-01-31T12:00:00Z"
Destination="http://www.sp.example.com/SSO">
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        [...]
    </ds:Signature>
    <saml:Issuer> http://www.idp.example.com </saml:Issuer>
    <samlp:Status>
        <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
    </samlp:Status>
</samlp:Response>
```

Figure 16: Simple Single Sign On Response example.

The response is structure in a similar way of the request but:

- it is digitally signed (<*ds:Signature*>);
- it states the issuer of the response;
- and that the authentication operation was successful.

This response will then be sent to the User Agent (step 5) which will forward it to the Service Provider (step 6). At this moment that the Service Provider knows the user is valid and it has an active session, it grants him access to the resource (step 7 and 8).

## User Sign In

Now, it will be assumed that the user has not yet logged in to the Identity Provider.

Since the Identity Provider is the component responsible for user authentication, the user will have to be redirected to it, for the user's credentials to be validated.

Even though this is not the usual SSO scenario, it can also occur and will proceed in a similar way to regular SSO, where the user is already authenticated at the Identity Provider.

The SAML protocol used will be the same as before, but at some point, the user will be redirected to the Identity Provider for authentication.

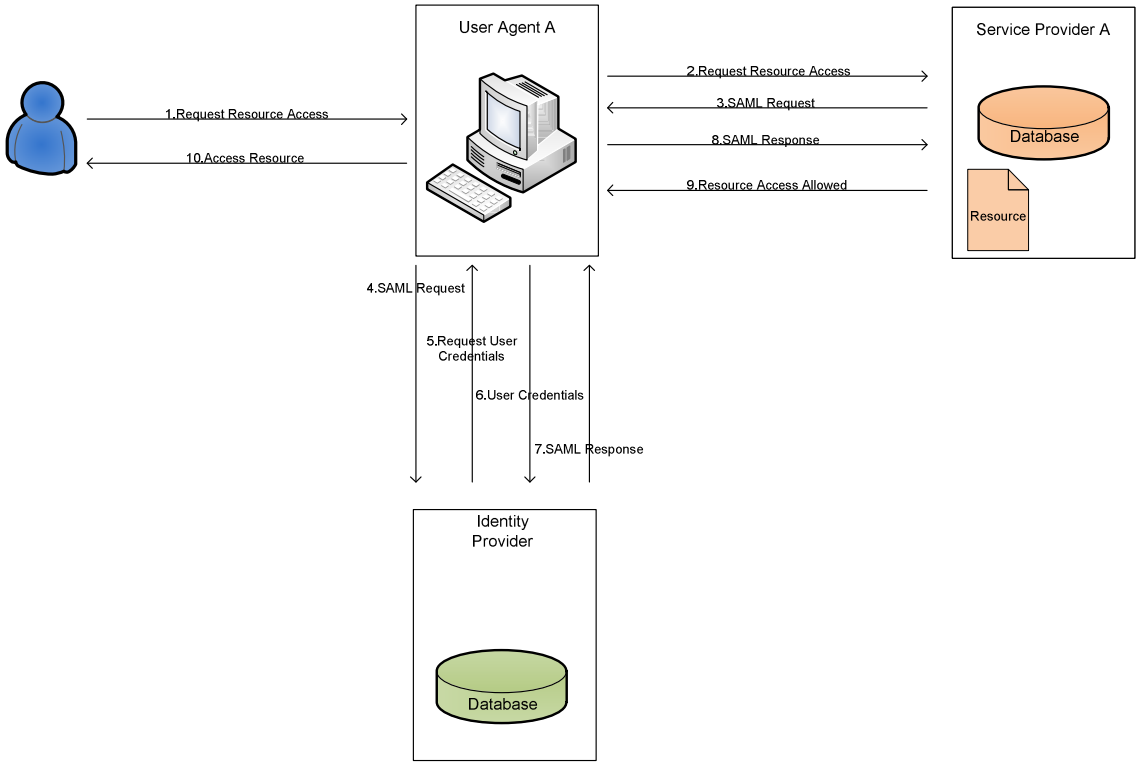


Figure 17: Single Sign On SAML exchange including user authentication actions.

In comparison to Figure 14, steps 1 to 4 are equal. But since there is no valid session at the Service Provider, the user will be redirected to it and asked to sign in. This is accomplished in steps 5 and 6.

The SAML message response will be again equal to the one in the previous section, as well as the following steps.

## Federation

The only difference between the SAML used for federation and the SAML used for simple SSO is that the response will contain user attributes. These attributes will allow

the Service Provider to map the user: for example, they can be used to perform an access check or to create a local session.

```
<samlp:Response xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns:samlp="urn:oasis:names:tc:SAML:2.0:assertion" Version="2.0" ID="abe567de6"
InResponseTo="example-ncname" IssueInstant="2005-01-31T12:00:00Z"
Destination="http://www.sp.example1.com/SSO">
    <saml:Issuer> http://www.idp.example.com </saml:Issuer>
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#”>
        [...]
    </ds:Signature>
    <samlp:Status>
        <samlp:StatusCode
        Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
    </samlp:Status>
    <saml:Assertion
    xmlns:saml=""urn:oasis:names:tc:SAML:2.0:assertion""
    ID=""s277a85ca92692821b3f4cc6b91fa4fed9c084a508""
    IssueInstant=""2009-05-06T18:31:08Z"" Version=""2.0"">
        [...]
        <saml:AttributeStatement>
            <saml:Attribute Name=""email"">
                <saml:AttributeValue
                xmlns:xs=""http://www.w3.org/2001/XMLSchema""
            xmlns:xsi=""http://www.w3.org/2001/XMLSchema-
            instance"" xsi:type=""xs:string"">
                        john@company.com
                </saml:AttributeValue>
            </saml:Attribute>
        </saml:AttributeStatement>
    </saml:Assertion>
</samlp:Response>
```

Figure 18: Single Sign On Response example.

In this case, the attribute “email” with the value “john@company.com” was encapsulated on the response.

## Single Logout

For Single Logout, it will be used the SAML “Single Logout Profile”. In the following Figure the user is visiting Service Provider A when he requests Single Logout (step 1 and 2).

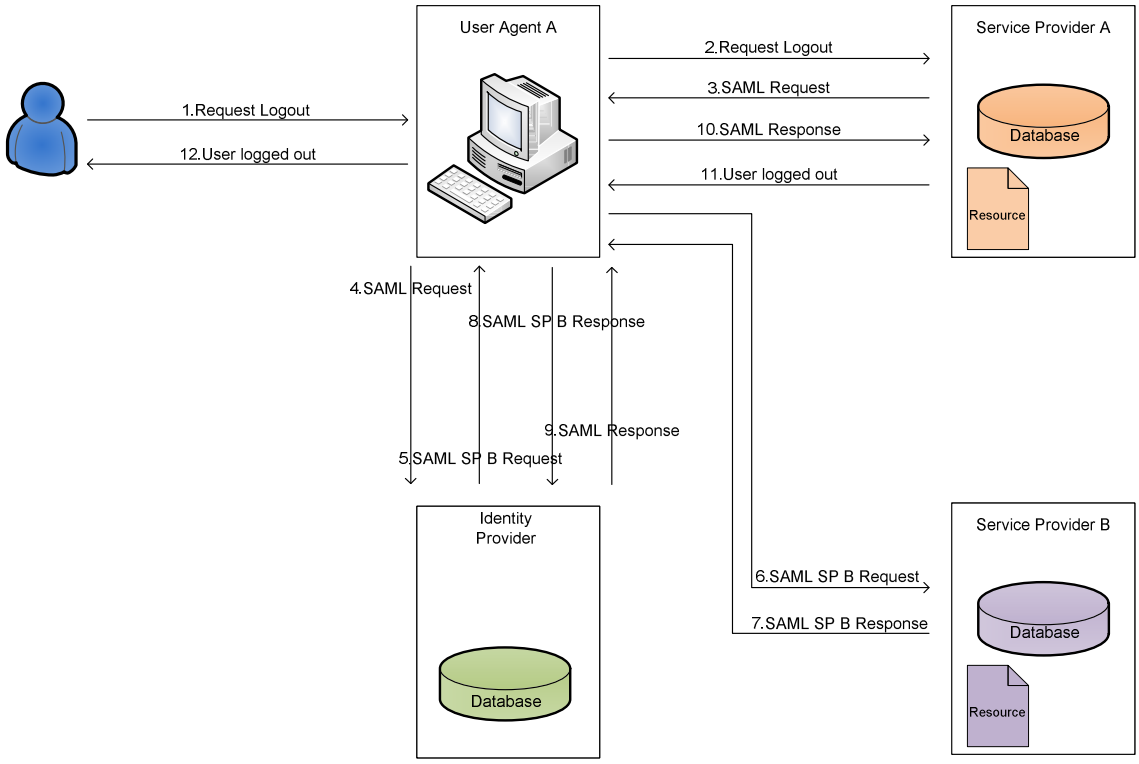


Figure 19: Single Logout SAML exchange.

The procedure is similar to Single Sign On: the Service Provider A will create a SAML Request and will wait for a SAML Response from the Identity Provider (step 3 and 4).

```
<samlp:LogoutRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns:samlp="urn:oasis:names:tc:SAML:2.0:assertion" Version="2.0"
ID="abe567de6" IssueInstant="2005-01-31T12:00:00Z"
Destination="http://www.idp.example1.com/SSO"
Reason="urn:oasis:names:tc:SAML:2.0:logout:user">
      <saml:Issuer> www.acompany.com </saml:Issuer>
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#”>
            [...]
      </ds:Signature>
      <saml:NameID format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient">
            32sHGn/qYH6cdyVlDQG/IpAMACbM
      </saml:NameID>
      <saml:SessionIndex xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol" >
            s2c5da2s23d2534bt53456345789
      </samlSessionIndex>
</samlp:LogoutRequest>
```

Figure 20: Logout Request example.

The request identifies the principal to be logged out (<*saml:NameID*>) as well as a unique session identifier (<*saml:SessionIndex*>).

After verifying that the request came from a known and trusted Service Provider, the Identity Provider will process the request and destroy any local session information about the user. Afterwards it will send a similar <*LogoutRequest*> to Service Provider B (step 5 and 6). Subsequent to receiving the response from the Service Provider B

(step 7 and 8), the Identity Provider will send the response to Service Provider A (step 9 and 10).

```
<samlp:LogoutResponse xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns:samlp="urn:oasis:names:tc:SAML:2.0:assertion" Version="2.0"
ID="abe567de6" InResponseTo="www.acompany.com"
IssueInstant="2005-01-31T12:00:00Z"
Destination="http://www.sp.example1.com/SSO"
Reason="urn:oasis:names:tc:SAML:2.0:logout:user">
      <saml:Issuer> www.idp.example.com </saml:Issuer>
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#”>
            [...]
      </ds:Signature>
      <saml:NameID format="urn:oasis:names:tc:SAML:2.0:nameid-format:emailAddress">
            j.doe@company.com
      </saml:NameID>
      <samlp:Status>
            <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
      </samlp:Status>
</samlp:LogoutResponse>
```

Figure 21: Logout Response example.

It is important to refer that the message sent in steps 5 and 6 is similar to the one in Figure 20 as well as the message in step 7 and 8 to Figure 21.

# 3. Architecture

Having defined the goals and the flow that the solution is to achieve, the systems architecture was defined.

Through modularity and encapsulation, an easily extensible solution was designed that meets all the requirements and respects the constraints, as well as it is easily pluggable to new software.

## 3.1. Relevant Requirements

Most of the architectural requirements mentioned on the following sections are related to the project's goal and play an important role on its success.

### 3.1.1. Functional Requirements

Functional requirements provide the application architecture of the system. The following use case reflects all the different interactions the system must perform.

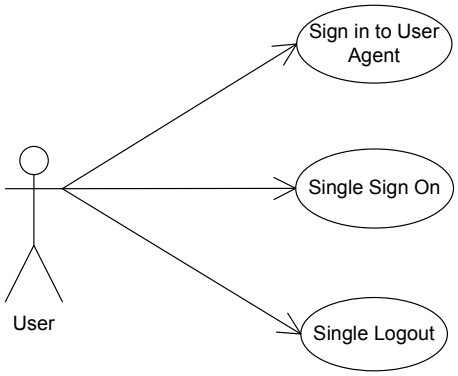


Figure 22: Key use cases.

The main purpose of this implementation is that a user can sign in to a user agent and therefore access protected resources. This user interaction can be decomposed into three scenarios:

- Application and external server: for example, a doctor using ALERT® may wish to access data about a specific patient that is stored in a national server;
- Two ALERT® applications: a user may wish to interact with two different ALERT® applications at the same time;
- ALERT® software and a third party tool: if two distinct applications are running in the same computer, there should be no need to login at both, rather login at one and the second one do an automatically login. An example would be an ALERT® and an NHS application that are running at the same computer where the user is interacting.

All these different user agents are either stand-alone applications or applications that run on a browser.

### 3.1.2. Non-functional Requirements And Constraints

Non-functional requirements drive the technical architecture of the system. The non-functional requirements and constraints for this SAML solution are:

- Interoperability
  - With ALERT® applications;
  - With as many authentication methods as possible, at least with
    - Smartcard;
    - Username and password;
    - Biometrics;
- Platform Compatibility and Portability
  - With the following operating systems
    - Client:
      - Windows (XP / Vista / Others);
      - Linux/Unix;
      - Macintosh, if possible;
    - Identity and Service Provider:
      - Windows (32 and 64 bits);
      - Linux (32 and 64 bits);
      - HP UX;
      - AIX;
  - With as many browsers as possible, at least with
    - Internet Explorer;
    - Firefox;
    - Chrome;
  - With XACML v2.0;
  - It should work on stand-alone applications.
- Security
  - Restrict server access;
  - Provide data security
    - During transfer;
    - While in storage;
- Usability
  - Solution interaction should be clear and elegant;
- Extensibility and Scalability
  - Adding new features or customizing the software should be straightforward;

- It should present a standard API that provides a transparent interface with the current ALERT® applications;
- Connecting multiple hosts and getting them to work as a unit should be possible;

- Documentation
  - Technical documentation;
- Availability and Reliability
  - It should present no single point of failure;
  - Should aim for continuous availability to its users.

## 3.2. Logical View

The following image aims to present a white box description of system, exposing its inner structure.

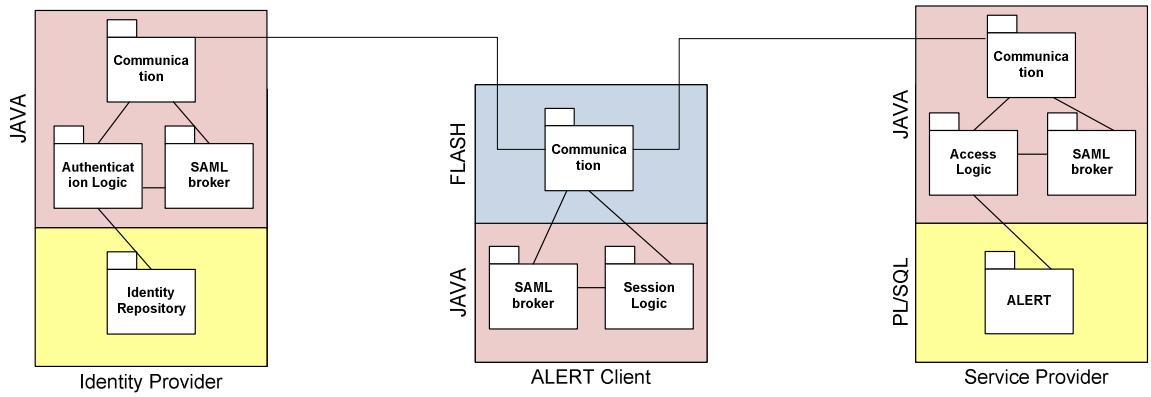


Figure 23: White box description of the SSO system.

The ALERT® Client contains modules for:

- Communication – located in the Flash layer, it is responsible for handling all the in and out coming communications and passing them to the correct package (SAML broker or Session Logic);
- SAML Broker – located in the Java layer, it contains the XML processor for SAML. It can either process the incoming SAML and then pass it to the Session Logic package, or receive data from the referred package and transform into SAML;
- Session Logic – It's responsible for handling the SAML requests and responses, as well as other type of actions.

The Identity Provider SAML broker and Communication module are similar to the ALERT Client modules with the same name. Nevertheless, the Authentication Logic package is responsible to extract the data such user credentials and validate them and to generate responses for the SAML requests.

Again, the SAML broker and Communication module in the Service Provider act alike the ALERT® client modules. The Access logic is responsible for verifying if the user can have access to determined resources thus it is able to generate requests for authentication, authorization and attributes.

## 3.3. Dependency View

Between the elements that compose the system, there are a number of dependencies that have to be taken into account in order for it to function properly.

Since both three elements will have a component for SAML management, there are SAML restrictions that will be shared. Since SAML v1.x and v2.0 are incompatible, all elements must communicate using the same version.

The framework will have to support all SAML versions. Even though it may seem a bit complicated how it can support all versions, it is important to clarify that:

- Between Service Provider A and User Agent A, the SAML version will always be the same, even though it may change to Service Provider B and User Agent B (see Figure 24);
- The Identity Provider will need to know what SAML version every User Agent is using, so he can communicate accordingly.

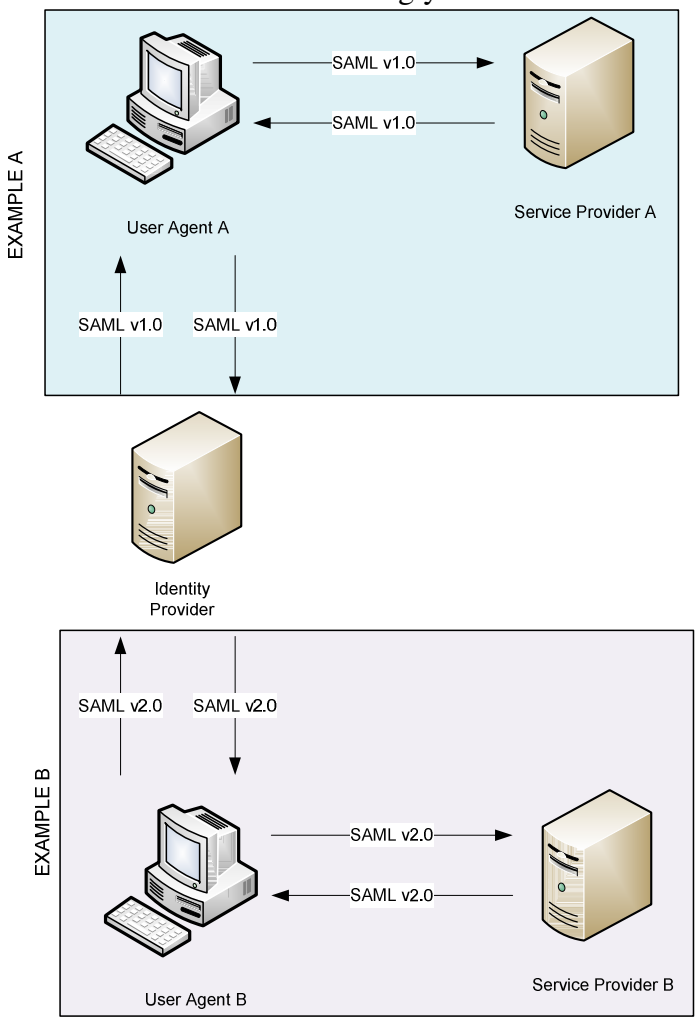


Figure 24: Example of SAML exchange with different SAML versions

Additionally, the system presents more dependencies which are not only related to SAML:

- User agents have to trust IdP – in order for user agents to allow users to authenticate, they need to have a trust relationship with the IdP;
- User agents need to have access to the user information;

- If the user agents are running on a browser, cookies have to be accepted so the user identifier can be stored;

- The User Agent has to support Java and needs to have a browser, as well as being connected to a network (internal or external, depending on the case).
- Authentication framework
  - Responsible for validating the user credentials and returning a subject (carries one or more principals if the authentication was successful)

# 3.4. Behavioral View

In order to have a clear view of the flow between the different software components and how each process interoperates with another one and in what order, sequence and state diagrams are included next.

## 3.4.1. Scenario & Collaboration Models

In order to elucidate a SSO scenario, a sequence model is presented. In this model it is assumed that the user has not yet signed on into any user agent.

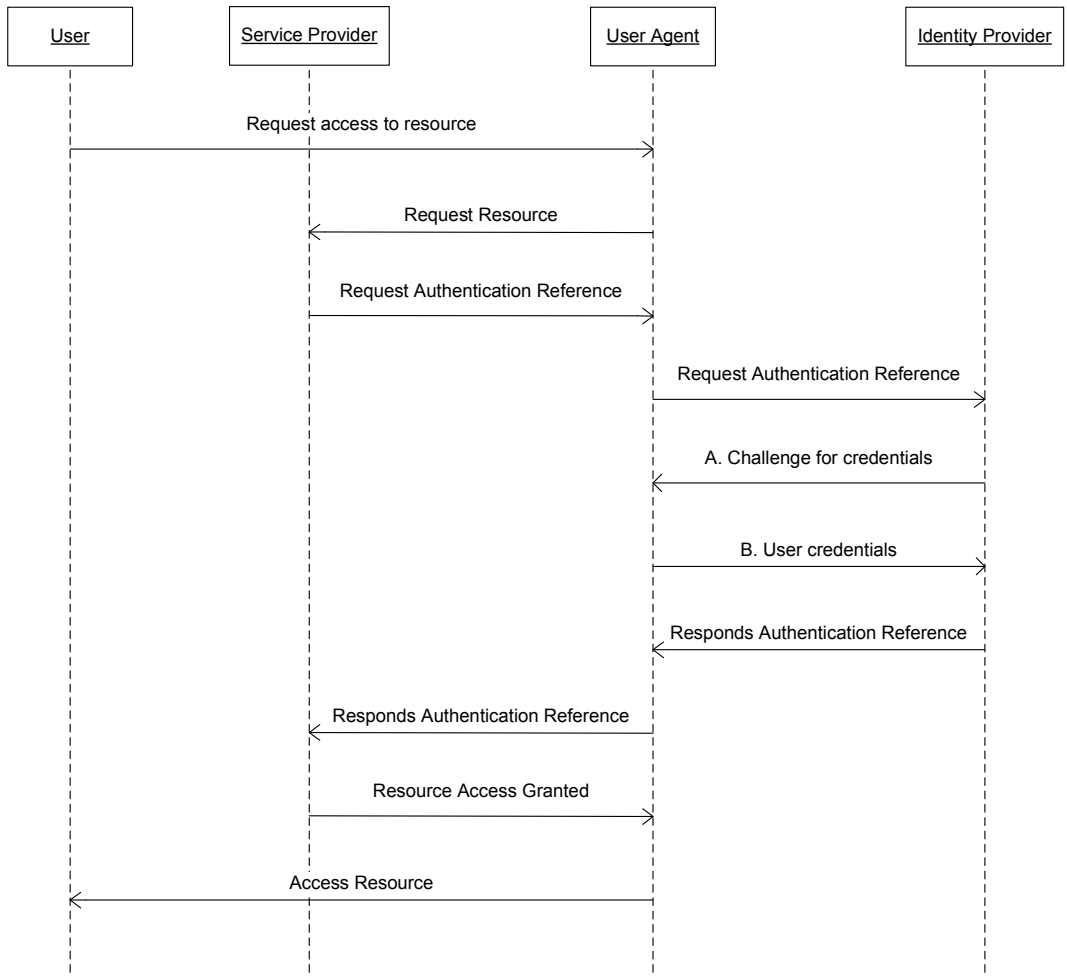


Figure 25: SSO Sequence Model

The sequence model showed on Figure 26 presents Single Logout when it's initiated by the Service Provider. The Identity Provider will be responsible for logging the user out in every system he was authenticated on. As before, Service Provider X aims to represent all Service Providers, others than the one that initiated the logout request.

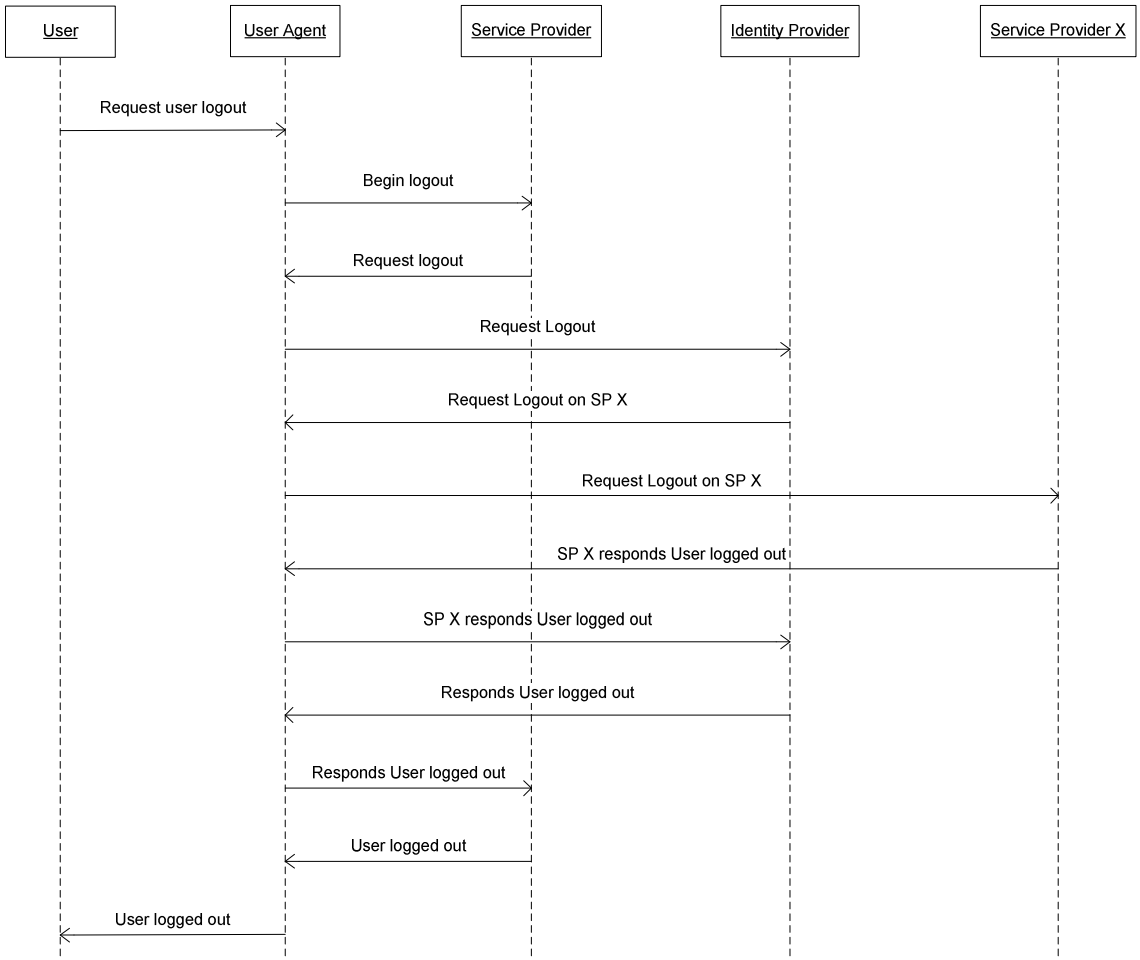


Figure 26: Single Logout sequence model.

## 3.4.2. State Models

The following state model (Figure 27) begins to clarify how SSO will be accomplished. In the following diagram one can understand the different actions that occur whether a simple sign on is occurring (on the right side) where user credentials are needed or if a single sign on operation is taking place (left side).

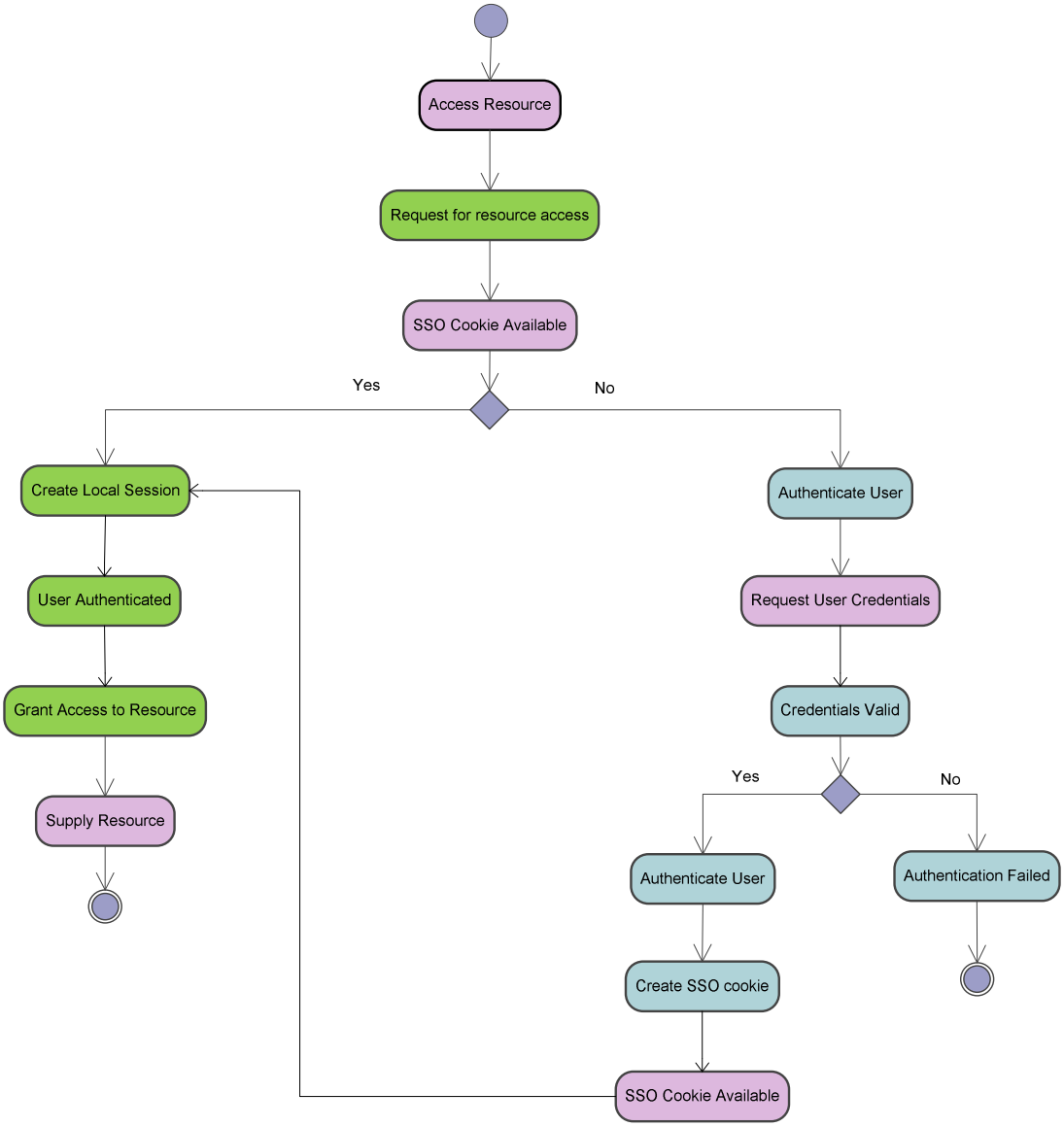


Figure 27: Single Sign On state diagram.

For Figure 27 and Figure 28 the actions that happen in the Identity Provider are marked in blue, for the Service Provider the color is green and for the User Agent is pink.

In Figure 28 the colors used are the same but when a state refers to a Service Provider rather than the one connected to the initial User Agent where the user asked for the logout, it's presented in orange. The Single Logout occurs on the left side, when the Identity Provider takes charge of logging the user off from all other Service Providers. On the right only regular logout actions occur.

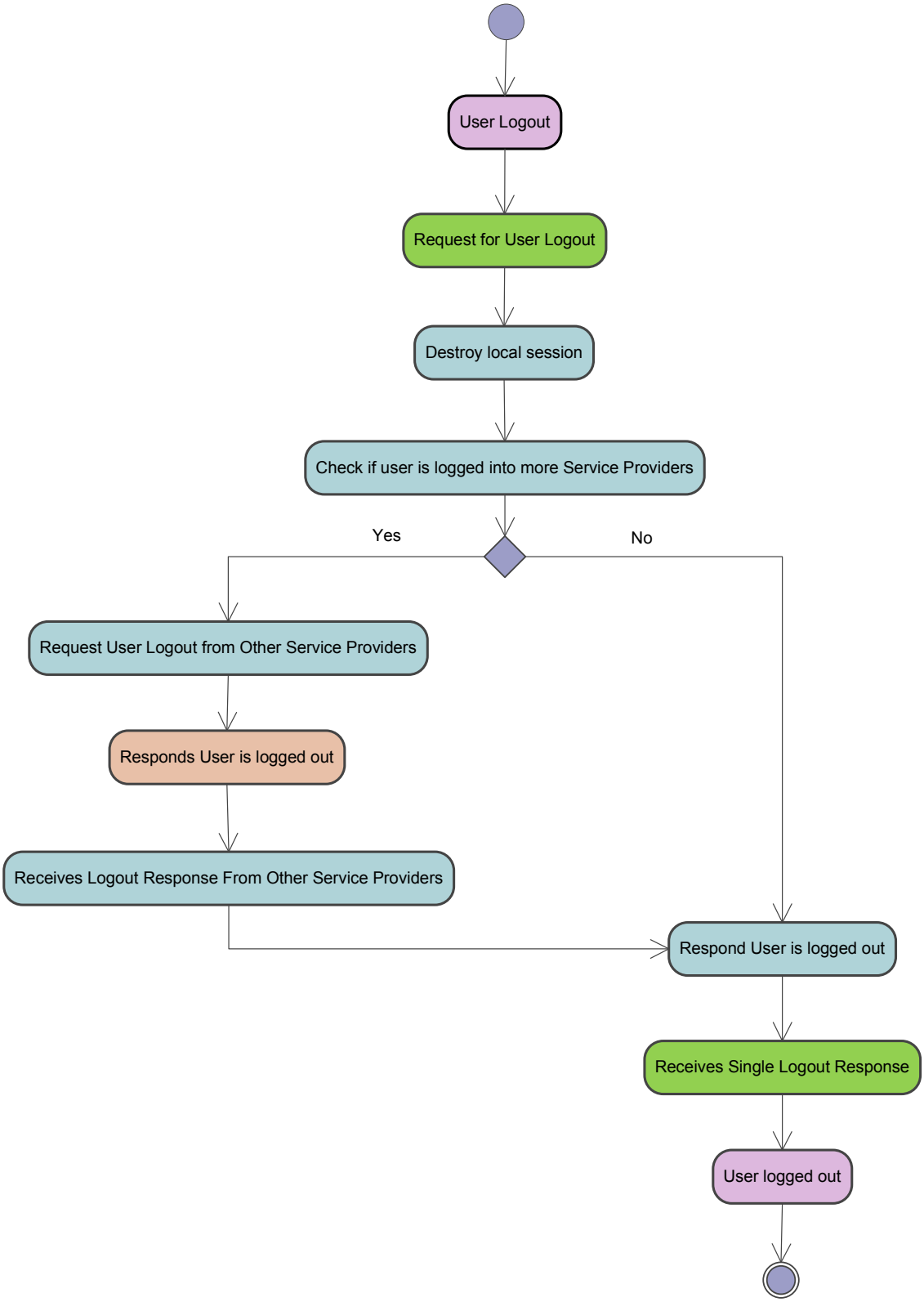


Figure 28: Single Logout state diagram.

## 3.5. Deployment View

The following model describes how functional entities in the logical view are deployed onto implementation entities.

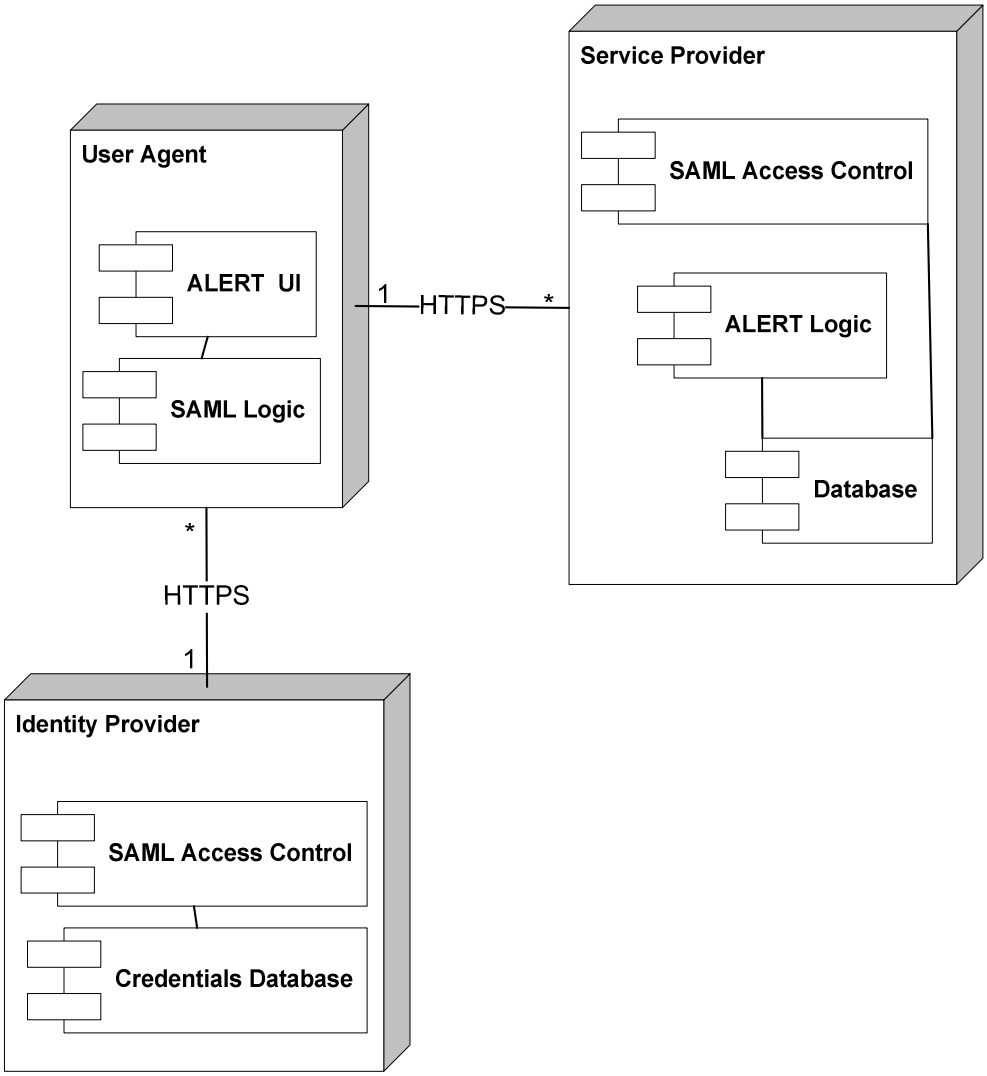


Figure 29: Deployment Diagram

Next four different deployment scenarios will be presented.

- Instantiation example of the components in the case of an ALERT® application interaction with a SPINE Server (Figure 30);

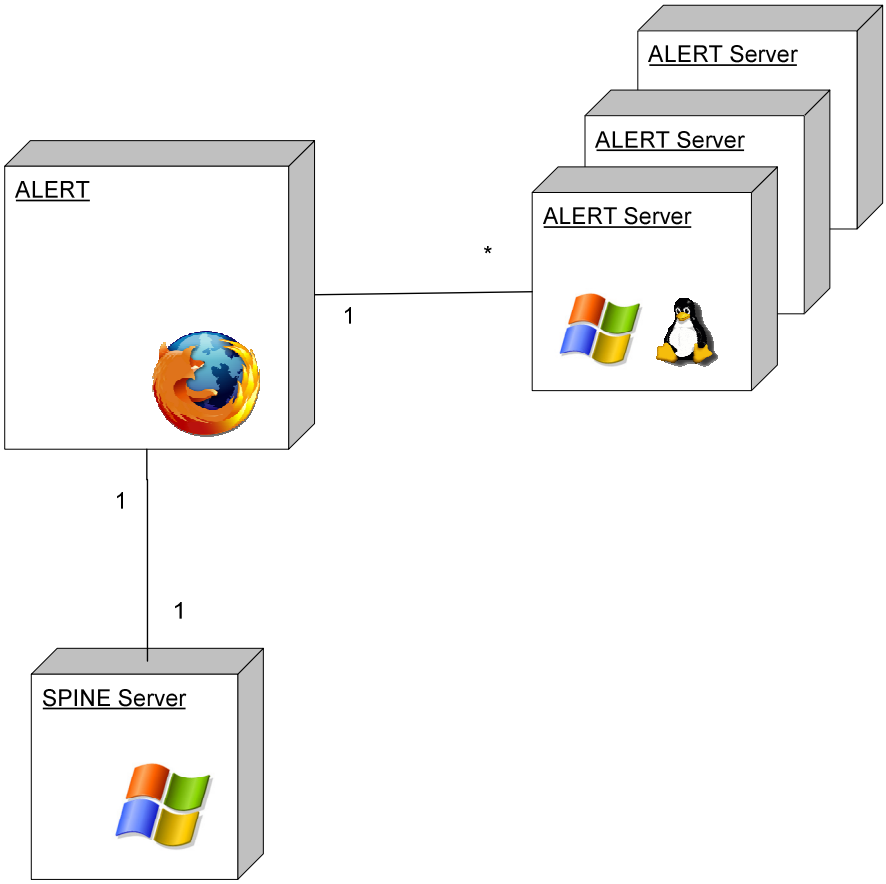


Figure 30: ALERT® application interaction with SPINE Server

- Deployment example on a Hospital (Figure 31);

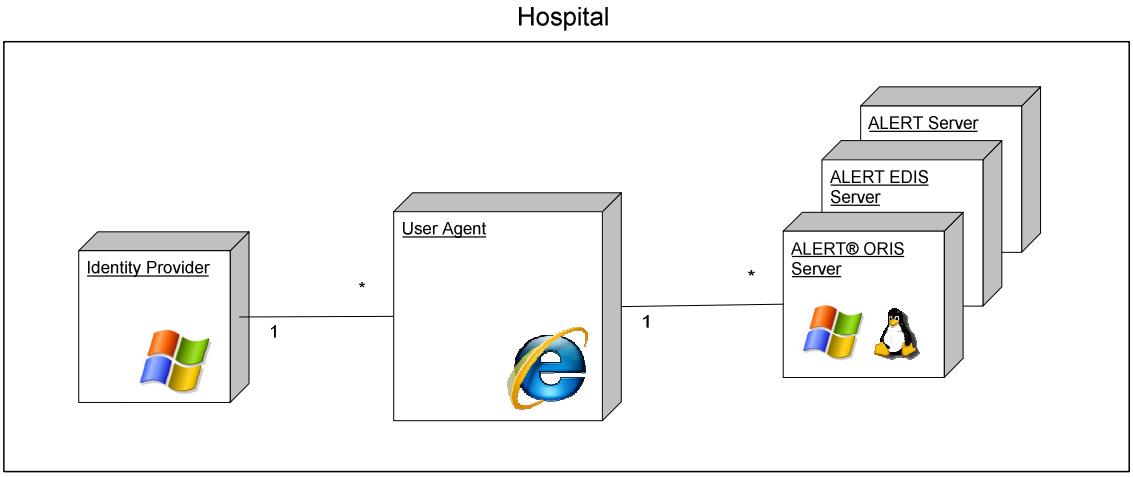


Figure 31: Hospital deployment example.

- Regional Health Information Organization (RHIO) example(Figure 32);

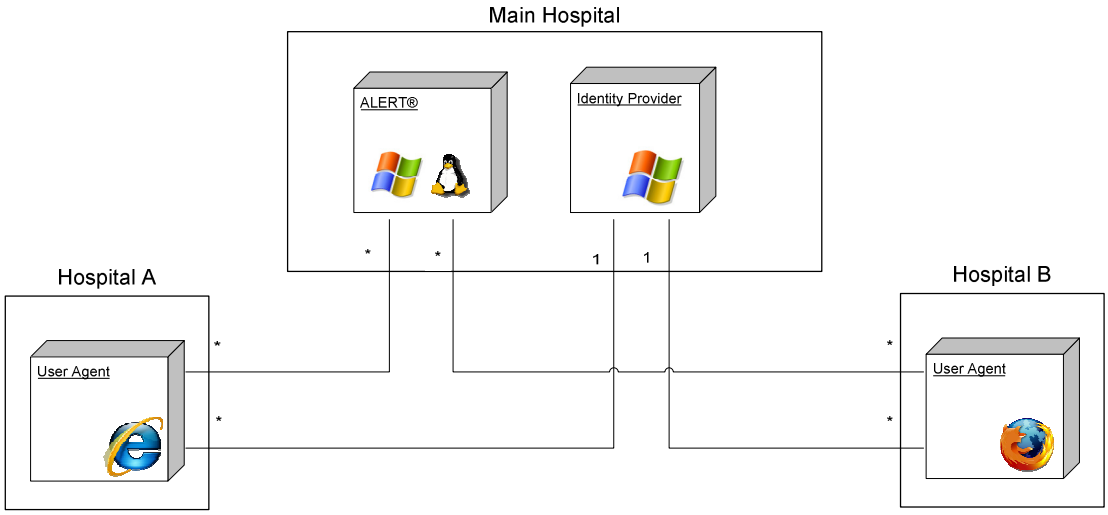


Figure 32: RHIO example.

- Deployment using a datacenter (Figure 33).

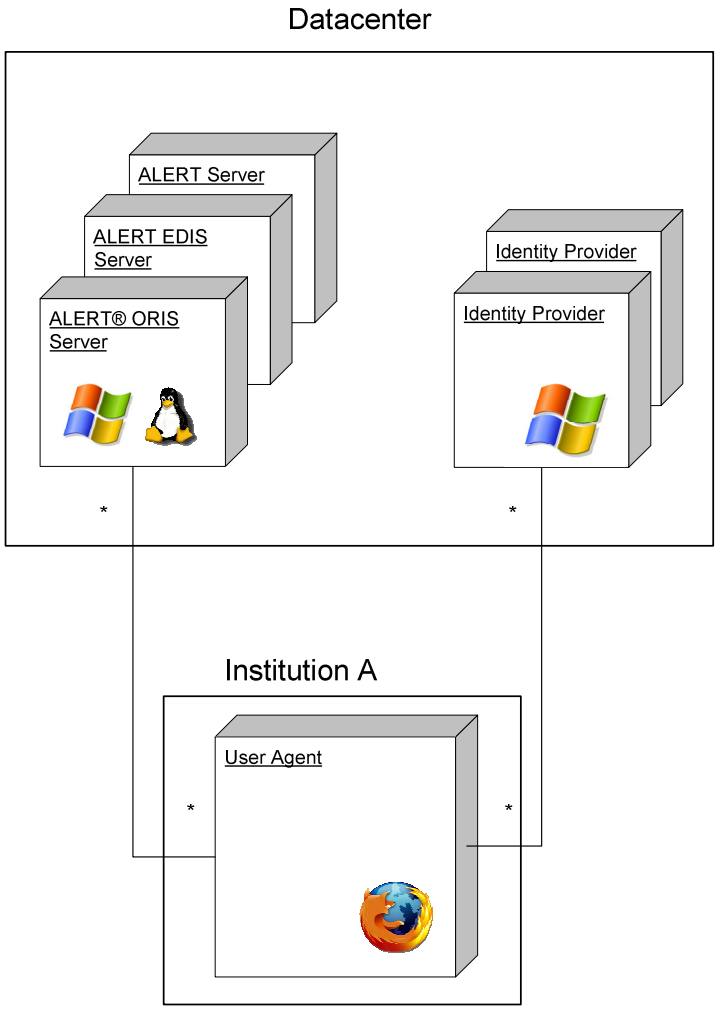


Figure 33: Datacenter deployment example.

# 4. Technology Review

Having discussed the goals for the SSO solution, it was time to evaluate the market in pursuit of a software application that would fit the requirements and help to track down the objectives. An extensive and thorough analysis of the software solutions the market has to offer was conducted.

The aim of this research was to find the best possible solution that would conform to the company's needs. If no available solution fitted the business' requirements, a fully customized solution would have to be developed.

It is important to refer that the technologies that do not use SAML or are not specially designed for healthcare (such as the well known OpenId) were not study objects given that the utilization of SAML is one of the main requirements because of the role it plays on the healthcare industry.

### 4.1.1. Relevant Requirements

One of the goals of organizations that implement SAML is to reduce its deployment time while maintaining the SAML standard's inherent security. Nonetheless, as stated in [Har08], a SAML solution deployment can take weeks or even months and, on such a competitive market as the healthcare industry, this delay is not affordable. Therefore, it was chosen, if possible, to make use of an already available software or library that could help reduce the deployment time.

In order to measure how well the different software solutions adapt to the project needs, metrics were defined to help compare and evaluate the different options available.

These requirements aim to reflect the need for a SAML solution that would perform SSO as well as SLO and is as compatible as possible with all types of systems.

The following metrics were defined:

- Ability to perform Single Sign On - since it is the purpose of the project, the chosen solution has to allow Single Sign On;
- Ability to perform Single Sign on without inter-site redirection – the need for links in each page for other applications would mean the existing legacy systems would have to be redesigned and altered every time a new application needed to support SSO, which is not a valid option;

- Ability to allow/perform Federated Identity – different applications may have different identity providers which means it must be allowed for applications to agree on user tokens that identify the same user defined by different attributes on different service providers;
- Ability to perform Single Logout – it is desirable for users to also be allowed to perform SLO after SSO is achieved;
- Ability to perform authorization requests – the authentication framework will communicate with web applications via SAML, therefore the application has to allow these types of requests;
- Support for SAML – meeting the standards is the best way to ensure interoperability with all applications that ALERT may wish to SSO with;
    - SAML v1.x
    - SAML v2.0
- Support for XML Signature;
- Support for XACML v2.0 – the authentication framework uses XACML thus for communication purposes XACML will be used;
- Support for SOAP – SOAP is the protocol usually used for exchanging XML-based messages over computer networks, normally using HTTP/HTTPS;
- Compatibility with the following authentication methods[2]
    - Smartcard;
    - Password;
    - Biometrics;
- Compatibility on the client side with[2]
    - Windows;
    - Linux/Unix;
    - Macintosh*;
- Compatibility on the server with[2]
    - Windows 64bits;
    - Linux 64bits;
    - HP UX;
    - AIX;
- Browser compatibility[2]
    - Internet Explorer;
    - Firefox;
    - Chrome*.

Fields marked with * are considered to be of low priority in comparison to the others.

### 4.1.2. Available Providers

Two different types of solutions were evaluated: on one hand software that already performed the majority of the actions desired and that should be easy to bundle, and on the other hand, libraries that could be helpful in case the solution had to be built from scratch.

Not all available and evaluated solutions are presented in the following sections since most of them did not meet the minimal requirements.

[2] These are the methods ALERT applications use or are compatible with.

## Software

### *Shibboleth*

Shibboleth [Shi09] is standards based, open source software package for web single sign-on across or within organizational boundaries. It allows sites to make informed authorization decisions for individual access of protected online resources in a privacy-preserving manner.

It implements OASIS standards like SAML in order to provide a federate single sign on and attribute exchange framework.

It is released under the Apache Software License.

### *Ping Federate*

Ping Federate [Pin09] is standalone federated identity management software that delivers secure Internet SSO for all external partner connections while working with the existing identity infrastructure.

Its capabilities include:

- Single Sign On;
- Single Logout;
- Federated Identity System Configuration;
- First and Last Mile Integration;
- External Data Look-Up;
- External Connections;
- SaaS Provisioning;
- SaaS Connectors;
- Auto-Connect™;
- Certificate Management.

It is based on several standards such as SAML and WS-Federation.

### *simpleSAMLphp*

SimpleSAMLphp [FEI091] is an application written in native PHP that deals with authentication. It supports several federation protocols, authentication mechanisms and can be used for local authentication, as a Service or as an Identity provider.

It is compliant with standard protocols such as SAML 2.0.

Even though it was written in PHP it also supports non-PHP environment by using the *Auth Memcookie* approach, but it is not fully documented.

### *OneSign*

OneSign [Imp09] is an application developed by Imprivata to address industry compliance initiatives, to increase user productivity and to reduce password management costs.

It is specially designed for the healthcare industry thus it is used in numerous hospitals and healthcare centers. It provides Single Sign On and it also deals with authorization issues.

The main difference from the other referred softwares is that it works as a desktop solution.

### *OpenSSO*

Sun OpenSSO Enterprise [jav09] is the single solution for Web access management, federation, and Web services security. OpenSSO, the open source access management and federation server platform, is the core of OpenSSO Enterprise.

Its goal is to provide an extensible foundation for an identity services infrastructure in the public domain, facilitating Single Sign On for web applications hosted on web and application servers.

OpenSSO was developed in Java and comprises the following modules on a free right-to-use basis:

- Session management;
- Policy;
- Console;
- Administration tools;
- Federation;
- Web services;
- Policy agents.

## Libraries

### *Lasso*

Liberty Alliance Single Sign On (LASSO) [Las09] is a software C library that aims to implement the Liberty Alliance standards, under the GNU General Public License. A commercial license is also available.

It has successfully taken Liberty Alliance interoperability tests and implemented Id-FF, ID-WSF and SAML 2.0.

It secures the access to applications as well as simplifies it by using the Single Sign On technology. It is built on top of libxml2, XMLSec and OpenSSL. At the moment Python, Perl, Java and PHP bindings are tested and distributed.

### *OpenSAML*

OpenSAML [Int09] is an open-source toolkit, in Java and C++, produced by Internet2 developers as a part of their work on the Shibboleth project.

It provides core message, binding, and profile classes for implementing applications based on SAML. It is able to create objects with the individual information fields that make up a SAML message, build the correct SAML representation and parse the SAML back into object form.

Currently, OpenSAML version 2.0 is in development which will support SAML 1.x and 2.0 and is provided under Apache 2.0 license.

### 4.1.3. Software Comparison

## Simple evaluation

Before proceeding to evaluating all the referred software providers, a first basic assessment was conducted for the five different software options.

Shibboleth, Ping Federate and OpenSSO seemed to present themselves as probably good solutions so the evaluation was to continue, which did not happen with simpleSAMLphp and OneSign.

### *simpleSAMLphp*

Even though it provided the basic requirements, the fact that it was written in PHP allowing just the *AuthMem cookie* to be used in order to allow Java, restricted possible customizations of the software.

Integrating software built in PHP into an ALERT client where Flash and Java layers coexist was considered too much of a mix when performance and scalability were taken into account.

It was also taken into consideration that simpleSAMLphp was a solution designed for academic environments. This made it unsuitable to function in a professional environment since it could not provide all the security and requirements.

### *OneSign*

The fact that OneSign is used by so many hospitals and healthcare centers seemed to be a good starting point. Nevertheless it was also decided that this would not be a good option for ALERT to consider:

- It is offered as a desktop solution:
  - Therefore it will take time to load before it can be used and it needs to be configured by each user;
  - ALERT became aware of healthcare facilities that had used a desktop solution and they measured the time it took for it to get into action being in average about 3 minutes;
  - Since time is critical in healthcare environments, solutions that are proven to be time-consuming cannot be accepted;
- It does not comply with SAML and therefore to SSO standards, which would make the creation of identity federations complex.

Consequently OneSign was excluded from the following steps in the software providers' evaluation.

## Requirements comparison

The fields marked with ✓ mean that the requirement is met. It should also be mentioned that the lack of ✓ on the field can either mean that the action is not supported (by default or at all) or that no specific tests to evaluate it have been conducted.

When it comes to the actions supported, OpenSSO comes out as the winner, supporting all actions that are required by ALERT. Shibboleth fails to implement Single Logout which is a major setback.

Table 2: Supported Actions

| | SSO | SSO without Inter-site redirection | Federated Identity | Single Logout | Authorization Requests |
|---|---|---|---|---|---|
| Ping Federate | ✓ | ✓ | ✓ | ✓ | |
| OpenSSO | ✓ | ✓ | ✓ | ✓ | ✓ |
| Shibboleth | ✓ | | ✓ | | ✓ |

As seen before, since PingFederate does not support Authorization Requests, it does not support XACML. Other than that, all three solutions support the required XML standards.

Table 3: XML Standards

| | SAML v1.x | SAML v2.0 | XACML v2.0 | XML Signature | SOAP |
|---|---|---|---|---|---|
| Ping Federate | ✓ | ✓ | | ✓ | ✓ |
| OpenSSO | ✓ | ✓ | ✓ | ✓ | ✓ |
| Shibboleth | ✓ | ✓ | ✓ | ✓ | ✓ |

ALERT supports a wide range of authentication methods. Unfortunately, when using default configurations, only PingFederate supports all three required authentication methods.

Table 4: Authentication Methods

| | Smartcard | Password | Biometrics |
|---|---|---|---|
| Ping Federate | ✓ | ✓ | ✓ |
| OpenSSO | ✓ | ✓ | |
| Shibboleth | | ✓ | |

On server interoperability matters, the only difference between the three providers is that OpenSSO also supports HP UX.

Table 5: Operating Systems - Server

| | Windows 32bits | Linux 32bits | Windows 64bits | Linux 64bits | HP UX | AIX |
|---|---|---|---|---|---|---|
| Ping Federate | ✓ | ✓ | ✓ | ✓ | | |
| OpenSSO | ✓ | ✓ | ✓ | ✓ | ✓ | |
| Shibboleth | ✓ | ✓ | ✓ | ✓ | | |

No software solution specifies if they can interoperate with Macintosh.

Table 6: Operating Systems - Client

| | Windows | Linux | Macintosh |
|---|---|---|---|
| Ping Federate | ✓ | ✓ | |
| OpenSSO | ✓ | ✓ | |
| Shibboleth | ✓ | ✓ | |

Again, no software solution specifies whether or not they can be executed on Chrome.

Table 7: Browser

| | Internet Explorer | Firefox | Chrome |
|---|---|---|---|
| Ping Federate | ✓ | ✓ | |
| OpenSSO | ✓ | ✓ | |
| Shibboleth | ✓ | ✓ | |

### 4.1.4. Software Evaluation

A more detailed evaluation was conducted therefore it was chosen to test more carefully the three providers in order to evaluate their implementation and possible configuration and customization.

#### Ping Federate

Ping Federate fails to implement XACML and authorization requests, therefore failing critical requirements. Nevertheless it proved to be simple software to install and test, when tested in a computer which is not connected to a local network since it assumed "*localhost*" for all configurations. This configuration fails to function in ALERT computers due to its internal network definitions, so it was deployed on a personal computer.

It provided sample applications for the Identity and Service Provider which were used and tested.

Ping Identity was contacted via e-mail in order to clarify some developer questions and their support was effective and fast.

Ping Federate is a commercial solution which is not cost free.

#### OpenSSO

OpenSSO is the software that complies with most of the requirements, making it eligible as a good possible solution.

Innumerous issues occurred when trying to test and install OpenSSO: the tutorials were not clear and setting up the environment was a complicated task. There were no clear instructions about how to use Apache Server for creating virtual hosts, how to link them to Apache Tomcat and most importantly, how to use them on OpenSSO.

Since this approach was unsuccessful, it was time to attempt to deploy OpenSSO on Glassfish since it is also developed by Sun Microsystems. Even though it was also not simple, it turned to be effective in the end.

Afterwards, it was time to follow the tutorials and evaluate the solution. Again, problems occurred with the guides, as well as with the deployment of new components of OpenSSO (such as the OpenSSO SDK).

All problems fixed and, after getting familiar with the software and understanding its flow, it was possible to evaluate OpenSSO.

The documentation is very extensive and offers various OpenSSO deployment scenarios. Given that ALERT's goals are complex, there will be the need to customize OpenSSO and also to use different scenarios at the same time. Since it was written in Java, it can be adapted through new Java modules and extensions.

OpenSSO offers support at an IRC Channel, where OpenSSO developers offer their expertise to help others solving their problems.

As already mentioned it is an open source solution and it is free.

### Shibboleth

Shibboleth also complies to most of the requirements, unfortunately trying to deploy an Identity Provider as well as a Service Provider at the same time with Shibboleth proved to be an impossible solution since the learning curve is very steep.

Even after receiving help from the Shibboleth mailing-list (official support), this task could not be accomplished.

When trying to deploy an Identity and a Service Provider at the same time, it is especially hard to understand how Shibboleth works. Problems arose in different areas (with certificates, metadata, federation, etc) and after solving an issue, a new one would emerge.

Shibboleth experts suggested to try to deploy just one part (Identity or Service Provider) and test it against Test Shib [Int091]. After being sure that one of the parts was working, deploying the other one to match it should be easier.

So it was decided to try to deploy just the Service Provider and test it. This proved not possible because there was the need to have an external IP address, and this scenario is not allowed in ALERT.

In the end, no valid Shibboleth testing was accomplished.

## 4.1.5. Libraries Comparison

In what concerns the libraries, the initial research showed no reason to exclude any, so both of them were evaluated.

### Requirements comparison

Since libraries were now being evaluated, some requirements did not apply. In these cases, the term N/A (Non Applicable) was used.

Once again, the fields marked with ✓ mean that the requirement is met. The fields that do not present any type of symbol mean that either there is no official information about the subject or that it is not met.

Only Lasso presented information about what could be accomplished with its usage so there is no data regarding OpenSAML to compare with.

Table 8: Supported Actions

| | SSO | SSO without Inter-site redirection | Federated Identity | Single Logout | Authorization Requests |
|---|---|---|---|---|---|
| Lasso | ✓ | | ✓ | ✓ | |
| OpenSAML | | | | | |

Both libraries fail to implement XACML but OpenSAML already supports SOAP.

Table 9: XML Standards

| | SAML v1.x | SAML v2.0 | XACML v2.0 | XML Signature | SOAP |
|---|---|---|---|---|---|
| Lasso | ✓ | ✓ | | ✓ | |
| OpenSAML | ✓ | ✓ | | ✓ | ✓ |

Since the libraries are only used to create and analyze SAML, it is not of their domain to support authentication methods.

Table 10: Authentication Methods

| | Smartcard | Password | Biometrics |
|---|---|---|---|
| Lasso | N/A | N/A | N/A |
| OpenSAML | N/A | N/A | N/A |

The behavior of both libraries when it comes to supporting different operating systems is identical.

Table 11: Operating Systems - Server

| | Windows 32bits | Linux 32bits | Windows 64bits | Linux 64bits | HP UX | AIX |
|---|---|---|---|---|---|---|
| Lasso | ✓ | ✓ | ✓ | ✓ | | |
| OpenSAML | ✓ | ✓ | ✓ | ✓ | | |

Again, they both support the same client operating systems. They both state that they support Macintosh, which all the software providers failed to achieve.

Table 12: Operating Systems - Client

| | Windows | Linux | Macintosh |
|---|---|---|---|
| Lasso | ✓ | ✓ | ✓ |
| OpenSAML | ✓ | ✓ | ✓ |

Such as for the authentication methods, browser support is not in the scope of these libraries. This does not mean that they do not work correctly on the following browsers, but that it will depend on how they are used on the SSO implementation.

Table 13: Browser

| | Internet Explorer | Firefox | Chrome |
|---|---|---|---|
| Lasso | N/A | N/A | N/A |
| OpenSAML | N/A | N/A | N/A |

Since ALERT® middle-tier uses Java, it was important to know whether or not they were written in Java.

Table 14: Java

| | Java |
|---|---|
| Lasso | |
| OpenSAML | ✓ |

### 4.1.6. Libraries Evaluation

The usage of SAML libraries was not tested. Nevertheless they were analyzed and evaluated.

Both libraries are cost free.

#### Lasso

Lasso is the library that meets most of the requirements but still falls short of minimum requirements.

Also, after an analysis and debate with Lasso experts it was discovered that it does not deal with XACML, and that the Lasso XML objects are not converted into Java DOM objects. When asked about Identity Providers mentioned that they had never written an Identity Provider in Java since usually the Identity Provider and much of the Service Providers are written in Python.

#### OpenSAML

OpenSAML does not by itself implement full SAML profiles such as single sign-on, but can be used to simplify the implementation of such profiles. Therefore, an extension to OpenSAML had to be performed in order for it to be used when creating Identity and Service Providers.

It appeared to be a good starting point, yet extensive programming would have to be done in order for it to be usable.

### 4.1.7. Decision

There was not a solution that comprised all the requirements defined at the beginning. The ALERT goals and aims for the project are very wide and hard to accomplish within a single solution.

A simple and effective SSO and SLO platform that complied with SAML standards was the main goal. Its interoperability and compatibility was also an important aspect.

The biggest challenge to overcome was measuring the value of each solution and whether the requirements that were not met were critical[3] or not. Nevertheless, the requirements defined do not overview all important aspects to take into account when choosing a software solution: its cost, support, documentation, scalability and simplicity of usage must also to be considered.

Making it all even more complex, the deployment of the three software solutions proved to be a thorny task that presented innumerous problems, taking longer than expected.

The final choice proved to be a very elaborate and hard decision to take due to the inexistence of the perfect solution. A summary table is presented next.

Table 15: Summary table of the software comparison.

| | SSO | Single Logout | SAML | XACML | Windows | Linux |
|---|---|---|---|---|---|---|
| Ping Federate | ✓ | ✓ | ✓ | | ✓ | ✓ |
| OpenSSO | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Shibboleth | ✓ | | ✓ | ✓ | ✓ | ✓ |

The major differences reside in the support for Single Logout and XACML, but also on the costs.

Ping Federate was excluded since it did not support XACML. Even though it is a simple and commercial solution, consequently integration and support would not be a complicated task, the fact that it is not cost free and that it does not support XACML made it an invalid option.

It was discovered that Shibboleth was used in academic environments and, even though it met most of the requirements, the support proved to be insufficient to allow an inexperienced user to install and test it.

When it comes to the libraries, they are both in an embrionary state which means that adopting them could become as difficult and complex as writing the whole software from scratch (see table below).

Table 16: Summary table of the library comparison.

| | SSO | Single Logout | SAML | XACML | Windows | Linux | Java |
|---|---|---|---|---|---|---|---|
| Lasso | ✓ | ✓ | ✓ | | ✓ | ✓ | |
| OpenSAML | | | ✓ | | ✓ | ✓ | ✓ |

[3] A requirement is considered to be critical if its non-implementation compromises the project's success.

OpenSAML lacks the implementation of the SAML 2.0 profiles for Single Sign On and Single Logout and none of them implement XACML. Therefore they had to be extended before they could be effectively used.

Thus OpenSSO was deemed to be the best existing solution for the project. It provides extensive and clear documentation, good support and it meets the most important requirements. Since it was developed by Sun Microsystems, a level of trust can be associated with the product and the fact that it is free adds up to the final decision.

## Prototype

Having decided that OpenSSO was the software to be used for the Single Sign On solution, more extensive and complex tests were made in order to be sure of its adaptability to ALERT® software.

During the technology research phase, some questions arose relative to the OpenSSO flow that needed to be answered. Those questions were:

- OpenSSO user data store has to be either Sun Directory Server or a supported LDAPv3 compliant directory server: can this restriction be overcome?
- Is OpenSSO integration with ALERT® possible respecting ALERT®'s software architecture and flow?
- Can OpenSSO be integrated with ALERT® according to SAML standards?
- Are there any downsides to the usage of OpenSSO?

It was chosen to implement a prototype whose flow was simple but still allowed to answer the above questions. It was written in J2EE because the final solution will be written in Java but also because the prototype must include web interaction, which can be easily done on J2EE.

For local deployment, Glassfish was user as he application server. For each OpenSSO instance and application, a domain was created. An overview of the prototype will be presented.

### *Important details*

Before proceeding to the prototype details there are some concepts that have to be clarified:

- SSO token – OpenSSO creates a token that represents a Single Sign On token. It contains token related information such as the authentication method but also session related information such as the maximum session time. It can also include other properties, such as specific user attributes;
- Metadata – refers to the descriptive information embedded inside a file. It contains information about the providers;
- Cookie – a cookie is automatically created by OpenSSO entitled "*iPlanetDirectoryPro*" that contains the SSO token (more information about the cookie is available on Appendix C);
- OpenSSO SDK – it includes client-side Java classes and configuration properties and it can be used to write web applications that access an OpenSSO Enterprise server.

### *User Interaction*

The user starts by accessing the Identity Provider (www.idp.com). He wishes to login so provides its credentials. After successful login, a page is displayed with information about the SSO token that was created.

Afterwards the user accesses a Service Provider and opens a new tab for www.sp.com. He clicks a button that invokes Single Sign On and is redirected to a restricted area on the Service Provider. This area presents a button for Single Logout that, when clicked, automatically signs off the user from the Identity as well as from the Service Provider.

### *Implementation issues*

First, it will be examined the deployment on the Identity Provider side.

OpenSSO is usually used with its login form, meaning, users are usually redirected to OpenSSO to login. This presented a problem because integration had to be transparent and provide the same look and feel throughout the whole process so this could not occur.

After experimenting and analyzing the OpenSSO SDK it was discovered that it could be written a custom way to sign on users to OpenSSO using nothing but a simple command line. The first obstacle had been overcome.

Subsequent to successfully passing the first obstacle, it was time to figure out a solution for how to delete OpenSSO restriction from only being compatible with LDAP v3 compliant servers. Research was done and it was learnt that custom authentication modules could be written. Authentication custom modules must include:

- a class that extends *AMLoginModule*, an abstract class that implements JAAS *LoginModule* that provides methods to access OpenSSO services;
- and the module XML configuration.

One of the methods that has to be overridden is *process()* that is called to authenticate a subject. This method can potentially connect to another type of data store or through RMI invoke an authentication framework that validates the user. The LDAP restriction was overcome.

On the Service Provider, the deployment is simpler. Since OpenSSO already provides simple interfaces for Service Provider initiated Single Sign On as well as Single Logout, there was no need for extensive developments.

Since federation is needed and there is no real linkage between OpenSSO instances and the data stores, the implementation had once again to be customized. It was chosen to use a Service Provider Adapter that includes methods that can be extended to perform user specific logic during SAMLv2 protocol.

### *Evaluation*

After the prototype was developed and tested it was concluded that it served its purpose of answering the initially posed questions:

- the dependency of a supported LDAP v3 compliant directory server was eliminated, which could have presented the major obstacle to OpenSSO implementation;
- integrating OpenSSO with ALERT® can be done respecting ALERT®'s flow and layers;

- through OpenSSO configurations, SSO can be achieved respecting SAML standards while dealing with OpenSSO restrictions and customizations;
- OpenSSO has not shown any downside to its usage.

The prototype was a success since it overcame all the difficulties and answered all questions. Nonetheless one challenge had yet to be overcome: the deployment of OpenSSO on Apache Tomcat and Apache HTTP servers.

Even after understanding the OpenSSO flow, deploying it on Apache servers was a complicated task since there are no tutorials available for this deployment option and it is considered a complex task by the OpenSSO team. However, because OpenSSO was better understood at this time the challenge was overcome. It was chosen to deploy the same prototype on Tomcat to test if the behavior was equal to the one in Glassfish, which proved to be the same as expected.

# 5. Proof of Concept

In order to demonstrate the feasibility of the solution, a proof of concept (POC) was developed. This POC aims to demonstrate that this innovative approach is viable, feasible and capable of solving the Single Sign On problem.

For this POC, two existing applications where used: the ALERT® Online and MyALERT®. It was chosen only to use SAML v2.0.

It was decided to use these two applications because they are online applications that offer a single point of entrance, presenting separated flows.

## 5.1. ALERT® Online

ALERT® Online (AOL) is the institutional website. AOL presents information about the company and its products, as well as the latest news related to ALERT. It provides the ability to purchase ALERT ® products as well as to access them.

## 5.2. MyALERT®

A Personal Health Record (PHR) that stores all the essential health information on an online record. A PHR is initiated and maintained by an individual and its main goal is to provide a complete and accurate summary of the individual's health care history. An example would be when the individual changes doctor, in case the individual decides to share its PHR, the new doctor will easily become acquainted of medical history and its details. MyALERT® is the PHR solution offered by ALERT.

## 5.3. User Interaction

In order to clarify the experience and interaction the user will face, it was chosen to include a diagram of the screens the user will see as well as the processing that will occur on the background on AOL and MyALERT® (Figure 34 to Figure 37). It is similar to a high-level state diagram.

The color purple refers to the Service Provider, in this case, MyALERT® and blue refers to the Identity Provider, AOL. The orange interactions indicate user's or redirection actions.

## Scenario

A user wishes to access the MyALERT® but is not yet logged in. The user will be redirected to ALERT® Online to login, and after successful authentication he will be redirected to MyALERT® and access it.

## Providers

- ALERT® Online: www.alert.com
- MyALERT®: www.phr.com

## Diagram

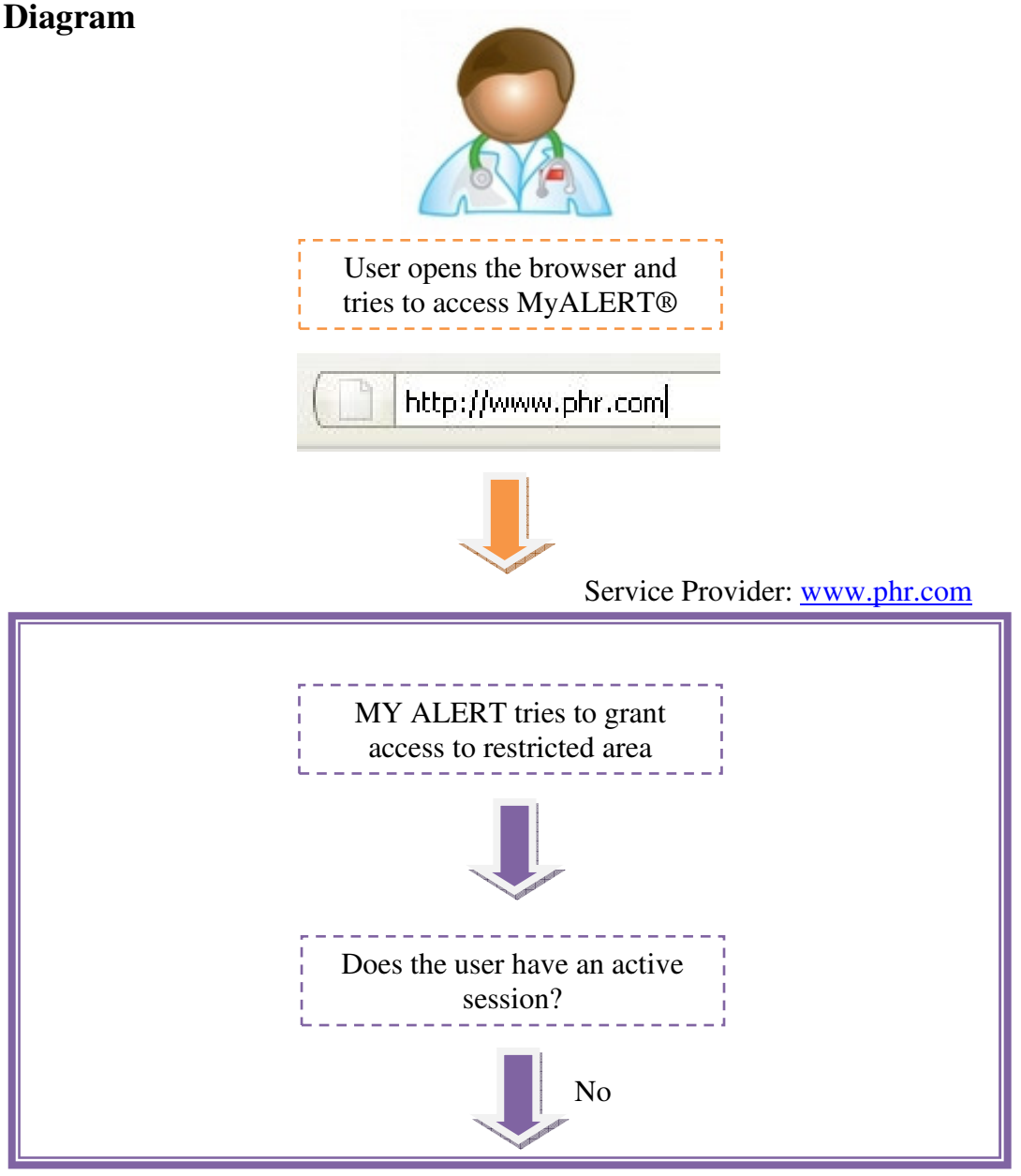


Figure 34: Example interaction on AOL and MyALERT® (Part I).

Service Provider: www.phr.com

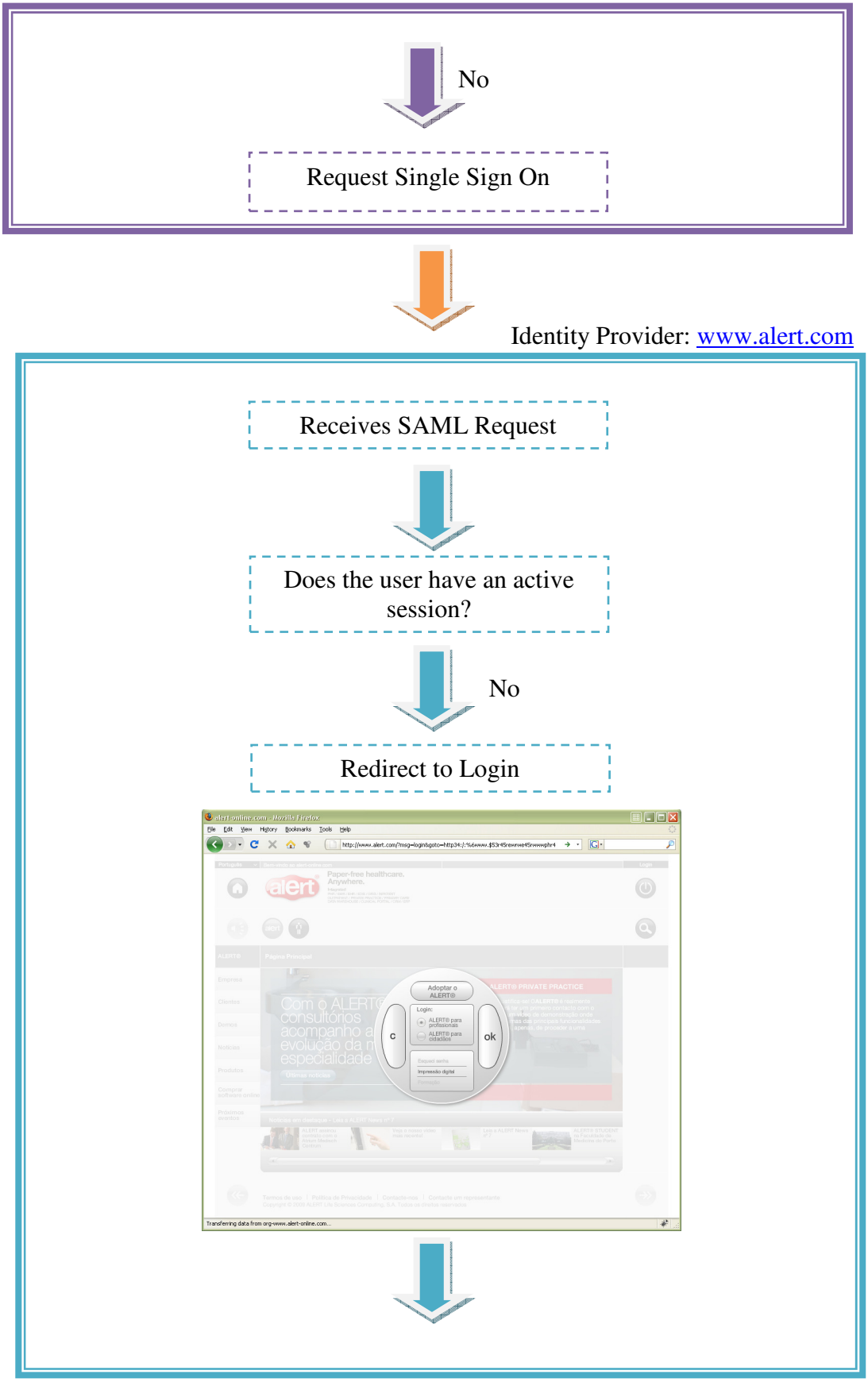


Figure 35: Example interaction on AOL and MyALERT® (Part II).

Identity Provider: www.alert.com

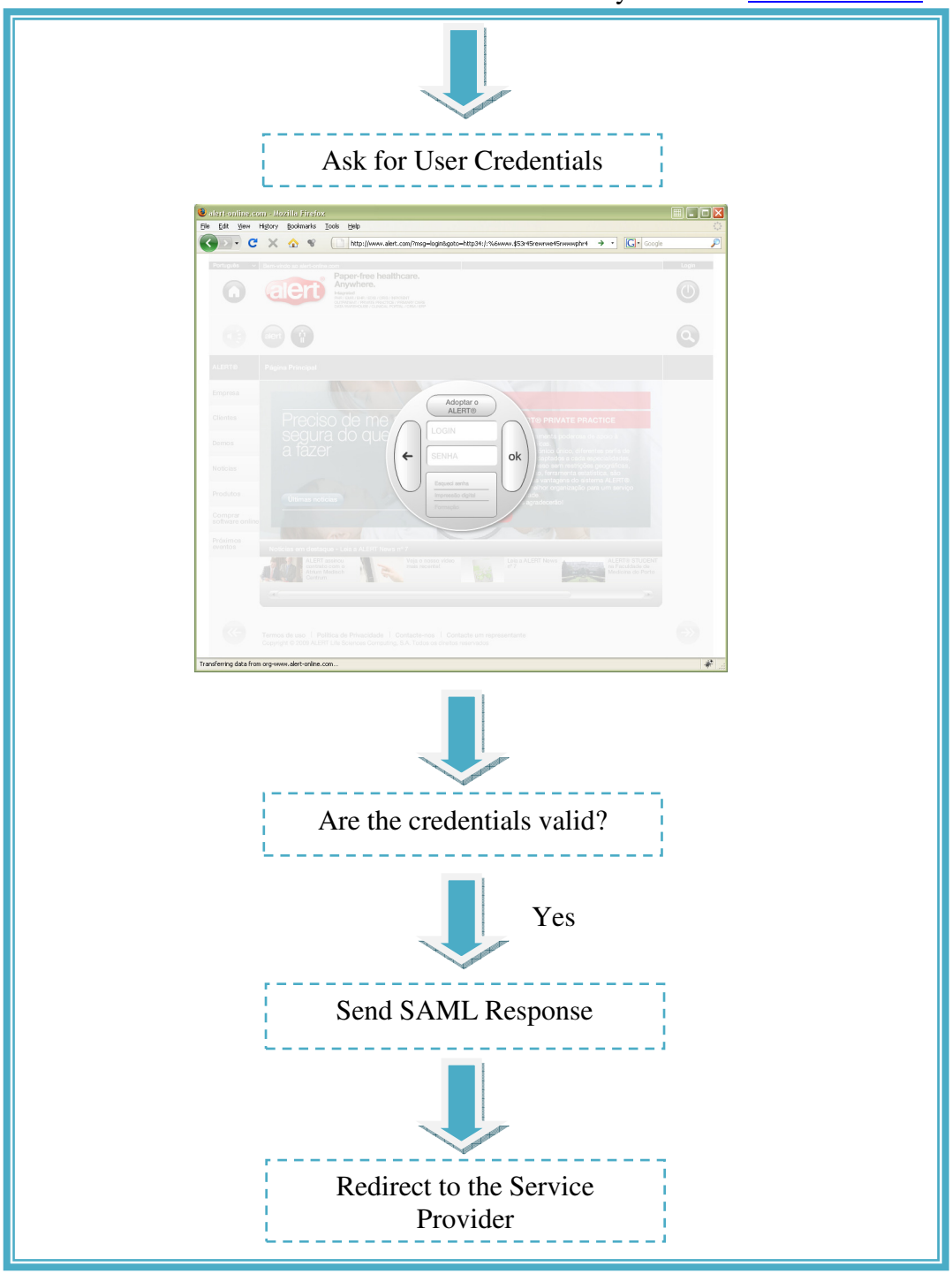


Figure 36: Example interaction on AOL and MyALERT® (Part III).

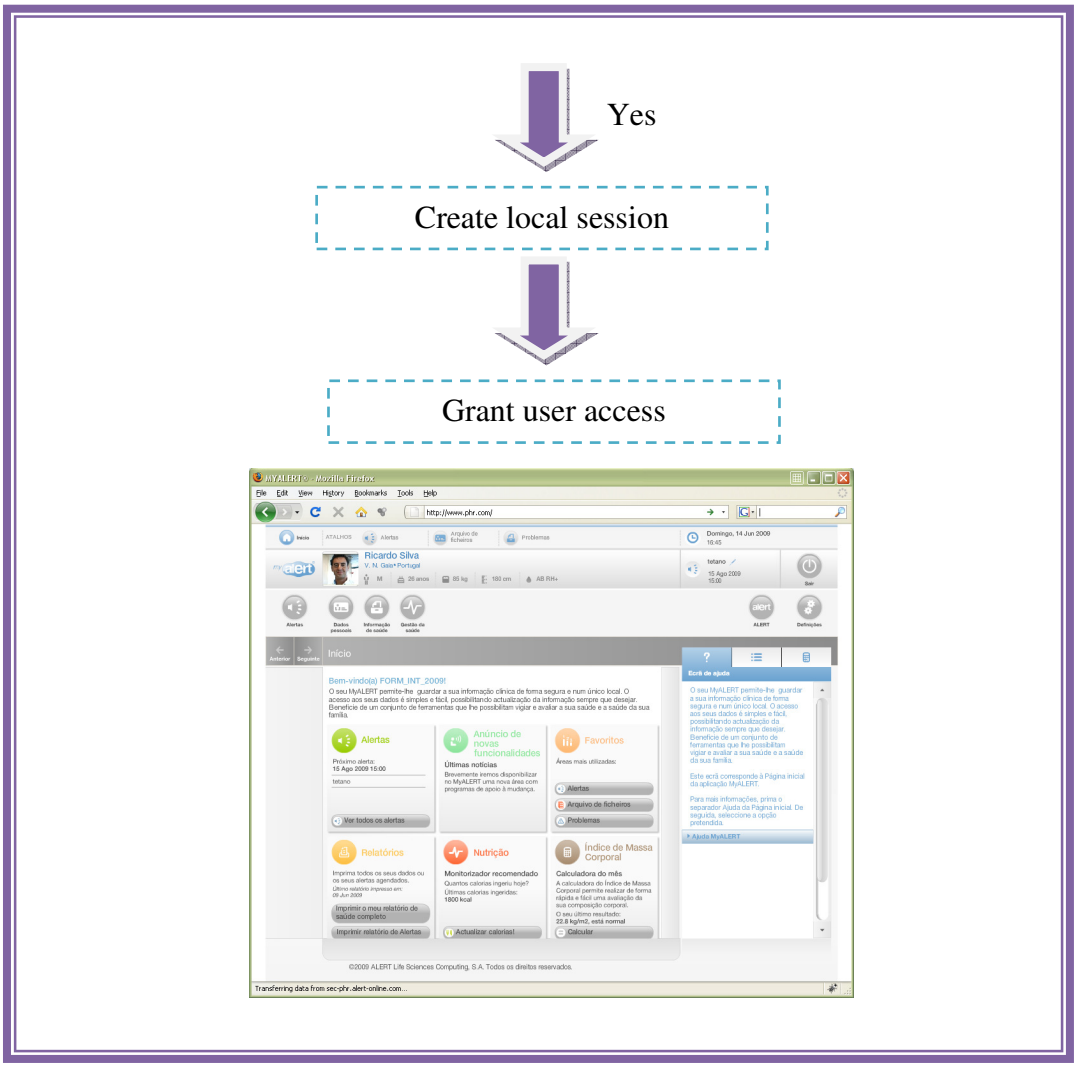


Figure 37: Example interaction on AOL and MyALERT® (Part IV).

## 5.4. Software Design

In order to implement the proof of concept, the structure of the two providers (AOL and MyALERT®) had to be studied for the software design to be adapted accordingly. The result of this analysis is explained on the following sections.

### 5.4.1. Overview

The Proof of Concept was implemented accordingly to the predefined architecture. Using OpenSSO forced to some small changes that will be explained next.

The Identity as well as the Service Provider will be composed not only by the application itself, but will also contain an OpenSSO instance. OpenSSO will be responsible for handling the SAML messages, including its creation and processing.

Therefore, this POC is composed by three major elements (Figure 38):

- User Agent – it will hold the SSO cookie and handle and redirect the SAML messages through an applet;

- Service Provider – OpenSSO is responsible for creating SAML requests and process the responses and the application represents the legacy system (MyALERT®);
- Identity Provider – OpenSSO processes the SAML requests and creates SAML requests and responses interacting with the Authentication Framework that validates the users' credentials. It also includes application specific logic (AOL logic).

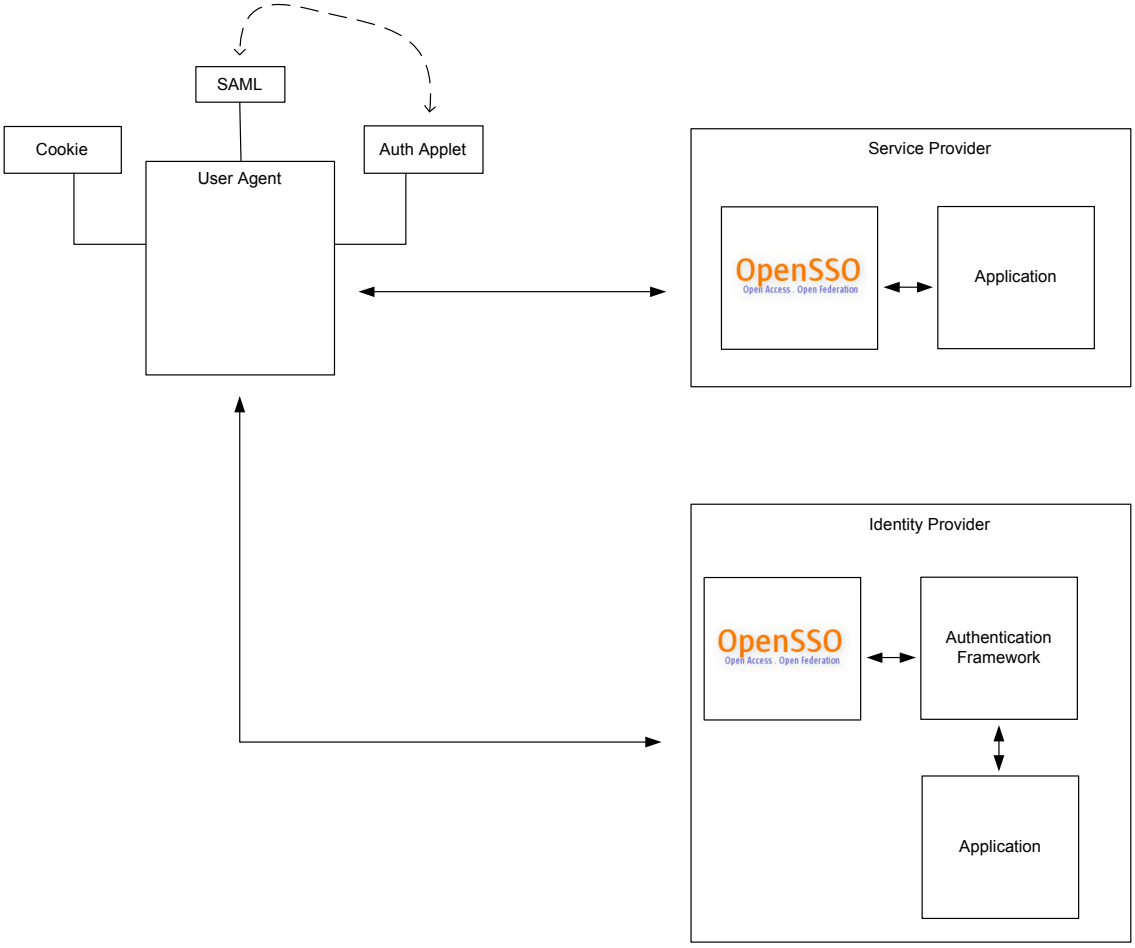


Figure 38: Black box view of the solution.

Here one can understand the importance of the utilization of OpenSSO since it allows complete separation of the application from SAML standards. This separation of concerns permits that the solution is easily integrated onto new applications since it is easily detachable.

## 5.4.2. Technology

For the implementation of the SSO solution, different technologies were used.

The choice for the programming language for the Service and Identity Provider turned out to be the simplest of all choices. The project needed one that was easy to master but, more important, platform independent, therefore Java was chosen. Not only is the language already used by ALERT® software, but also OpenSSO provides Java libraries for customization modules.

Since OpenSSO and the application don't share the same context and there was the need for Inter Process Communication, Java Remote Method Invocation was used.

When it came to the User Agent, Adobe Flash was used because ALERT® User Experience Tier already uses it and there were no disadvantage in employing it. PHP was also used for the same reasons.

The application servers where ALERT® software usually runs are Apache HTTP Server 2.2.11 and Apache Tomcat 5.5. In order to be able to link Apache HTTP and Apache Tomcat, the module jk from Apache Jakarta was employed; a module for PHP support was also used.

During the development of the framework, the main tools used were:

- Eclipse – an open source development platform, mainly designed for Java programming;
- Macromedia Flash 8 – is an advanced environment for creating interactive websites;
- Subversion – Version Control System;
- Service Capture – an application that captures all HTTP traffic sent from the browser or the IDE.

### 5.4.3. Logical View

This section aims to present a white box description of the system, exposing its inner structure. It will only be shown the components that are related to this project.

It is important to understand the different modules on each component and how they interact in order to comprehend the full extent of the solution.

On the following page, Figure 39 clarifies the interactions between the different modules can be seen, as well as the flow of SAML messages.

OpenSSO will be used for session management and SAML processing therefore:

- OpenSSO will receive and be responsible for all SAML exchange;
- On the Identity Provider, OpenSSO authentication module will be invoked in order to authenticate the user on OpenSSO so future session management is available.

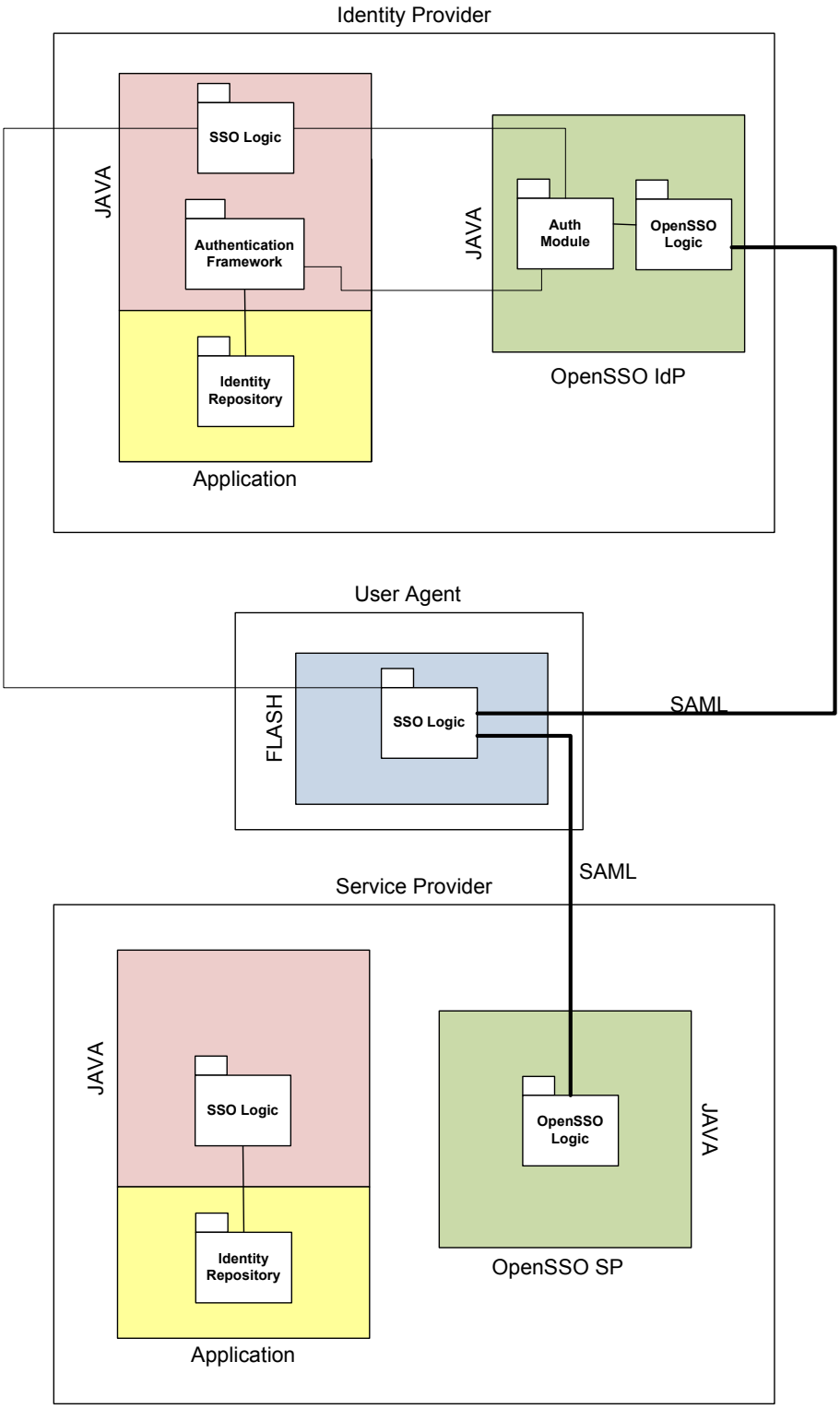


Figure 39: Logical view of the proof of concept.

## Identity Provider

The Identity Provider is composed by two parts: the application and OpenSSO (Figure 40).

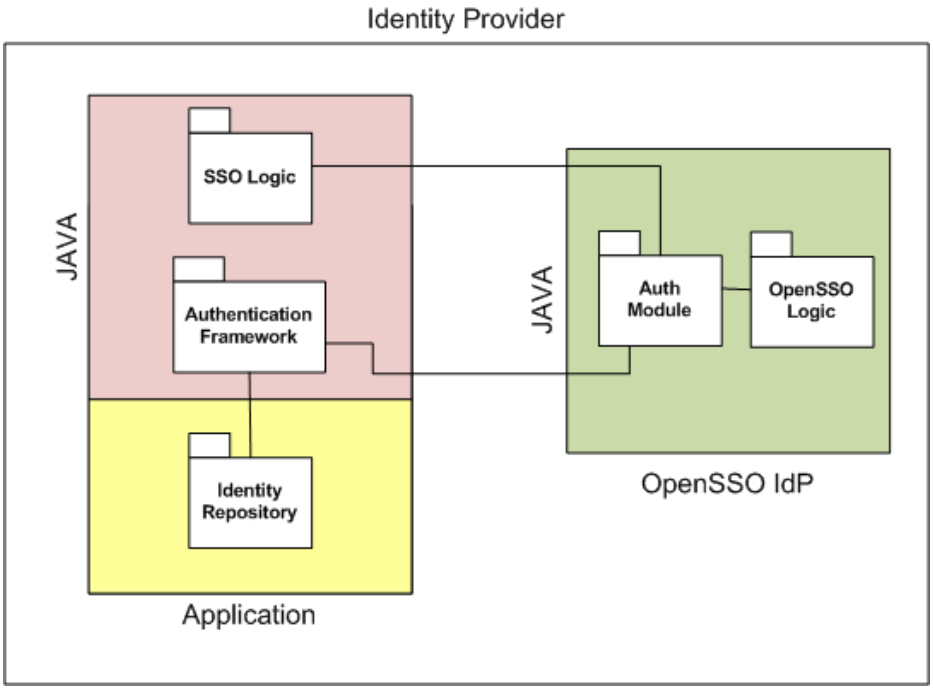


Figure 40: Logic view of the Identity Provider.

### *Application*

The application is composed by two layers: a Java layer where all the logic is built upon and another layer (which could be PL/SQL depending on the ALERT® application) that communicates with the data store.

The three major modules on the application that are related to the project are the SSO Logic, Authentication Framework and the Identity Repository.

#### SSO Logic

Its main goal is to manage all the session logic related to SSO operations therefore it:

- Receives the user credentials, sends them to the Auth module and waits for the response;
- Creates the session cookie and updates the SSO token;
- Is responsible for all other session management processing needed.

#### Authentication Framework

This framework is responsible for validating the user credentials, however it was not developed under the scope of this project.

#### Identity Repository

More than a simple data store where the user information is stored, it is also responsible for processing the data that is extracted from the database.

### *OpenSSO Identity Provider*

On the OpenSSO side, there was the need to develop a customized authentication module that could validate the user accordingly to ALERT procedures.

#### AuthModule

This module authenticates the user on OpenSSO so sessions can be managed by this software.

#### OpenSSO Logic

OpenSSO already comprehends its own logic which includes SAML processing and session management. This module is not target of any development but it was worth referring to on the logical view.

## Service Provider

The Service Provider is composed by a custom module on the application side, on the Java layer.

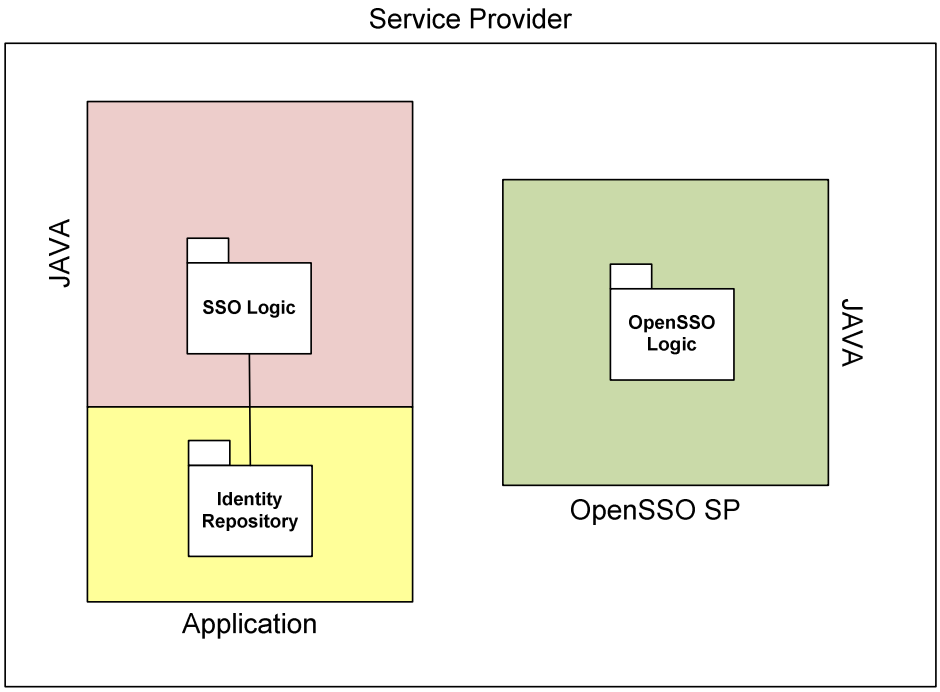


Figure 41: Logic view of the Service Provider.

### *SSO Logic*

This module will authenticate the user locally if federation is possible.

### *OpenSSO Logic*

Just like on the identity Provider, represents default OpenSSO logic.

## User Agent

The User Agent is composed only by a Flash Layer.

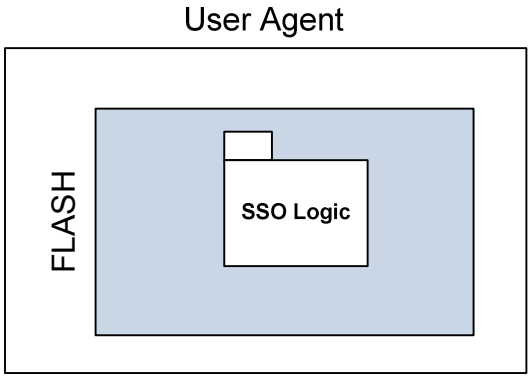


Figure 42: Logic view of the User Agent.

### *SSO Logic*

The goal of this module is to redirect the SAML messages being exchanged.

## 5.4.4. Behavioral View

On the Proof of Concept, the Single Sign On is as follows:

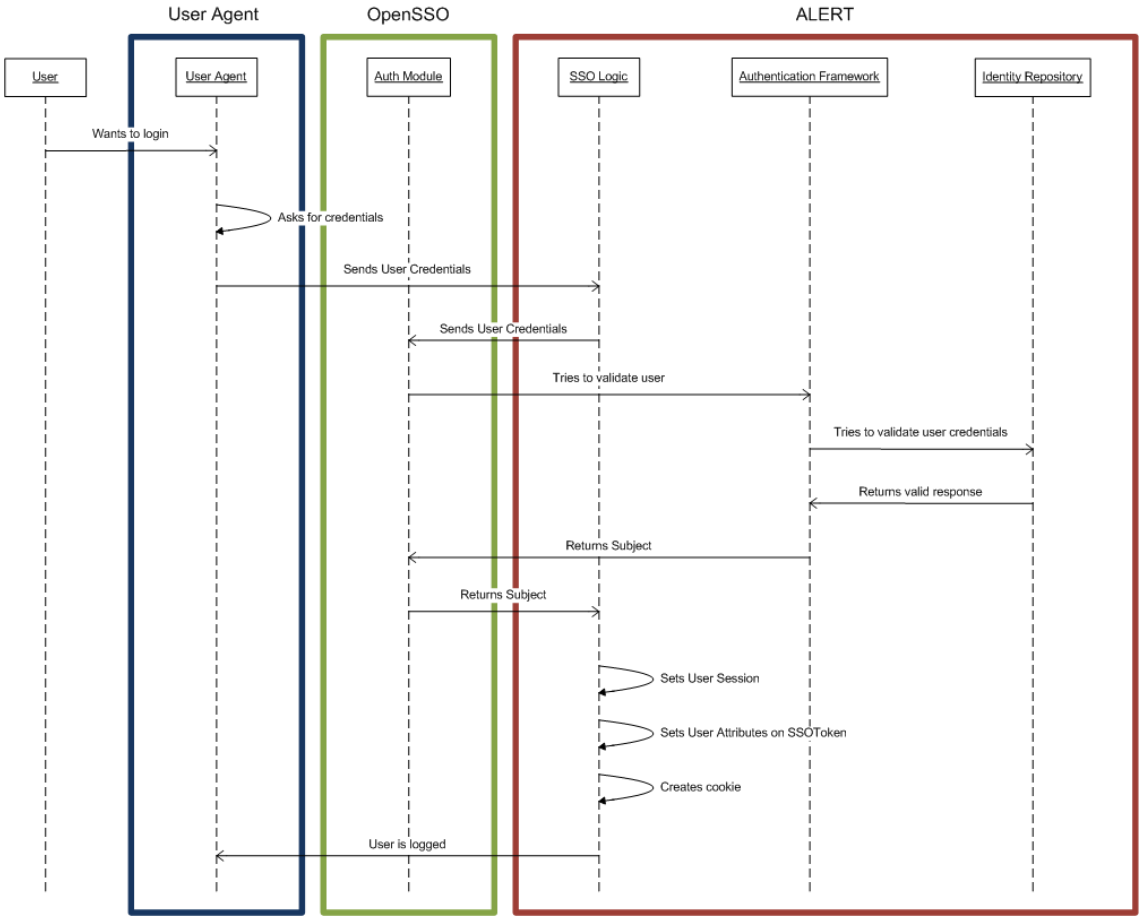


Figure 43: Process flow on the Identity Provider during Single Sign On.

On the Identity Provider, when a user wants to login at the application, the user agent retrieves its credentials and sends them to ALERT software that will redirect them to the customized module in OpenSSO. This module will create a local session after checking that the user's credentials are valid. Then he will set the user attributes used for federation on the SSO token created by OpenSSO and the user is finally logged. All these actions are transparent to the user.

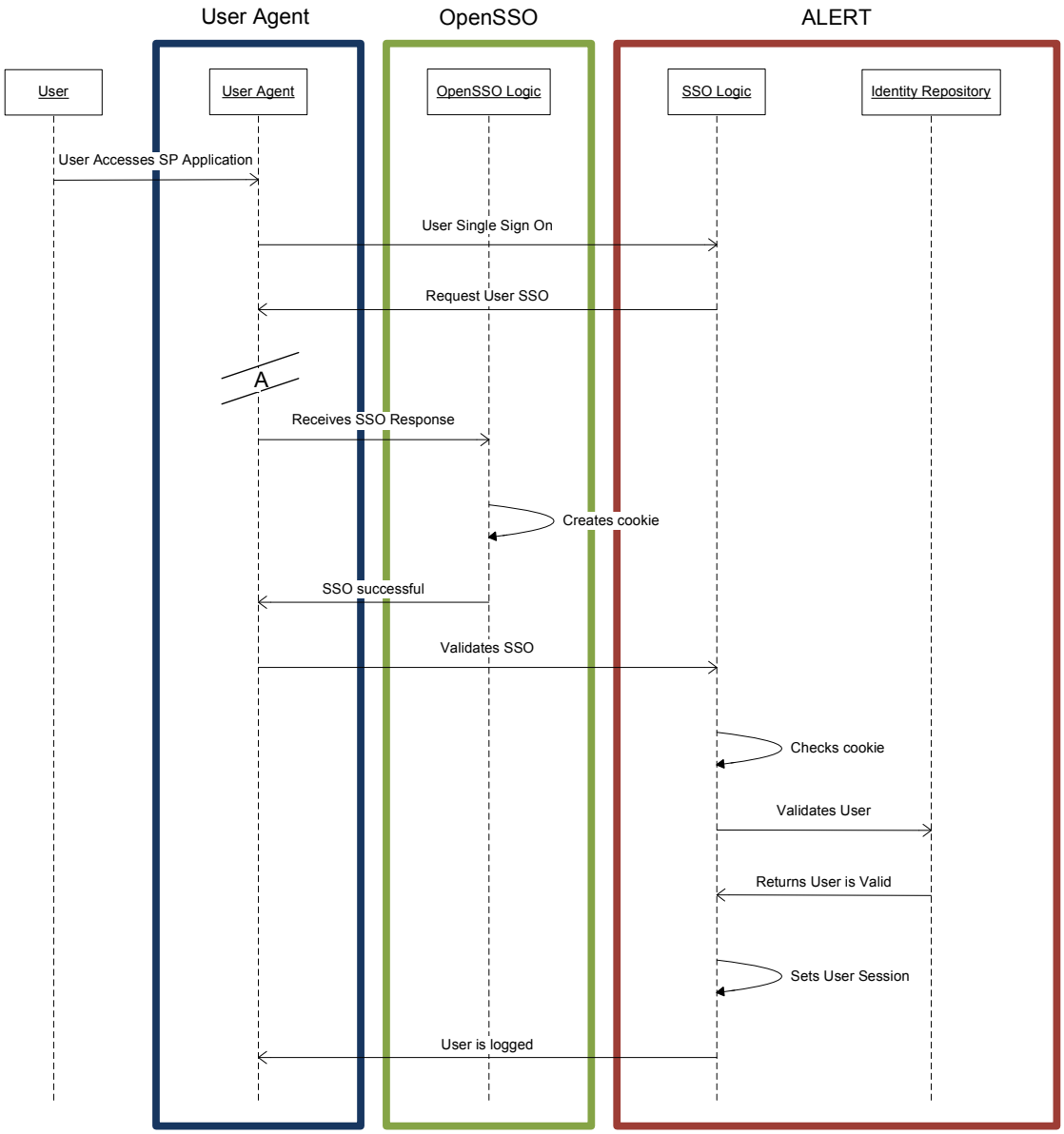


Figure 44: Process flow on the Service Provider during Single Sign On.

The Service Provider flow starts when a user wants to access the application. Since he needs to be logged on, and assuming he is not at the moment, the Service Provider will request SSO. On "A" the flow was omitted because it was already described. After receiving the response, OpenSSO will automatically create a cookie containing the SSO token and the user is redirected to the Service Provider. If the cookie exists, the Service Provider will try to validate the user by performing federation actions. Given the user is valid, it sets the user's session.

For the Single Logout sequence diagrams, it was chosen not to include Service Provider X since it is not relevant for the implementation since its behavior is very simple.

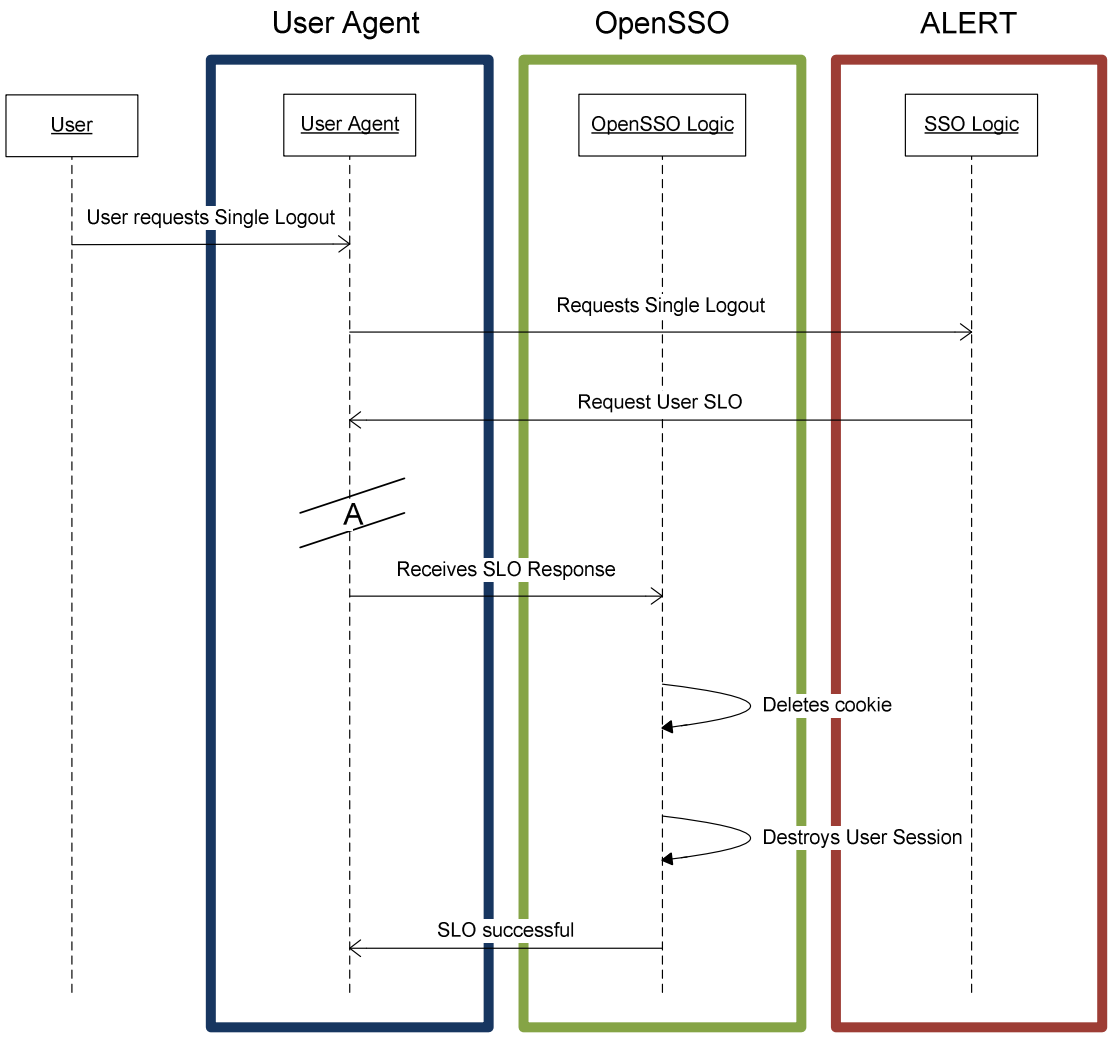


Figure 45: Process flow on the Service Provider side for SP-initiated SLO.

Figure 45 refers to the flow on the Service Provider when the user requests Single Logout. A SAML SLO request will be created and sent to the Identity Provider. The flow on the Identity Provider hidden on “A” is visible on the next figure.

After receiving a SLO response, the OpenSSO on the Service Provider will delete the SSO cookie and destroy the user session. Single Logout is then achieved.

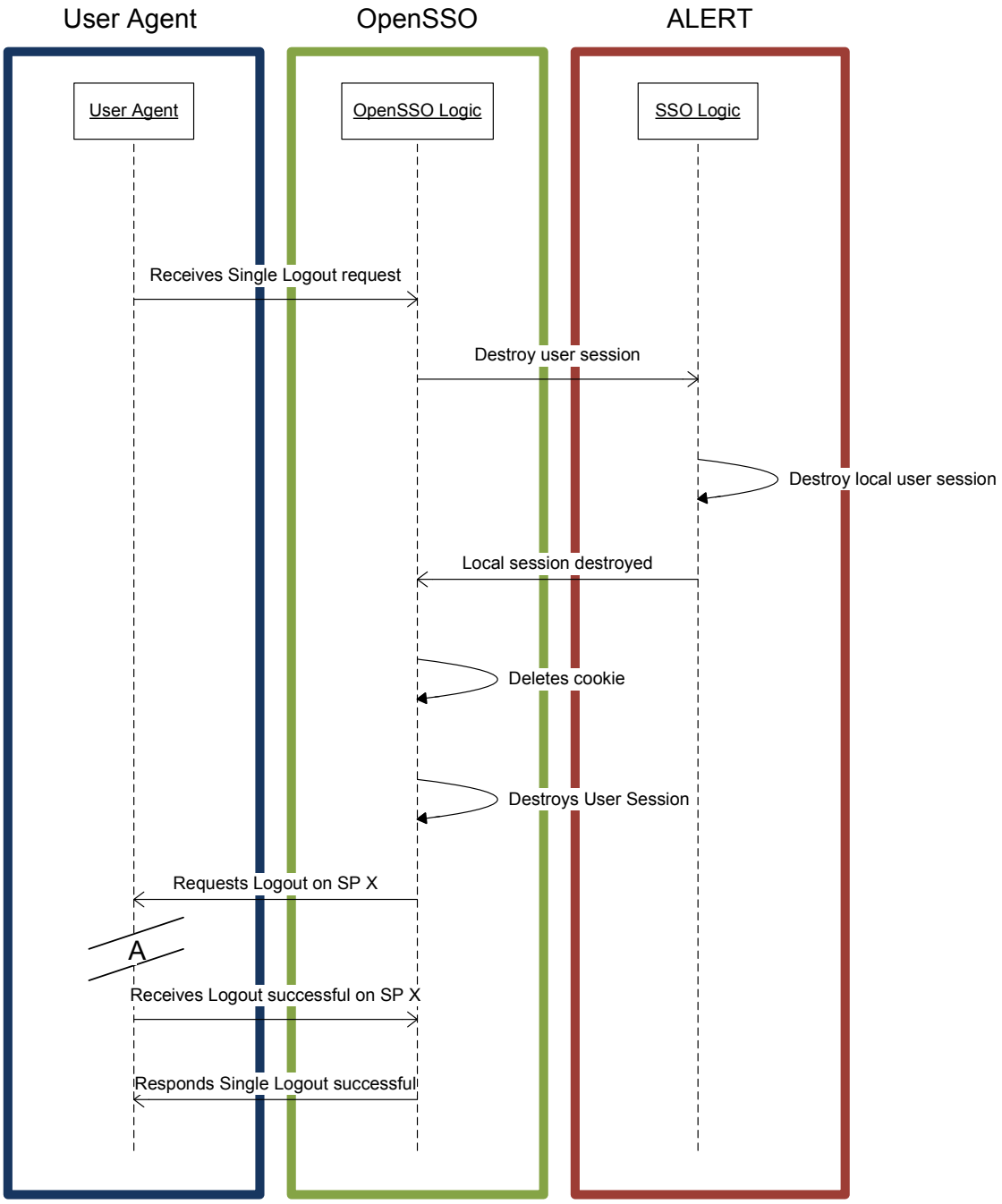


Figure 46: Process flow on the Identity Provider side for SP-initiated SLO.

After receiving a SLO request, OpenSSO calls the module SSO Logic through RMI to destroy the user session on ALERT. After ALERT's user session is destroyed, it will delete the SSO cookie and local session details. It is time to request logout from all other Service Providers where the user is logged in to. After receiving successful Single Logout responses, it will respond to the Service Provider that initiated the request. The processing done by Service Provider X was omitted (represented by "A").

## 5.5. Implementation

After adapting the initial architecture with OpenSSO usage, it was time to develop the framework for Single Sign On.

The implementation was more difficult than expected since AOL and MyALERT® are complex applications and understanding their flow was not straightforward. The same applies to the deployment on a local environment. The major cause for these difficulties was the insufficient documentation available:

- there were no appropriate guides about how to deploy AOL and MyALERT® locally since this was never done before;
- regarding the applications flow, the documentation proved to be inadequate for someone who isn't familiar with them.

### 5.5.1. Deployment Environment

In order to develop the solution, a local environment had to be set up. Three different workstations were used to emulate real life scenarios, where all three components are physically separated.

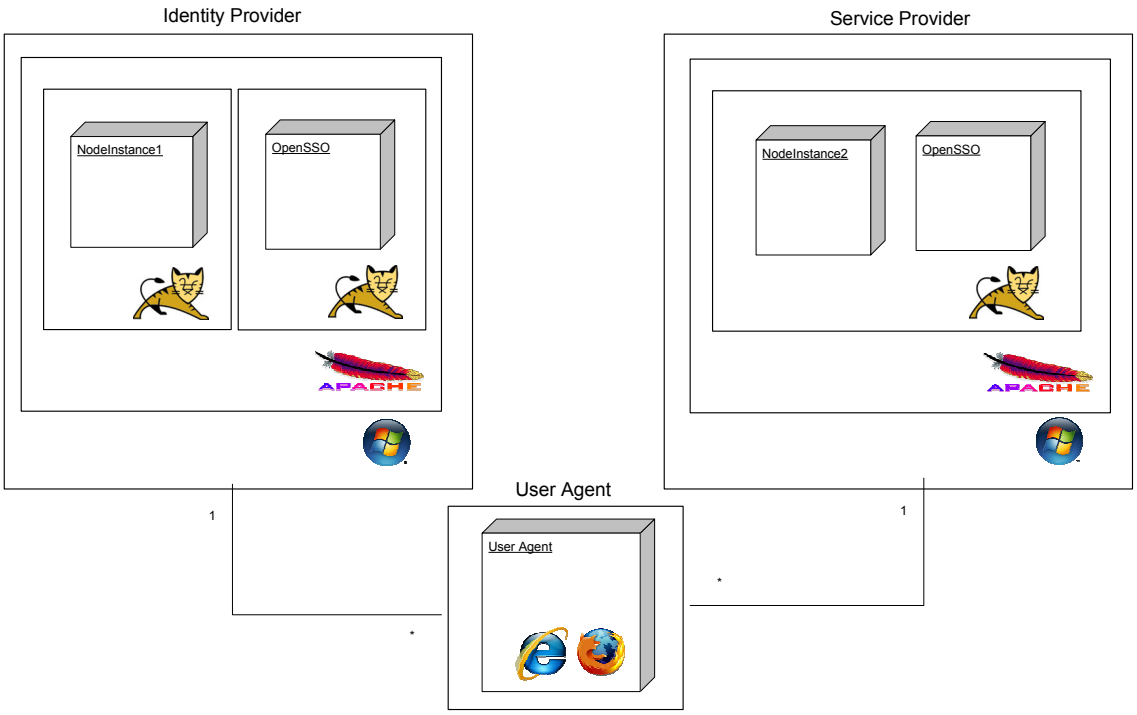


Figure 47: Deployment environment of the proof of concept.

## Identity Provider

On the Apache HTTP Server, different Virtual Hosts were created for AOL and OpenSSO. After setting up all the configurations needed, they were linked to Apache Tomcat through workers that use the jk module. This module is a part of the Apache Jakarta Project that enables linking both servers. A PHP module was also included.

OpenSSO and AOL were deployed into different Apache Tomcat Servers.

## Service Provider

For the Service Provider, the same virtual host as well as the same Tomcat instance was used for both MyALERT® and OpenSSO.

### User Agent

No setup was done for the User Agent.

## 5.5.2. Computer Specifications

As already mentioned three separate computers were used for deployment and testing. Each computer specification is as follows.

Table 17: Computer specifications.

| | Identity Provider | Service Provider | User Agent |
|---|---|---|---|
| Number of processors | 2 | 2 | 2 |
| Processor | 3.00 GHz | 1.86 GHz | 3.00 GHz |
| Cache Memory | 4,00 GB | 2,00 GB | 3,00 GB |
| Architecture | 32 bits | 32 bits | 32 bits |
| Operating System | Windows Vista | Windows Vista | Windows Vista |

## 5.5.3. Details

Before proceeding to the framework implementation, some details have yet to be clarified.

### Cookies

Each time a SAML response is created or received, a cookie is also created and associated with it. In order to detach the utilization of SAML of the SSO operation, an SSO token is created by OpenSSO that contains details about the authenticated user. Therefore there will be two cookies: on the Identity Provider after successful login indispensable for following SSO actions, and on the Service Provider after receiving a successful response from an Authentication Request.

### Auxiliary files

#### *ALERT® ONLINE*

- opensso.properties – contains information about the OpenSSO deployment such as its naming URL;
- openssoauth.properties – contains information about the realm where login will occur such as the name of the authentication module and the cookie name.

#### *MyALERT®*

- opensso.properties – in a similar way to the equally named file on the Identity Provider, it also contains information about the OpenSSO deployment.

- openssocookie.properties – contains information about the cookie created by OpenSSO.

### 5.5.4. Code

It was not considered relevant to include all the details about the framework, but rather its main components. Development was needed in the Java and Flash layer, as well as on index.php. Most of the implementation makes use of OpenSSO SDK libraries, especially for login actions.

## ALERT® ONLINE

### *Java*

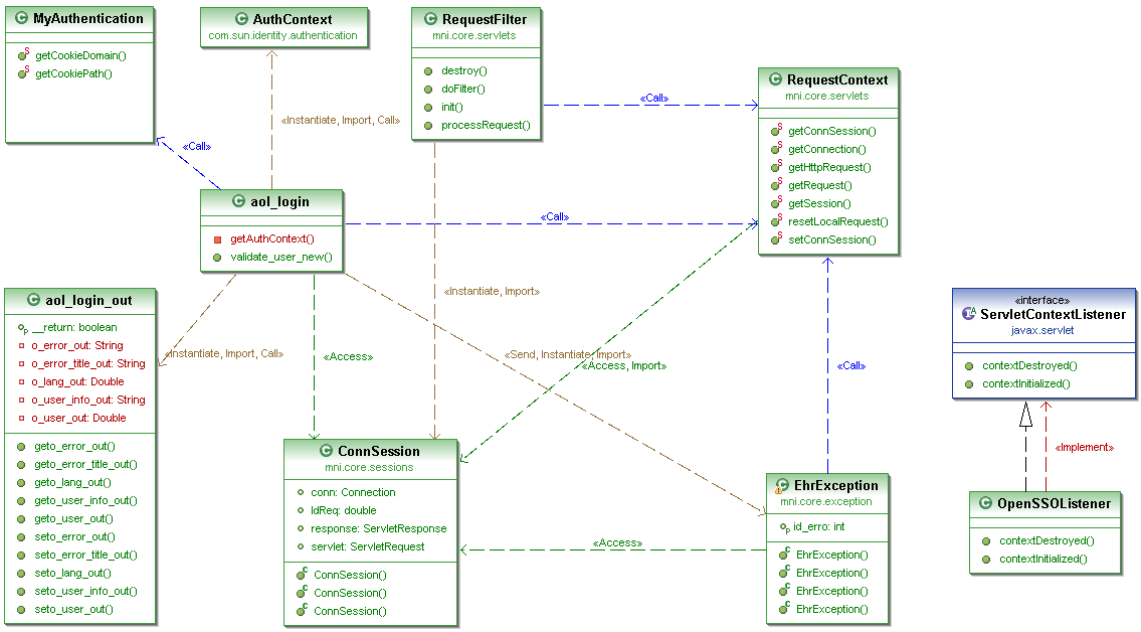


Figure 48: Java class diagram of ALERT® Online.

package mni.core.servlets

**class RequestFilter implements javax.servlet.Filter**

In order to allow cookie creation a ServletResponse had to be available.

- public void processRequest(ServletRequest request, ServletResponse response, BufferedHttpRequestWrapper wrapped, FilterChain chain, Connection c, double idRequest) throws ServletException, IOException – the parameter ServletResponse was added and the method was restructured.

package mni.core.sessions

**class ConnSession**

Since *RequestFilter* invokes *ConnSession*, the constructor also had to be altered to support the ServletReponse parameter.

package mni.aol.sso

**class Aol_login**

The goal of this class is to validate the users' credentials and create its session.

- private com.sun.identity.authentication.AuthContext getAuthContext(String orgName, String i_desc_user, String i_pass_user)throws AuthLoginException, IOException – gets OpenSSO authentication context and submits the users credentials to the OpenSSO login module (*ALERTLogin*);
- public aol_login_out validate_user_new(String i_desc_user, String i_pass_user, Double i_id_language) throws EhrException – logs the user into OpenSSO and, if successful, proceeds to set the federation attributes onto the SSO token and creates the cookie that contains the OpenSSO token.

**class Aol_login_out implements Serializable**

Aol_login_out implements the output for Aol_login, containing variables for errors and user info.

**class MyAuthentication**

Reads OpenSSO custom authentication properties and stores them.

**class OpenSSOListener**

This listener is used to initialize OpenSSO properties and load them into the System Properties.

## *Flash*

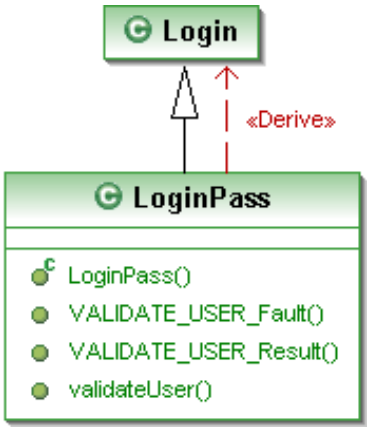


Figure 49: Flash class diagram of ALERT® Online.

It was chosen not to include all methods of both classes on Figure 49 since they weren't the target of any development.

**class mni.online.screens.LoginPass extends mni.online.screens.Login**

This class extends the Login and is used for user sign on using a password.

- private function validateUser(user_str:String, pass_str:String):Void – responsible for user validation and invoke *validate_user_new*;

- private function VALIDATE_USER_Result(validate_re:Object):Void – if user login operations occurred successfully, it will redirect the user to the OpenSSO *goto* URL.

### *PHP*

#### index.php

Index.php was altered to reflect the new application flow.

## OpenSSO Identity Provider

### *Java*

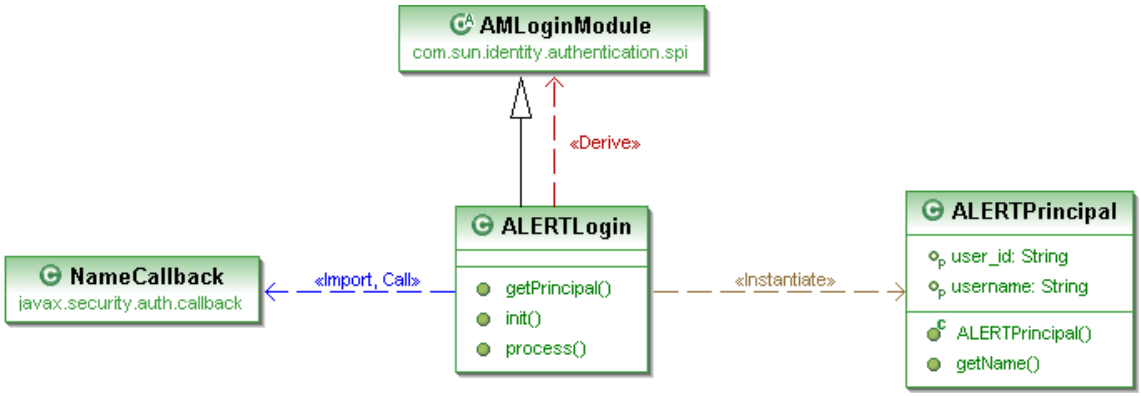


Figure 50: Java class diagram of OpenSSO custom authentication module.

Since the solution is very particular because neither AOL nor MyALERT® use LDAP data stores, there was the need to develop a custom authentication module to deal with authentication on OpenSSO. The XML file with the module configuration was also included on OpenSSO.

package authentication

#### class ALERTLogin extends AMLoginModule

Since ALERTLogin extends AMLoginModule, it needs to override three methods. The most important method for user sign on is described next:

- public int process(Callback[] arg0, int arg1) throws LoginException – validates the user credentials against the database, creates an ALERTPrincipal and returns -1 if validation was successful.

#### class ALERTPrincipal implements Principal

This class implements the Principal therefore it contains the getters, setters as well as the constructor.

## MyALERT®

### *Java*

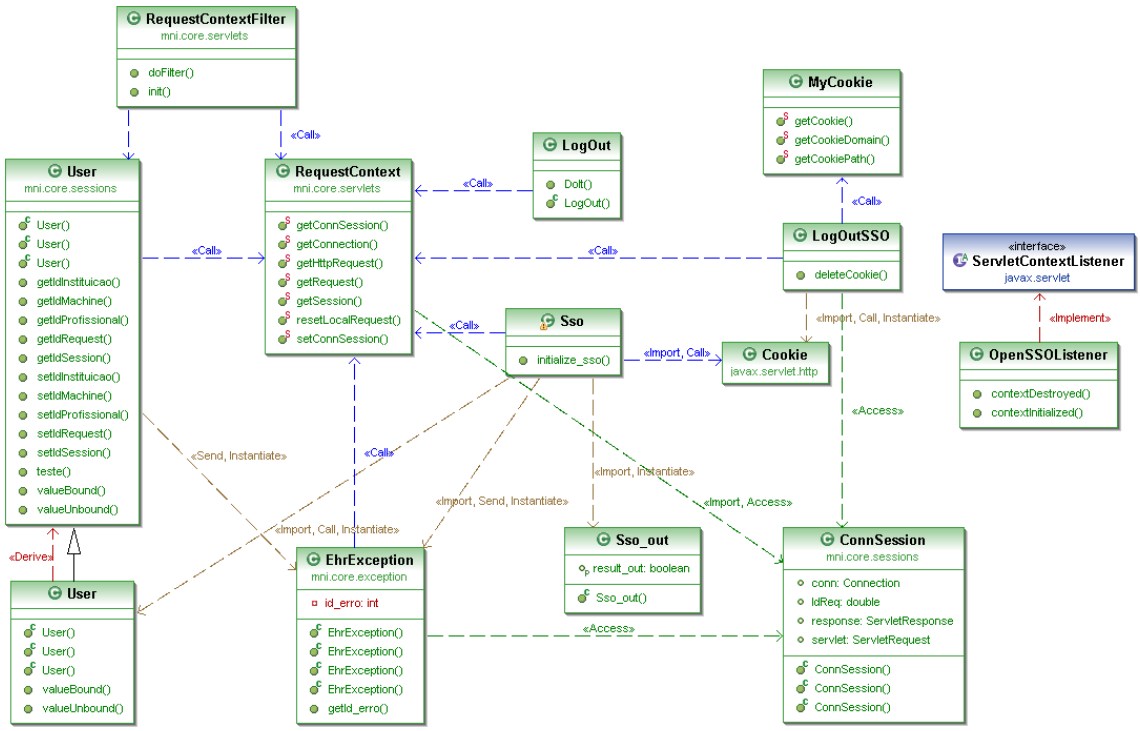


Figure 51: Java class diagram of MyALERT®.

package mni.phr.plsql.*

All the classes had to be restructured in order to act accordingly to the Single Sign On flow, especially in regard to the user session.

package mni.core.servlets

**class RequestContextFilter**

The filter had to be reconfigured in order to allow *initialize_sso()* to be invoked before checking for user permissions.

package mni.core.sessions

**class ConnSession**

Just like on AOL, the constructor also had to be altered to support the ServletReponse parameter.

package mni.phr.sessions

**class LogOutSSO**

This class is responsible for clearing the cookie created by OpenSSO that contains the SSO token. It is used for local logout.

- public Boolean deleteCookie() – retrieves the OpenSSO cookie and deletes it.

**class MyCookie**

Reads OpenSSO cookie properties and stores them.

**class Sso**

This class is responsible for initializing Single Sign On operations. It validates if the user has already logged in or not.

- public Sso_out initialize_sso() - checks for the existence of the OpenSSO SSO cookie. If it exists, it compares the attributes stored on the SSOtoken with the ones existing in the user data store:
  - if mapping is possible then federation is possible, therefore it creates a local user session in MyALERT® and returns *true*;
  - in case mapping is not possible or the cookie doesn't exist, it will return *false*.

**class Sso_out implements Serializable**

Sso_out is used to implement the output for Sso. It implements getters and setters as well as the constructor.

**class OpenSSOListener**

This class presents the same behavior as on the Identity Provider.

## *Flash*

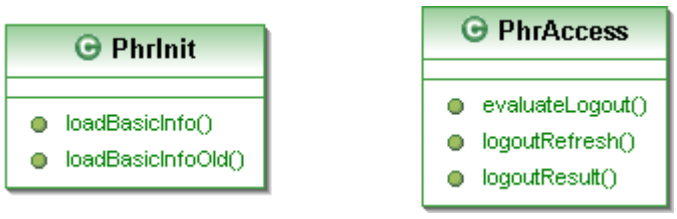


Figure 52: Flash class diagram of MyALERT®.

Again, only the relevant methods of the classes were included.

**class mni.phr.entrance.PhrAccess**

This class controls access and buttons events.

- private function evaluateLogout(buttonParam):Void – evaluates the exit;
- private function logoutResult():Void – evaluates the result of the function mentioned above;

- private function logoutRefresh():Void – refreshes the browser after all logout actions are done.

**class mni.phr.entrance.PhrInit**

This class initializes MyALERT® variables and services.

- private function loadBasicInfo():Void – initializes global managers and invokes *initialize_sso()*;
- private function loadBasicInfoOld(re:ResultEvent):Void - loads basic info with the services complete and, in case the return from *initialize_sso()* was:
  - *true* - call user preferences and saves them into the global scope;
  - *false* – invokes Single Sign On.

### *PHP*

**index.php**

This file was modified in order to include more parameters to pass for the flash layer, but also to change its normal behavior related to ALERT® Online.

## 5.5.5. OpenSSO Setup

For each OpenSSO instance a Fully Qualified Domain Name was created:

- www.alert.com for ALERT® Online;
- and www.phr.com for MyALERT®.

Afterwards, there was the need to setup the JVM for each Apache Tomcat instance so that OpenSSO could be correctly deployed and could function properly. A custom configuration was made for each OpenSSO Provider.

After the initial deployment, specific configuration was needed. The most important steps of this stride will be described next.

To create the SAML v2.0 providers there was the need to set up a Hosted Identity Provider and a Remote Service Provider:

- for the Hosted Identity Provider, a Circle of Trust was created and a signing key was enabled;
- for the Remote Service Provider, the metadata had to be exported.

On the Service Provider, analogously to the Identity Provider, there was the need to setup a Hosted Service Provider and a Remote Identity Provider:

- for the Hosted Service Provider, the same signing key and circle of trust as of the Identity Provider were used;
- for the Remote Identity Provider, metadata had to be exported.

Since there is no real linkage between the user data stores from the Service and the Identity Provider, the user profile had to be set to *ignored*. For the same reason, on the attribute mapper the values for federation had to be set (in this case, *userid*) and, on the Service Provider, auto federation was enabled.

For security reasons, it was chosen that all the assertions had to be digitally signed.

Finally, after the custom authentication module was developed, it was integrated into OpenSSO Identity Provider.

## 5.6. Tests

After implementing the solution, it was tested in order to assess its quality. Therefore the developed solution was tested against the project specification including its goals and requirements.

The solution's scope was to implement a Single Sign On solution using SAML, which was done with the aid of OpenSSO. Unit testing wasn't conducted because it was more relevant to the final solution the deployment configuration than the developed code. Following the project's ambit, integration tests, as well as interoperability and usability tests, were better suited.

All the tests presented a successful output.

### Integration

The final solution allowed a clear and distinct interaction between Service and Identity Provider.

It allows authentication services to be separated from the Service Provider which will only deal with specific application logic. Identity Provider is the component responsible for authentication.

ALERT® ONLINE was easily integrated with MyALERT® because they were already separate services, even though the user's authentication was done on MyALERT® . Since the prototype had already been developed, it was clear where to interfere with the applications flow and make changes.

Nevertheless, the solution cannot be considered final because MyALERT® 's user data store has to reside in ALERT® ONLINE.

### SAML usage

Single Sign On was achieved using SAML exchange and according to standards. It was used SAML v2.0 and Single Sign On protocols. The messages that are exchanged during Single Sign On actions can be consulted on Appendix D.

### Interoperability

Interoperability tests for other operating systems on the server side could not be conducted because there wasn't equipment available for testing.

Nevertheless, for the user agent the following tests were made:

- Operating System
  - Windows 32bits (Windows Vista);
  - Linux 32bits (Ubuntu 8.10);
  - Mac OS 64bits (Mac OSX 10.5 Leopard).
- Browser
  - Mozilla Firefox 3.0;
  - Internet Explorer 7.0;
  - Google Chrome 2.0.

All the tests were successful.

### Usability

Comparing the initial usability of MyALERT® and ALERT® Online to the final, it was concluded that it was not compromised, rather simplified by providing a single point of entrance for MyALERT® .

## 5.7. Solution Evaluation

The solution was developed to implement a Single Sign On solution on ALERT® applications. ALERT® Online and MyALERT® were the applications chosen for the proof of concept. After the solution has been finalized it is time to compare it with the initial requirements in order to measure its success.

### 5.7.1. Overview

Evaluating the proof of concept is a complex and multifaceted task. The final solution implemented Single Sign On protocols through the utilization of SAML and state of the art technologies.

Even though only SAML v2.0 was used, altering the solution to also support SAML v1.x, is expected to be a simple task since all major mechanisms for SSO are already set up. Nevertheless one of the goals was also to create a framework that could be easily extensible to support other standards and protocols: because of its modularity the solution is easily detachable so this is also considered to be a plain chore.

In comparison to the prototype, no Service Provider adapter was used. The non-usage of this module was a restriction inherent from MyALERT® 's flow and Tomcat contexts:

- MyALERT® sets an attribute on the session with the User that is currently logged in;
- On Tomcat, each application is seen as a different web app which means no context is shared between them;
- The SPAdapter runs on the OpenSSO instance which is deployed in a different context than the one from MyALERT® ;
- The initial idea was to use RMI on the SP Adapter to connect to MyALERT® and verify if federation was possible and afterwards set the session.

The problem is a result of all the above factors: because OpenSSO and MyALERT® don't share the same context, they don't share the same session therefore setting the session attribute with the User wasn't possible. This is the reason why, on the proof of concept, federation validation is done on MyALERT® after receiving the SAML response.

High availability was not achieved since it obligated that there should be at least two OpenSSO Enterprise instances per provider and they should be deployed behind a load balancer. The main reason why it was not used, was because MyALERT® and AOL were only used as a proof of concept. When going to production, load balancing should be set up. It is not expected that any type of complications arise on the Single Sign On deployment.

Since the solution is SAML compliant it can support other Identity Providers rather than ALERT's own, which leads to MyALERT® making its SAML requests to this other Identity Provider. No other changes have to be made.

### 5.7.2. Non-functional requirements

Making a comparison between the final solution and the initial non-functional requirements, the following can be perceived:

- Interoperability
  - With ALERT® applications interoperability was achieved, since the proof of concept was made on AOL and MyALERT® , integrating with other ALERT applications should be straightforward;
  - The only authentication method tested was username plus password because it is the method used by AOL and MyALERT®. Supporting multiple authentication systems is considered an easy task, only needing SAML configurations according to the authentication method used.
- Platform compatibility and portability
  - On the client three different Operating Systems were tested and the output was successful;
  - For the Identity and Service Provider, it was not possible to test another operating systems because of equipment restrictions;
  - All the tests performed on browsers had a positive output.
- Security
  - Since HTTPS wasn't used, cookie injection is possible which leads to session hijacking[4]. When HTTPS is put into use, the probability of occurring session hijacking diminishes and can be eliminated by storing another type of information (it could be the MAC address of the computer that created the cookie or its IP address).
- Extensibility and scalability
  - Adding new features is considered to be straightforward;
  - the final solution is modular thus can be easily integrated.

[4] For a long time, Google functioned with SAML and was aware of the described security leach. Google only resolved this problem a few months ago.

# 6. Conclusion

The final goal of this project was to design a Single Sign On solution that was compliant with SAML standards.

One of the objectives for the Single Sign On implementation was the support for non-web applications, which was proven not to be possible with SAML. Since SAML was designed for online applications, using it for something rather than it was designed for implies changes that would compromise the standards. A valid alternative could involve the usage of Kerberos tickets, which allows cross user authentication (cross domain and realm authentication).

For web applications, a proof of concept that achieved the defined goal was successfully implemented on two ALERT® applications: Online and MyALERT®.

Nowadays, Single Sign On solutions exist on ALERT® software but they are custom made and don't respect any standards. This compromises interoperability with other applications since they were specially designed for ALERT® applications and the other applications may not be compatible with the designed flow.

Altering the existing solutions to be compliant with SAML standards opens the way for ALERT® applications to interoperate with third party tools.

During the solution design and implementation, various difficulties emerged. The overcome challenges range from the set up of local environments using Apache servers to the configuration of complex software such as OpenSSO.

One of the most valuable outcomes derives from the implementation of SAML standards that allow to perform Identity Federation. Identity Federation's advantages come as very important for nowadays applications and businesses:

- It helps to reduce costs and complexity by allowing organizations to collaborate freely;
- through quicker and simpler access to more services and products, the user experience was improved;
- enterprise security was enhanced because each user only needs one pair of credentials.

OpenSSO proved to be an excellent choice given that:

- when used with ALERT® applications, it can provide a centralized security policy and infrastructure that mitigates the risks from both internal users and external threats;
- it mitigates operational inefficiency by reducing the need to duplicate resources, since it is not required to create a separate identity infrastructure for each new application or online service;
- it provides a SDK for customizations;
- and enables effective access management.

OpenSSO SDK played a major role on the final solution because it helped in the integration with the legacy systems as well as it helped to overcome some of the OpenSSO restrictions.

Creating a prototype that used OpenSSO was definitely one of the major factors that lead to the project's success. OpenSSO is a very complex and extensive solution that offers multiple deployment options: deploying a prototype helped to discover the most appropriate configuration for the solutions' goal. It is assumed that its implementation lead to a better final structure of the proof of concept as well as more efficient coding.

It can be concluded that the initial requirements analysis was broad and took into account all the concerns that a SSO solution must handle. The designed architecture provided a strong modular basis for software extensibility and scalability, that allows easy integration with legacy systems.

The software library that was developed is ready for integration with other systems and, because it was developed in Java, is interoperable with other platforms rather than Windows.

It was written extensive and detailed technical documentation of the architecture, technological review and final solution that allows users that aren't familiar with the solution to become easily and rapidly acquainted with it.

The developed framework is ready to be put into production. Since it is modular, integrating it with new applications is simple, only depending on configuration.

## 6.1. Future Work

During the research and development of the project, there were identified features that would benefit the final solution.

Even though SAML v2.0 is the version more commonly used, a framework that implements SAMLv1.x Single Sign On could be developed. This would allow ALERT® applications to interact with all applications that support SAML standards, no matter what version they have chosen to implement. The two frameworks would be available and, during the deployment, through simple configurations, one of the two frameworks would be included, according to the application goals.

The only situation where the two SAML versions had to be supported would be on the Identity Provider. Since OpenSSO supports all available SAML versions, in order to support SAML v1.x only configuration would be needed.

The major goal for the SAML v1.x framework would be to improve interoperability with other legacy systems.

Another option would be to use SAML on an Authorization framework. Since SAML has the capability of encapsulating XACML, SAML could be used for communications on an Authorization framework, more particularly, between the Policy

Information Point and external servers as well as between the requester and the Policy Enforcement Point.

Again, since OpenSSO supports authorization requests and XACML, this new feature could be implemented using OpenSSO SDK.

Finally, an additional feature could also be the simple exchange of user profiles. Integrating the Healthcare Enterprise (IHE) advices the utilization of SAML profiles for cross enterprise user authentication. This feature is more appropriate for the healthcare market.

# Glossary

**Authentication Framework**

Responsible for authenticating a user, usually by checking if the provided credentials are valid.

**Authorization Framework**

Responsible for making authorization decisions, such as if a specific user is able to access a particular resource.

**Biometrics**

Refers to the technologies that use human characteristics like fingerprints or eye retinas for authentication purposes.

**Circle Of Trust**

A federation of Service and Identity Providers with whom principals can transact business in a secure environment.

**Datacenter**

A facility used to store servers and associated components.

**eXtensible Access Control Markup Language**

It is an XML language for declarative access control policies, including a processing model that describes how to interpret the policies.

**Identity Provider**

Manages credentials of the individual end users and verifies if they are valid. Rather than being just a simple database where the user credentials are stored, it also presents logic that allows to authenticate and access information about users.

**Legacy system**

A legacy system refers to an old application that continues to be used because it is not viable to replace it.

**Man in the middle attack**

It is an attack where independent connections are made in order to confuse the victims to make them believe they are talking directly to each other, when in fact the entire conversation is controlled by the attacker.

**Organization for the Advancement of Structured Information Standards**

OASIS is a not-for-profit consortium that drives the development, convergence and adoption of open standards for the global information security [OAS100].

**OpenSSO Enterprise**

OpenSSO is the single solution for Web access management, federation and Web services security offered by Sun.

**Policy Enforcement Point**

It is the logical entity that is responsible for installing and enforcing policies.

**Policy Information Point**

A Policy Information Point is the source of attributes values of a policy.

**Proof of concept**

POC is the evidence which demonstrates that an innovative approach is viable, feasible and capable of solving a particular problem. It is drawn from actual experience using an innovation in a real-world environment for a sufficient amount of time to prove that the model: provides the intended results.

The most competitive applicants can show that they have assessed the effectiveness of the proposed approach and have incorporated lessons learned in preparation for replication or scaling up.

**Security Assertion Markup Language**

SAML is an XML-based standard developed by the Security Services Technical Committee of the Organization for the Advancement of Structured Information Standards (OASIS). Its major goal is to solve the Web Single Sign On problem.

**Service Provider**

Entity that provides services. It stores the resources that the user wishes to access thus needs to know whether or not the user is authenticated.

**Session hijacking**

Session hijacking is the exploitation of a valid computer session and is used to gain unauthorized access to resources.

**Single Sign On**

SSO is a mechanism whereby a single authentication action grants access to all other systems where the user has access permission.

**Single Logout**

SLO permits near real-time session logout of a user from all active sessions associated with a user.

**Sun Microsystems**

Sun Microsystems, Inc. is an American company that provides network computing infrastructure solutions that include computer systems, software, storage and services. Its core brands include the Java technology platform, the Solaris operating system, MySQL, StorageTek and the UltraSPARC processor [Sun093].

**User Agent**

Represents the application the user wishes to interact with and is responsible for providing the request resources to the user and managing the exchange of messages between Identity and Service Provider.

**User credentials**

User credentials refer to the information a user provides to attest his identity. It can go from a simple combination of username and password to the user's fingerprint.

# References


[ALE09] ALERT, ALERT Online. [Online]http://www.alert-online.com/.

[And09] Skolberg Andreas et al., Interoperable SAML 2.0 Web Browser SSO Deployment
Profile. [Online]http://rnd.feide.no/documents/saml2simple.html.

[AOL09] AOL, Webmaster.Info : About Cookies. [Online]http://webmaster.info.aol.com/aboutcookies.html.

[Car09] Geyer Carol, Advantages of SAML | SAML XML.org. [Online]http://saml.xml.org/advantages-saml.

[Coo09] Cookie Central, The Unofficial Cookie FAQ. [Online]http://www.cookiecentral.com/faq.

[Dat09] Datamonitor, Sun Microsystems provides open source software to US Department of
HHS. [Online]2009. http://opensource.cbronline.com/news/sun_microsystems_provides_open_source_software_to_us_department_of_hhs_080409
.

[Dav09] Kearns Dave, Single sign-on is top priority for healthcare execs - Network World. [Online]http://www.networkworld.com/newsletters/netware/2006/1002nw2.html.

[eHe09] eHealthNews.eu, eHealthNews.EU Portal - iSOFT Offers Healthcare Single Sign-on Solutions after Agreement with Sentillion. [Online]http://www.ehealthnews.eu/content/view/1036/26/.

[Eri00] *Meta Access Management system – A Summary for DEST SII Proposals*.

VullingsErik, DalzielJames

[Eve09] Maler Eve, SAMLV2.0-basics.pdf . [Online]http://www.oasis-open.org/committees/download.php/12958/SAMLV2.0-basics.pdf.

[FEI091] FEIDE, simpleSAMLphp | Feide RnD. [Online]http://rnd.feide.no/simplesamlphp.

[FEI092] FEIDE, Single Sign On - Single Log Out. [Online]http://docs.feide.no/fs-0034-1.0-en.html.

[Goo09] Google, SAML Single Sign-On (SSO) Service for Google Apps - Google Apps APIs - Google Code. [Online]http://code.google.com/intl/pt-

PT/apis/apps/sso/saml_reference_implementation.html.

[Gre88] Greenberg Warren, *Competition in the Health Care Sector: Ten Years Later*. s.l., Duke University Press, 1988.

[Häk09] Sagehaug Häkon, Programming Guide SAML_XACML.pdf. [Online]http://www.bccs.uib.no/~hakont/SAMLXACMLExtension/files/ProgrammingGuideSAML_XACML.pdf.

[Hal07] *OASIS XACML Update*. Lockhart Hals.l., OASIS. NAC 2007 Spring Conference.

[Har08] *Dynamic Security Assertion Markup Language: Simplifying Single Sign-On*. Harding P., Johansson L., Klingenstein N.s.l., IEEE, 2008. IEE. pp.83-85.

[IEE09] IEEE, Standards Make Good Business Sense. [Online]http://standards.ieee.org/sa-mem/why_std.html.

[IHE09] IHE, IHE.net Home. [Online]Integrating the Healthcare Enterprise, 2009. http://www.ihe.net/.

[Imp09] Imprivata, Single Sign-On for Healthcare Access Management & Compliance. [Online]http://www.imprivata.com/onesign_solutions_healthcare.

[Int09] Internet2, Home - OpenSAML - Internet2 Wiki. [Online]http://www.opensaml.org/.

[Int091] Shibboleth, TestShib.org. [Online]https://www.testshib.org/.

[Jak05] *Authorization-Authentication Using XACML and SAML*. Wu Jake, Periorellis Panos s.l., University of Newcastle upon Tyne, 2005.

[Jam01] Snell James, Tidwell Doug, Kulchenko Pavel, *Programming Web Services with Soap*. s.l., O'Reilly Media, Inc, 2001.

[jav09] java.netopensso: Home. [Online]https://opensso.dev.java.net/.

[Las09] Lasso, Lasso - Free Liberty Alliance Single Sign On. [Online]http://lasso.entrouvert.org/.

[Lib09] Liberty Alliance, Healthcare / Adoption / Home - Liberty Alliance. [Online]http://www.projectliberty.org/liberty/adoption/healthcare.

[Mar09] Raepple Martin, Getting Started: Security Assertion Markup Language. [Online]https://www.sdn.sap.com/irj/scn/go/portal/prtroot/docs/library/uuid/2a563903-0b01-0010-b9a1-d3875ff74b32.

[Mic09] Microsoft“Geneva” Team Blog : Microsoft “Geneva” Framework. [Online]http://blogs.msdn.com/card/archive/2008/11/04/microsoft-geneva-framework.aspx.

[OAS09] OASIS, sstc-saml-tech-overview-2 0-draft-13.pdf (application/pdf Object). [Online]http://www.oasis-open.org/committees/download.php/22553/sstc-saml-tech-overview-2%200-draft-13.pdf.

[OAS091] OASIS, sstc-saml-tech-overview-2.0-cd-02.pdf (application/pdf Object). [Online]http://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf.

[OAS092] OASIS, History of SAML | SAML XML.org. [Online]http://saml.xml.org/history.

[OAS093] OASIS, draft-sstc-solution-profile-kerberos-03.pdf (application/pdf Object). [Online]http://www.oasis-open.org/committees/download.php/5392/draft-sstc-solution-profile-kerberos-03.pdf.

[OAS094] OASIS, SAMLv20ImplementationDraft01.pdf. [Online]http://xml.coverpages.org/SAMLv20ImplementationDraft01.pdf.

[OAS095] OASIS, access_control-xacml-2.0-saml-profile-spec-os.pdf. [Online]http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-saml-profile-spec-os.pdf.

[OAS096] OASIS, Oasis Security Services Use Cases -- Straw Man Draft 3. [Online]http://www.oasis-open.org/committees/security/docs/draft-sstc-use-strawman-03.html.

[OAS097] OASIS, XACML-ProfileSAML-WD20040819.pdf. [Online]http://xml.coverpages.org/XACML-ProfileSAML-WD20040819.pdf.

[OAS098] OASIS, SAML Assertion XSD. [Online]http://docs.oasis-open.org/security/saml/v2.0/saml-schema-assertion-2.0.xsd.

[OAS099] OASIS, SAML Protocol XSD. [Online]http://docs.oasis-open.org/security/saml/v2.0/saml-schema-protocol-2.0.xsd.

[OAS100] OASIS, OASIS: Advancing open standards for the global information society. [Online]http://www.oasis-open.org/home/index.php.

[OMI09] OMII, OMII EUROPE – What are SAML and XAML?[Online]http://library2.nesc.ed.ac.uk/fedora/get/lib:8713/DS1.

[Pat09] Patterson Pat, Sum Marina, Single Logout: A Demo. [Online]http://developers.sun.com/identity/reference/techart/single-logout.html.

[Pau09] Madsen Paul, XML.com: SAML 2: The Building Blocks of Federated Identity. [Online]http://www.xml.com/pub/a/2005/01/12/saml2.html.

[Pin09] PingIdentity, PingFederate: Internet Single Sign-On, Internet User Account Management, & Identity-Enabled Web Services. [Online]http://www.pingidentity.com/products/pingfederate.cfm

[Pre09] Gralla Preston, How SAML works. [Online]http://searchsoa.techtarget.com/tip/0,289483,sid26_gci818643,00.html.

[Rom09] Pletka Roman, Secure Propagation of Identities Using SAML. [Online]http://www.security-zone.info/download/kongress08/T6/R_Pletka.pdf.

[Sah08] *Architecture of a single sign on (SSO) for internet banking*. Bhosale Sahana K.s.l., IEEE, 2008. IET International Conference. pp.103-105.

[Sah09] Shah Sahid N., Single sign on (SSO) solutions in healthcare | The Healthcare IT Guy. [Online]http://www.healthcareguy.com/index.php/archives/218.

[Sen09] Sentillion, Sentillion: Identity and Access Management for Healthcare. [Online]http://www.sentillion.com/.

[Shi09] Shibboleth, Shibboleth. [Online]http://shibboleth.internet2.edu/.

[Shi091] Shibboleth, SAMLDiffs – Shibboleth 1 Documentation. [Online]https://spaces.internet2.edu/display/SHIB/SAMLDiffs.

[Sin09] Single Logout Protocol. [Online]http://www.parallels.com/r/docs/sso/sso-2.0-guide/index.htm?fileName=55765.htm.

[Sun09] Sun Microsystems, Sun Federated Identity Management. [Online]http://www.sun.com/software/media/flash/demo_federation/index.html.

[Sun091] Sun Microsystems, Sun OpenSSO Enterprise 8.0 Deployment Planning Guide. [Online]https://opensso.dev.java.net/public/use/docs/fampdf/FAMDPG.pdf.

[Sun092] Sun Microsystems, Deployment Example: SAML v2 Using SunOpenSSO Enterprise 8.0. [Online]https://opensso.dev.java.net/public/use/docs/fampdf/FAMDPG.pdf.

[Sun093] Sun Microsystems, Company Info. [Online]http://www.sun.com/aboutsun/company/index.jsp.

[The09] The Times 100, Business Case Studies | BSI | Why are standards important?[Online]http://www.thetimes100.co.uk/case-study--creating-world-class-quality-standards--53-250-3.php.

[TRS09] T R S Cravo, SAMLDiffs - Shibboleth 1 Documentation - Internet2 Wiki. [Online]https://spaces.internet2.edu/display/SHIB/SAMLDiffs.

[Tun09] Namli Tuncay, Dogac Asuman, Chapter8.pdf. [Online]http://www.srdc.metu.edu.tr/webpage/projects/sapphire/publications/Chapter8.pdf.

[Val09] Rosset Valerio, Filippin Cleber V., Westphall Carla M., Distribuição de Direitos para Sistemas DRM Utilizando Padrão de Segurança SAML. [Online]http://inf.unisul.br/~ines/workcomp/cd/pdfs/2862.pdf.

[web09] Webex, WebEx: Federated Authentication Service. [Online]http://developer.webex.com/c/document_library/get_file

?p_l_id=10914&folderId=11421&name=DLFE-201.pdf.

[Wik09] Wikipedia, Integrating the Healthcare Enterprise - Wikipedia, the free encyclopedia. [Online]2009. http://en.wikipedia.org/wiki/Integrating_the_Healthcare_Enterprise.

[Wik091] Wikipedia, Windows Live ID - Wikipedia, the free encyclopedia. [Online]2009. http://en.wikipedia.org/wiki/Windows_Live_ID.

[Wik092] Wikipedia, Standard - Wikipedia, the free encyclopedia. [Online]http://en.wikipedia.org/wiki/Standard.

[Wik093] Wikipedia, FOSS Open Standards/Importance and Benefits of Open Standards - Wikibooks, collection of open-content textbooks. [Online]http://en.wikibooks.org/wiki/FOSS_Open_Standards/Importance_and_Benefits_of_Open_Standards.

[Wik094] Wikipedia, Separation of concerns - Wikipedia, the free encyclopedia. [Online]http://en.wikipedia.org/wiki/Separation_of_concerns.

[Wik095] Wikipedia, Security Assertion Markup Language - Wikipedia, the free encyclopedia. [Online]http://en.wikipedia.org/wiki/SAML.

[Yun07] *Design and Enhance a Dynamic Healthcare Portal Site.* WengYung-Chinget al.s.l., IEEE, 2007. IEEE/WIC/ACM International Conferences. pp.173-176.

# Appendix A: ALERT

ALERT Life Sciences Computing is a Portuguese software house specialized in healthcare solutions.

Created in 1999 under the name "Médicos na Internet" ("*Doctors on the Internet*"), which mainly produced websites for medical and clinical associations, it grew into ALERT on 2003. It currently has a multidisciplinary team of 750 employees.

Nowadays ALERT's mission is to improve health and prolong life, achieve profitability to benefit society, and inspire others to excel like it does.

With headquarters based in Porto, ALERT is spread all over the world. Composed by six branches stationed in Portugal, Spain, United Kingdom, Netherlands as well as in Northern America, Brazil and Singapore, the ALERT group of companies is fully committed to the development, distribution and implementation of ALERT® healthcare solutions, designed to create paper-free clinical environments.

Having won important awards over the past years as the Innovation Prize in 2006 and 2007, COTEC-BPI also in 2007 and the Medal of Honor from the Portuguese Business Association in 2008, ALERT has been rapidly increasing its revenue every year. In 2008 it presented a turnover of over 35 million Euros, mainly resulting of contracts made outside of Portugal.

Its main products comprise a Paper Free Hospital suite which includes solutions for entire hospitals such as for the emergency rooms, operating rooms, outpatient and inpatient departments. It also provides non-clinical solutions as the ALERT® ERP and ALERT® CRM.

ALERT® has already been adopted in Portugal, Spain, Italy, the Netherlands, United Kingdom, Alaska, United States, Brazil as well as Malaysia and has never been uninstalled.

# Appendix B: Gantt Diagram

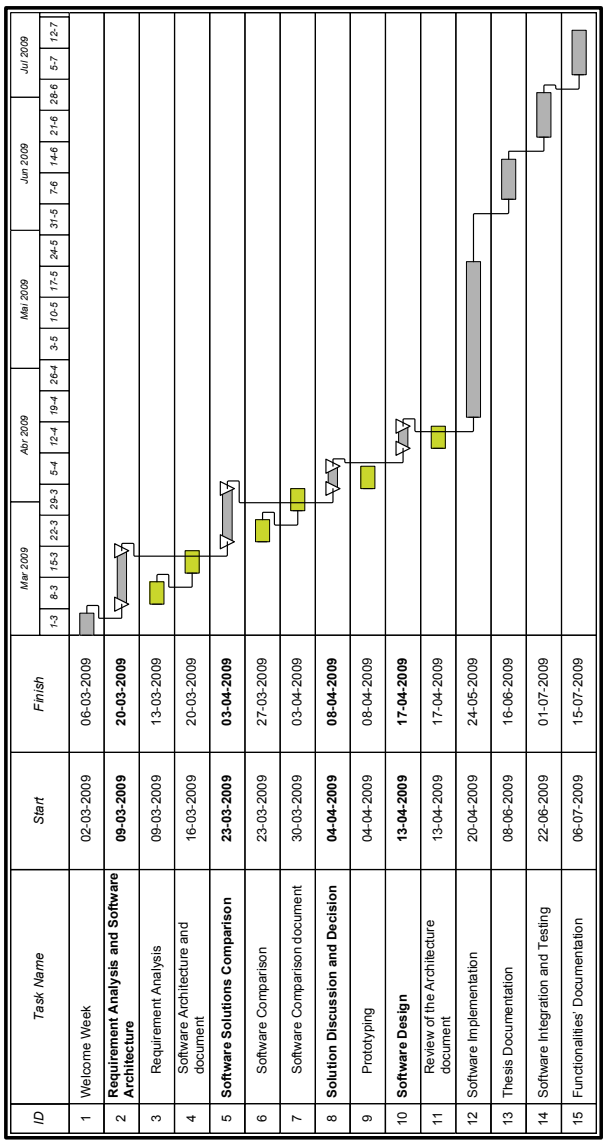


Figure 1: Project's Gantt diagram.

# Appendix C: Browser Cookie

A cookie that contains SSO related information will be used. This cookie will contain information related to the SSO operation, which could range from the authentication method used to particular user attributes. There will also be another cookie containing session data (such as the session ID).

An HTTP Cookie is a parcel of text sent by a server to a Web client (in this case, to the browser), which will be sent back unchanged by the client each time it access that server.

A cookie has six definable attributes:

- Name – name of the cookie,
- Value – value associated with the cookie;
- Expires – the date until when the cookie is valid;
- Path – the subset of directories in a domain for which the cookie is valid;
- Domain – the domain for which the cookie is valid;
- Secure – a Boolean attribute which defines if there must be a secure https connection in order for the cookie to be sent.

For Single Sign On, since the value of the cookie will contain innumerous information and this information should not be easily read, it will be encoded. The domain will be related to the issuer and, since the cookie is only sent through HTTPS connections, the secure attribute shall be set to true.

Cookies present some limitations that will determine their implementation for SSO:

- Cookies cannot be set for domains other than those that the response is originated from;
- Cookies can only be retrieved if they are valid for the document the script resides in.

Therefore, the cookies will only be available to one domain and will not be shared among Identity Provider and Service Providers.

# Appendix D: SAML exchange

### *SSO Request made by the Identity Provider*

```
<samlp:AuthnRequest ID=""s2bc965d244bd77a0ac1963d5d7ba5c6569ad46140""
Version=""2.0"" IssueInstant=""2009-06-17T13:22:15Z""
Destination=""http://www.alert.com:9999/opensso/SSORedirect/metaAlias/
idp"" ForceAuthn=""false"" IsPassive=""false""
ProtocolBinding=""urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact""
AssertionConsumerServiceURL=""http://www.phr.com:2222/opensso/Consumer
/metaAlias/sp"">
    <saml:Issuer>http://www.phr.com:2222/opensso</saml:Issuer>
    <samlp:NameIDPolicy  Format=""urn:oasis:names:tc:SAML:2.0:nameid-
format:transient"" SPNameQualifier=""http://www.phr.com:2222/opensso""
AllowCreate=""true""></samlp:NameIDPolicy>
    <samlp:RequestedAuthnContext Comparison=""exact"">
    <saml:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:P
asswordProtectedTransport</saml:AuthnContextClassRef>
    </samlp:RequestedAuthnContext>
</samlp:AuthnRequest>
```

### *Identity Provider receives the request and sends a response*

```
<samlp:ArtifactResponse
xmlns:samlp=""urn:oasis:names:tc:SAML:2.0:protocol""
ID=""s2b1032277c3a9c7bd9565e6eabe7d4ec0237906d2""
InResponseTo=""s2bb521332e0f0265625ca2917165d52548da6b805""
Version=""2.0"" IssueInstant=""2009-06-17T13:22:27Z""
Destination=""http://www.phr.com:2222/opensso/Consumer/metaAlias/sp"">
    <saml:Issuer
xmlns:saml=""urn:oasis:names:tc:SAML:2.0:assertion"">http://www.alert.
com:9999/opensso</saml:Issuer>
    <samlp:Status xmlns:samlp=""urn:oasis:names:tc:SAML:2.0:protocol"">
        <samlp:StatusCode
xmlns:samlp=""urn:oasis:names:tc:SAML:2.0:protocol""
Value=""urn:oasis:names:tc:SAML:2.0:status:Success""></samlp:StatusCod
e>
    </samlp:Status>
    <samlp:Response
xmlns:samlp=""urn:oasis:names:tc:SAML:2.0:protocol""
```

```
ID=""s2595bb04813abaab9019ef4585b1418587d239947""
InResponseTo=""s2bc965d244bd77a0ac1963d5d7ba5c6569ad46140""
Version=""2.0"" IssueInstant=""2009-06-17T13:22:27Z""
Destination=""http://www.phr.com:2222/opensso/Consumer/metaAlias/sp"">
        <saml:Issuer
xmlns:saml=""urn:oasis:names:tc:SAML:2.0:assertion"">http://www.alert.
com:9999/opensso</saml:Issuer>
        <samlp:Status
xmlns:samlp=""urn:oasis:names:tc:SAML:2.0:protocol"">
            <samlp:StatusCode
xmlns:samlp=""urn:oasis:names:tc:SAML:2.0:protocol""
Value=""urn:oasis:names:tc:SAML:2.0:status:Success""></samlp:StatusCod
e>
        </samlp:Status>
        <saml:Assertion
xmlns:saml=""urn:oasis:names:tc:SAML:2.0:assertion""
ID=""s2e6b57cbd135f16268ac9e17b8ce0eb805074a043"" IssueInstant=""2009-
06-17T13:22:26Z"" Version=""2.0"">

    <saml:Issuer>http://www.alert.com:9999/opensso</saml:Issuer>
            <Signature xmlns=""http://www.w3.org/2000/09/xmldsig#"">
                <SignedInfo>
                    <CanonicalizationMethod
Algorithm=""http://www.w3.org/2001/10/xml-exc-c14n#""/>
                    <SignatureMethod
Algorithm=""http://www.w3.org/2000/09/xmldsig#rsa-sha1""/>
                    <Reference
URI=""#s2e6b57cbd135f16268ac9e17b8ce0eb805074a043"">
                    <Transforms>
                        <Transform
Algorithm=""http://www.w3.org/2000/09/xmldsig#enveloped-signature""/>
                        <Transform
Algorithm=""http://www.w3.org/2001/10/xml-exc-c14n#""/>
                    </Transforms>
                    <DigestMethod
Algorithm=""http://www.w3.org/2000/09/xmldsig#sha1""/>

    <DigestValue>UDyPcmjf6wPYutjG1aPjoN7Gf24=</DigestValue>
                    </Reference>
                </SignedInfo>

    <SignatureValue>X/6BLSxLoFx1CiCsU2HP4cTZN5rDqF3Br3OqxS3VV22sYddmnaM
Nq7DDq2pNuZJktm0rR3KJhzb3INH+lWDAENz2NV/pPKc5UVDNFhwfiuwE9R0EF77a1pTen
2uTjtu8pYGmass2lE35QruU+kTtnDF21jzcQfuW1RQkjFW7NiE=</SignatureValue>
                <KeyInfo>
                    <X509Data>
    <X509Certificate>nMIICQDCCAakCBEeNB0swDQYJKoZIhvcNAQEEBQAwZzELMAkGA
1UEBhMCVVMxEzARBgNVBAgTCkNhnbGlmb3JuaWExFDASBgNVBAcTC1NhbnRhIENsYXJhMQ
wwCgYDVQQKEwNTdW4xEDAOBgNVBAsTB09wZW5TU08xDTALBgNVBAMTBHRlc3QwHhcNMDgw
MTE1MTkxOTM5WhcNMTgwMTEyMTkxOTM5WjBnMQswCQYDVQQGEwJVUzETMBEGA1UECBMKQ2
FsaWZvcm5pYTEUMBIGA1UEBxMLU2FudGEgQ2xhcmExDDAKBgNVBAoTA1N1bjEQMA4GA1UE
CxMHT3BlblNTTzENMAsGA1UEAxMEdGVzdDCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgY
EArSQc/U75GB2AtKhbGS5piiLkmJzqEsp64rDxbMJ+xDrye0EN/q1U5Of+RkDsaN/igkAv
V1cuXEgTL6RlafFPcUX7QxDhZBhsYF9pbwtMzi4A4su9hnxIhURebGEmxKW9qJNYJs0Vo5
+IgjxuEWnjnnVgHTs1+mq5QYTA7E6ZyL8CAwEAATANBgkqhkiG9w0BAQQFAAOBgQB3Pw/U
QzPKTPTYi9upbFXlrAKMwtFf2OW4yvGWWvlcwcNSZJmTJ8ARvVYOMEVNbsT4OFcfu2/PeY
oAdiDAcGy/F2Zuj8XJJpuQRSE6PtQqBuDEHjjmOQJ0rV/r8mO1ZCtHRhpZ5zYRjhRC9eCb
jx9VrFax0JDC/FfwWigmrW0Y0Q==</X509Certificate>
                    </X509Data>
                </KeyInfo>
            </Signature>
            <saml:Subject>
                <saml:NameID
Format=""urn:oasis:names:tc:SAML:2.0:nameid-format:transient""
```

```
NameQualifier=""http://www.alert.com:9999/opensso""
SPNameQualifier=""http://www.phr.com:2222/opensso"">CNhLrwTsxJt1W/7icC
bVOGg7VE80</saml:NameID>
                <saml:SubjectConfirmation
Method=""urn:oasis:names:tc:SAML:2.0:cm:bearer"">
                <saml:SubjectConfirmationData
InResponseTo=""s2bc965d244bd77a0ac1963d5d7ba5c6569ad46140""
NotOnOrAfter=""2009-06-17T13:32:27Z""
Recipient=""http://www.phr.com:2222/opensso/Consumer/metaAlias/sp""/><
/saml:SubjectConfirmation>
            </saml:Subject>
            <saml:Conditions NotBefore=""2009-06-17T13:12:27Z""
NotOnOrAfter=""2009-06-17T13:32:27Z"">
                <saml:AudienceRestriction>

    <saml:Audience>http://www.phr.com:2222/opensso</saml:Audience>
                </saml:AudienceRestriction>
            </saml:Conditions>
            <saml:AuthnStatement AuthnInstant=""2009-06-17T13:22:26Z""
SessionIndex=""s20c67847be65a7bbf15622ef3aba3050b8d349101"">
                <saml:AuthnContext>

    <saml:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:P
asswordProtectedTransport</saml:AuthnContextClassRef>
                </saml:AuthnContext>
            </saml:AuthnStatement>
            <saml:AttributeStatement>
                <saml:Attribute Name=""userid"">
                    <saml:AttributeValue
xmlns:xs=""http://www.w3.org/2001/XMLSchema""
xmlns:xsi=""http://www.w3.org/2001/XMLSchema-instance""
xsi:type=""xs:string"">1</saml:AttributeValue>
                </saml:Attribute>
            </saml:AttributeStatement>
        </saml:Assertion>
    </samlp:Response>
</samlp:ArtifactResponse>
```

### *Service Provider receives and extracts the response*

```
<samlp:Response ID=""s2595bb04813abaab9019ef4585b1418587d239947""
InResponseTo=""s2bc965d244bd77a0ac1963d5d7ba5c6569ad46140""
Version=""2.0"" IssueInstant=""2009-06-17T13:22:27Z""
Destination=""http://www.phr.com:2222/opensso/Consumer/metaAlias/sp"">
    <saml:Issuer>http://www.alert.com:9999/opensso</saml:Issuer>
    <samlp:Status>
        <samlp:StatusCode
Value=""urn:oasis:names:tc:SAML:2.0:status:Success""></samlp:StatusCod
e>
    </samlp:Status>
    <saml:Assertion
xmlns:saml=""urn:oasis:names:tc:SAML:2.0:assertion""
ID=""s2e6b57cbd135f16268ac9e17b8ce0eb805074a043"" IssueInstant=""2009-
06-17T13:22:26Z"" Version=""2.0"">
        <saml:Issuer>http://www.alert.com:9999/opensso</saml:Issuer>
        <Signature xmlns=""http://www.w3.org/2000/09/xmldsig#"">
            <SignedInfo>
                <CanonicalizationMethod
Algorithm=""http://www.w3.org/2001/10/xml-exc-c14n#""/>
                <SignatureMethod
Algorithm=""http://www.w3.org/2000/09/xmldsig#rsa-sha1""/>
                <Reference
URI=""#s2e6b57cbd135f16268ac9e17b8ce0eb805074a043"">
```

```
                    <Transforms>
                        <Transform
Algorithm=""http://www.w3.org/2000/09/xmldsig#enveloped-signature""/>
                        <Transform
Algorithm=""http://www.w3.org/2001/10/xml-exc-c14n#""/>
                    </Transforms>
                    <DigestMethod
Algorithm=""http://www.w3.org/2000/09/xmldsig#sha1""/>

    <DigestValue>UDyPcmjf6wPYutjG1aPjoN7Gf24=</DigestValue>
                </Reference>
            </SignedInfo>

    <SignatureValue>X/6BLSxLoFx1CiCsU2HP4cTZN5rDqF3Br3OqxS3VV22sYddmnaM
Nq7DDq2pNuZJktm0rR3KJhzb3INH+lWDAENz2NV/pPKc5UVDNFhwfiuwE9R0EF77a1pTen
2uTjtu8pYGmass2lE35QruU+kTtnDF21jzcQfuW1RQkjFW7NiE=</SignatureValue>
            <KeyInfo>
                <X509Data>

    <X509Certificate>MIICQDCCAakCBEeNB0swDQYJKoZIhvcNAQEEBQAwZzELMAkGA1
UEBhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWExFDASBgNVBAcTC1NhbnRhIENsYXJhMQww
CgYDVQQKEwNTdW4xEDAOBgNVBAsTB09wZW5TU08xDTALBgNVBAMTBHRlc3QwHhcNMDgwMT
E1MTkxOTM5WhcNMTgwMTEyMTkxOTM5WjBnMQswCQYDVQQGEwJVUzETMBEGA1UECBMKQ2Fs
aWZvcm5pYTEUMBIGA1UEBxMLU2FudGEgQ2xhcmExDDAKBgNVBAoTA1N1bjEQMA4GA1UECx
MHT3BlblNTTzENMAsGA1UEAxMEdGVzdDCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEA
rSQc/U75GB2AtKhbGS5piiLkmJzqEsp64rDxbMJ+xDrye0EN/q1U5Of+RkDsaN/igkAvV1
cuXEgTL6RlafFPcUX7QxDhZBhsYF9pbwtMzi4A4su9hnxIhURebGEmxKW9qJNYJs0Vo5+I
gjxuEWnjnnVgHTs1+mq5QYTA7E6ZyL8CAwEAATANBgkqhkiG9w0BAQQFAAOBgQB3Pw/UQz
PKTPTYi9upbFXlrAKMwtFf2OW4yvGWWvlcwcNSZJmTJ8ARvVYOMEVNbsT4OFcfu2/PeYoA
diDAcGy/F2Zuj8XJJpuQRSE6PtQqBuDEHjjmOQJ0rV/r8mO1ZCtHRhpZ5zYRjhRC9eCbjx
9VrFax0JDC/FfwWigmrW0Y0Q==</X509Certificate>
                </X509Data>
            </KeyInfo>
        </Signature>
        <saml:Subject>
            <saml:NameID Format=""urn:oasis:names:tc:SAML:2.0:nameid-
format:transient"" NameQualifier=""http://www.alert.com:9999/opensso""
SPNameQualifier=""http://www.phr.com:2222/opensso"">CNhLrwTsxJt1W/7icC
bVOGg7VE80</saml:NameID>
            <saml:SubjectConfirmation
Method=""urn:oasis:names:tc:SAML:2.0:cm:bearer"">
                <saml:SubjectConfirmationData
InResponseTo=""s2bc965d244bd77a0ac1963d5d7ba5c6569ad46140""
NotOnOrAfter=""2009-06-17T13:32:27Z""
Recipient=""http://www.phr.com:2222/opensso/Consumer/metaAlias/sp""/>
            </saml:SubjectConfirmation>
        </saml:Subject>
        <saml:Conditions NotBefore=""2009-06-17T13:12:27Z""
NotOnOrAfter=""2009-06-17T13:32:27Z"">
            <saml:AudienceRestriction>

    <saml:Audience>http://www.phr.com:2222/opensso</saml:Audience>
            </saml:AudienceRestriction>
        </saml:Conditions>
        <saml:AuthnStatement AuthnInstant=""2009-06-17T13:22:26Z""
SessionIndex=""s20c67847be65a7bbf15622ef3aba3050b8d349101"">
            <saml:AuthnContext>

    <saml:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:P
asswordProtectedTransport</saml:AuthnContextClassRef>
            </saml:AuthnContext>
        </saml:AuthnStatement>
        <saml:AttributeStatement>
            <saml:Attribute Name=""userid"">
```

```
                <saml:AttributeValue
xmlns:xs=""http://www.w3.org/2001/XMLSchema""
xmlns:xsi=""http://www.w3.org/2001/XMLSchema-instance""
xsi:type=""xs:string"">1</saml:AttributeValue>
            </saml:Attribute>
        </saml:AttributeStatement>
    </saml:Assertion>
</samlp:Response>
```