MACQUARIE UNIVERSITY

INVESTIGATING LIBERTY ALLIANCE AND SHIBBOLETH INTEGRATION

ITEC809 – FINAL REPORT

Nishen Naidoo
30396468
nishen.naidoo@mq.edu.au
Supervisor: Steve Cassidy (steve.cassidy@mq.edu.au)

ABSTRACT

The growth of online activity has brought with it several challenges in the arenas of privacy, security and identity management. Federated Identity Architectures aim to mitigate the risks, costs and complexities within each of these arenas. Domain specific requirements have evolved two frameworks to address these issues for their respective domains. Liberty Alliance is a project and a Federated Identity Framework that specifically targets requirements within the commercial sector, while Shibboleth was founded with the intent of providing resource sharing for higher education.

Users, who are often members of both domains, are still left with multiple identities to manage across these federations, some of which exist only due to their technology limitations. These identities are exposed to multiple identity providers, limiting their privacy.

This project investigates these two architectures, identifying their profiles, protocols and bindings and establishes what would be required for their core components, the Service Provider and the Identity Provider, to communicate across different frameworks.

We show how having these frameworks based upon the same parent specification provides common communication patterns that can be leveraged to provide desired functionality through the addition of message translation at strategic locations within the architectures.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

The growth of online activity has brought with it several challenges in the arenas of privacy, security and identity management. Federated Identity Architectures aim to mitigate the risks, costs and complexities within each of these arenas.

The 'silo' model for identity management is where each service provider maintains identity information for each of their users. This model is by far the most prevalent due to its lack of complexity. From the service provider perspective, this is the simplest, most cost effective model to employ. From the end user perspective, this is just another of the 25 user accounts they have for which they need to remember a username and password.

Apart from the privacy concerns with such a model, where user information is spread across the internet, there are also security concerns to bear in mind. Typically, a user does not use different usernames and passwords for each of these services. If they can, they will use the same credentials across any service they can get away with using (sometimes there maybe username conflicts). This poses a security risk, as credentials that are compromised at one location can potentially compromise all locations and not all service providers in this model would have the same level of security employed to safeguard these credentials.

There is a slow shift towards Federated Identity Architectures to aid in addressing some of these issues. One of the founding principles of Federated Identity Management is the issue of privacy. Privacy is tightly controlled by the user by giving them the ability to choose what information they are willing to release to a service provider. In addition, service providers do not store authentication information for their users. Instead, the service provider has to contact a user's identity provider and have the user authenticate there. If a service provider is to allow their users to be authenticated at a third party there has to be some sort of trust between the service provider and the third party, or identity provider. This trust relationship is established between service providers and identity providers and is the basis of forming a federation. Within a federation, service providers and identity providers trust one another for the purposes of authenticating users and asserting various types of information about them. As privacy is a primary concern, an identity provider will only assert information that they are permitted to by the user.

To explain with an example, let us examine 'John' who is a student at 'Alpha Tech University' (ATU). ATU is John's Identity Provider (IdP). ATU has a trust relationship with 'Random House Publishers' (RHP) to allow their postgraduate students access to certain academic publications. RHP has similar trust relationships with many universities and other organisations around the world. When John attempts to access a resource at RHP over the web, he is prompted to select his IdP. He is then forwarded to the IdP (which in this instance would be ATU), where he is challenged for his credentials. On providing his credentials he is authenticated, has an authentication token added to his web session (stating where and when he was authenticated and a transparent ID) and returned to where the transaction originated – RHP. On returning, RHP is able to identify that John has in fact been authenticated at ATU, even though RHP has no idea about who John is. Using the transparent ID (to preserve John's privacy/anonymity), RHP can send a request to ATU asking for specific attributes about the user. Provided John is amenable to the requested attributes being divulged, ATU can release these attributes to RHP (e.g. an attribute stating whether John is a postgraduate student or an undergraduate student). Based on the attributes returned, RHP can then make an authorisation decision as to whether to allow John access to the resource he requested. In this case, as John is a postgraduate student, he is granted access and provided with the full text of the publication (Cantor 2005).

ATU, RHP and the other connected organisations form a federation.

From the end user perspective, services within a federation would provide a welcome relief from the numerous credentials they are required to maintain. This way, they have to manage just a single set of credentials per federation, which is hopefully stored in a highly secured location. Each set of credentials allows access to services across that federation using single sign on (i.e. only ever be prompted to authenticate the first time using a federation service for a session) and ensure that each service provider has the bare minimum amount of information about them that it requires to provide the service to them.

A Federation is formed through policy and process but is supported by various technology specifications that form the framework for delivering federation services. Two popular federation framework technologies are currently gaining traction. These are Liberty Alliance and Shibboleth.

From the 'silo' model of one set of credentials per service, we've moved to one set per federation. With the advent of multiple frameworks, we add new credentials for federations of differing technologies. While this is understandable, it is not ideal, as users still have their identities scattered across multiple federations, some of which would exist purely due to technology constraints. Why should a federation not be able to extend beyond technology boundaries imposed by the framework that it employs?

This project intends to investigate exactly that question.

To begin, we will introduce and examine Federated Identity Management (FIM) and attempt to provide a baseline understanding of its purpose. We will do so by following the evolutionary stages of online identity that has led to the concepts and designs that are core to these architectures. We will also include a basic analysis for the reasoning of some of these design decisions and introduce privacy as a primary concern.

As part of the discussion we will also introduce both the Liberty Alliance and Shibboleth Federated Identity Architectures and provide high level overviews of each. We examine the role of privacy within these architectures and how the various privacy requirements are addressed or omitted. In addition, we introduce a set of criteria for measuring Federated Identity Architectures against which we assess both Shibboleth and Liberty.

Following this, we delve more deeply into each of these architectures in turn. We examine each of their specifications to gain an understanding of how they work, what their differences are and what it would take to extend a single federation to encompass both technologies. We examine the profiles, protocols, bindings and assertions that are used within each framework.

Once we have analysed both these frameworks to a sufficient degree, we identify their differences and similarities and establish what it would take to integrate them. We identify some of the limitations imposed by the framework that hinder integration and some of the commonly shared specifications that enable some components to work together with little or no adapting. We conclude with the recommendations of how cross framework component communication could occur with the aid of message translation.

2 BACKGROUND

Evolutionary requirements in the identity domain have spawned new issues that modern management architectures attempt to resolve. One of the biggest issues regards privacy. We will attempt to analyse how two different architectures handle the issues of privacy.

A core component in the interaction of business entities online is the digital identity. While it is possible in a face-to-face transaction to establish a person's identity by using their physical credentials (credit card, driver's licence etc.), the equivalent online process tends to be much more complex and contains a significant 'trust' requirement. In addition to the business drivers, there is an increasing requirement for a digital identity to facilitate online collaboration and personal preferences. This is resulting in the merging of profile information into the concept of the digital identity [Eap 2007].

As more businesses begin operating online, it is becoming necessary for end users to maintain multiple identities at multiple locations. It is often the case that a user manages a digital identity for each service that they use. This means that each service provider (SP) used maintains a copy of a set of credentials and profile information for a particular user. Bhargav-Spantzel [2006] discusses this model and explains it as the 'silo' model and states that it is the most common form of identity management today. He argues that this form of management is cumbersome for the end user, although its predominant position is due to it being the simplest model for managing identity. This system has duplicated costs associated with it for SPs with regards to managing user data. There are also inconveniences for users with such a system. Being simple, however, also means that there are many growing requirements that are not, and cannot, be met. Bhargav-Spantzel [2006] is supported by Dhamija and Dusseault [2008] with regards to end user experience in the management of identity and states that 'password fatigue' is becoming more of an issue with the average user having approximately twenty five accounts and typing in eight passwords a day. This could quickly become unmanageable with the growth of online activity.

As the requirements for identities and identity management change with the growth of online activity, management techniques have to evolve with them. The next phase after the 'silo', according to Bhargav-Spantzel [2006], was centralised identity provisioning where a single identity provider (IdP) supported multiple online services. Such services as Microsoft Passport and a host of OpenID providers are examples of this form of centralised identity provisioning. While this model does improve the management of identities for the end user by shifting the responsibility of the management to a single third party, it does introduce a single point of failure, as well as issues of trust where the service provider does not trust the identity provider and vice versa.

El Maliki [2007] has highlighted the issue of these centralised models tending to allow the use of phishing attacks by untrusted service providers to trick users into exposing their details. There is no trust relationship between a SP and an IdP in this model which might make a system like OpenID a good system for establishing identity (i.e. you are the person that you say you are) but not an ideal solution, for example, for reducing the ability of spammers to obtain your information. It is not enough to just authenticate a user, but the IdP and SP as well to prevent such types of phishing attacks and by establishing a certain level of trust between the parties of online transactions [Dhamija and Dusseault 2008].

Another issue with such a centralised model is that the identifier that you use from your IdP tends to be common across all the SPs that use it. In this way it is possible for a third party to identify that a person from a particular SP is also the same person at other SPs. This information could allow a person to be tracked across multiple SPs and be used to create a profile of them and their habits.

Privacy in identity management is no longer just a desirable system requirement; it is also a legal requirement. Hansen et al. [2008] has discussed the increased pressure in countries such as the United States, Canada and the European Union in mandating specific guidelines for organisations dealing with identity data. While the United States is still behind the European Union and Canada in data protection legislation, even they have guidelines in place for financial and medical information. This point of view is also supported by Peyton et al. [2007] and applies pressure for workable solutions to be found in this space.

The next phase in evolution of identity management uses a Federated Identity Architecture (FIA) and attempts to address some of these outstanding issues.

> *A Federated Identity Architecture (FIA) is a group of organisations that have built trust relationships among each other in order to exchange digital identity information in a safe way, preserving the integrity and confidentiality (privacy) of the user personal information. The FIA basically involves Identity Providers (IdP) and Service Providers in a structure of trust by means of secured communication channels and business agreements.* [Fragoso-Rodriguez et al. 2006]

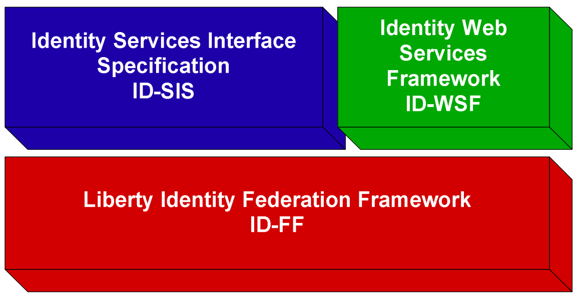
Fragoso-Rodriguez et al. [2006] succinctly describes the purpose of FIA and states that there are currently three such architectures which attempt to solve the issues of identity management in a federated way, with each having their strengths and weaknesses. In order to maintain the principle of privacy in these federations, we can attempt to examine the different architectures and identify any weaknesses. To this purpose we will examine two FIA's, Shibboleth and the Liberty Alliance Project.

2.1 LIBERTY ALLIANCE PROJECT

The Liberty Alliance Project (Liberty) is a conglomeration of over 150 different organisations that have come together to establish some best practices and open standards in the pursuit of identity management solutions. Shim [2005] explains that the goal of Liberty is to create an infrastructure that can support both current and emerging network access devices. To attain this goal Liberty is based on open standards such as XML, SOAP and SAML and is comprised of three core specifications, the Identity Federation Framework (ID-FF), the Identity Web Services Framework (ID-WSF) and the Identity Services Interface Specification (ID-SIS) [Shim 2005].

The ID-FF introduces the foundation of Liberty, the Circle of Trust (CoT). The CoT is the federation of IdPs and SPs that are bound together by a set of technical and business policies to establish the foundation for identity management.

FIGURE 1 - LIBERTY SPECIFICATIONS


Identity Services Interface Specification ID-SIS
Identity Web Services Framework ID-WSF
Liberty Identity Federation Framework ID-FF

Although the specifications and the architecture overview do not isolate any single sector as a target for using Liberty, both Liberty Project Contributors [2003] and Fragoso-Rodriguez et al. [2006] acknowledge that many of the assumptions upon which Liberty is based are actually assumptions that originate from the business sector. Under these assumptions there are some limitations to the level of privacy a system can and should accommodate. For example, an anonymous identifier would be insufficient between an airline system and a car booking system for a person that wishes to reserve a car. The car rental agency would need to have some persistent method of identifying the user and possibly their driver's licence information.

Wason [2003] states that the functional requirements for Liberty are:

1. Identity Federation.
   This set of requirements stipulates that SPs and IdPs communicate to the user and to each other when federating and de-federating accounts, terminating accounts and allow a user to view their federated accounts. This also stipulates that SPs may also use anonymous temporary identities for a user.
2. Authentication.
   This section outlines the various requirements for communication between SPs and IdPs in establishing the identity of a user and stipulates that the communication of user data has to be done with regards to confidentiality, integrity and authenticity. In these scenarios, the IdPs and the SPs need to be authenticated as well as the user.
3. Use of Pseudonyms.
   The Liberty framework supports the user of pseudonyms which need to be unique across a particular CoT. Pseudonyms can be either temporary or permanent. They allow a user to maintain a certain level of privacy.
4. Support for Anonymity.
   This ties in with 3. A SP can request that an IdP provide a temporary pseudonym for a user rather than any permanent identifier in order to maintain user anonymity. This pseudonym can then be used to obtain information for or about the user pending their consent.
5. Global logout.
   This requirement states that should a user logout from an IdP, he should be logged out from all SPs.

The functional requirements of the system tie in closely with the Fair Information Practice (FIP) principles that were first developed in the 1970's: openness, individual participation, collection limitation, data quality, finality, security, accountability [Hansen et al. 2008].

The principle of openness states that the systems containing personal data should be publicly known along with the system's purpose. Under the Liberty functional requirements of Identity federation, users are permitted to view all systems to which they are federated well as the ability to de-federate themselves from any IdP or SP.

Individual participation suggests that users should have the right and ability to manage any data about them that is held. There is nothing mentioned in the requirement specification for the ability of users to manage their own data. This kind of requirement ultimately rests with the IdP and perhaps rightfully so. It would not, for example, be appropriate for a user to modify their driver's license details without going through a proper verification process which in such a case might involve a face-to-face component.

Collection limitation simply states that data should only be collected with the consent of the user. Once again, this is more the realm of the IdP and up to them to manage such a process and, as such, does not impact on the architecture of Liberty. The same applies to the principle of data quality which advocates that the data collected be relevant to the purposes for which it is being used.

Finality suggests that personal data disclosure should be limited for the use under which it was collected in the first place. Although Fragoso-Rodriguez et al. [2006] states that Liberty does not allow the user to control the release of information about themselves to SPs, new specifications of Liberty have since rectified this issue. Users are now able to control how and what information is disclosed to service providers adding a critical layer of functionality with regards to privacy.

The principle of security is one which Liberty holds in the highest regard. This principle says that user data should be reasonably protected against loss and unauthorised access, destruction, use, modification and disclosure. The Liberty architecture mandates secure, authenticated communication between all parties involved in a transaction. While data protection typically falls under the mantle of business continuity, the authentication measures and secure channels (using SSL and PKI) for all communication mitigates the risk of unauthorised activity.

The seventh principle, accountability, states that keepers of personal data should be accountable for complying with fair information practices. As stated above, the legal requirements are enforcing accountability in identity management systems. Without legislation in this area, it is difficult to enforce accountability. While the United States is still behind Canada and the European Union on many privacy laws, they are swiftly catching up as evidenced by the health and financial sectors [Peyton 2007].

In addition to the FIP principles, Hansen et al. [2008], devised three additional principles that they felt were important. These were added to account for the vast changes in digital information management since the 1970's.

The first of these additional principles is diversity and decentralisation. They felt that authentication options should be "like keys on a key ring", where a user can select an appropriate key to access a particular service. They also felt that identity information should not be centralised although they do not provide any justification for such a statement. Within the Liberty architecture, identity can be either centralised or decentralised. Pieces of identity can be, but does not have to be, kept at a single identity provider within a circle of trust.

However, as stated before, the requirements of a service in the business sector tend to require pieces of user information stored locally such as a local account in the service system in order to facilitate business processes. It tends to be rare that no local account information is required to be stored in such instances and, as such, the identity in Liberty is usually decentralised. There are many arguments for and against centralisation and it is difficult to go one way or another unless a context is provided. As stated, the Liberty model is more suited to a decentralised identity and architecture such as Shibboleth works best with a more centralised model. It should be noted however, that both architectures can use both centralised and decentralised models.

The second principle is proportionality. Simply stated, it suggests that a system should only collect information that it actually requires. This sounds very similar to the data quality and collection limitation principles but is possibly reiterated as the ability to collect and store vast amounts of user information has become trivial.

And the final principle, that of 'privacy by design', is one that is a cornerstone of Liberty. It is difficult to dispute that the architecture was created with privacy and security in mind. A potential problem in the architecture was observed by Peyton et al. [2007] who were attempting to address privacy issues for the E-Health system in Canada and whether the Liberty system would in fact comply with the Canadian privacy legislation, Personal Health Information Privacy Act (PHIPA).

Peyton et al. [2007] identified three principles that their identity management system had to adhere to for legislative compliance:

1. Organisations must identify how they intend to use personal data and receive content from the individual.
2. Organisations must establish internal procedures to document and safeguard their use of data.
3. Individuals must be given access to their data and have recourse to challenge its accuracy and use.

They made the observation that although members within the CoT were in fact trusted, it was not possible for a user to obtain details on how their identity information was being used. They proposed an addition to the Liberty architecture – an audit trail. The audit trail would be able to monitor the use of a user's data, thus allowing them to see how their data was being employed and allow challenges to its use. It also would allow third party certification of the first and second principles.

2.2 SHIBBOLETH

The Shibboleth framework is a product of the Internet2 Middleware Initiative which is a consortium of over 200 universities in the United States working in cooperation with over 100 corporations and government agencies. Unlike the Liberty Alliance Project, Shibboleth is specifically targeted to universities and solving their identity issues with regards to collaborative sharing of resources [Fragoso-Rodriguez et al. 2006][Needleman 2004].

The Shibboleth architecture is simpler in comparison to the Liberty architecture. Part of the reason for this is that while Liberty supports both browser based and web services based applications, Shibboleth is currently restricted to browser based applications only (Liberty Alliance Project 2003). This simplifies the requirements for the architecture and positions it to suitably address the domain for which it was designed – institutional resource sharing. Another factor that makes the Shibboleth architecture simpler is that the IdP is centralised. A user typically has a 'home' IdP (which might be their academic institution).

Similar to the Liberty Circle of Trust is the Shibboleth concept of the 'club', even though members of one club can use resources of another club (Liberty Alliance Project 2003). This federation is comprised of similar components to that of Liberty. However, unlike Liberty, there is a logical distinction between an IdP and any SPs in the framework. It should also be noted that the interaction between an IdP and a SP with regards to the exchange of attributes, has to be founded on a common vocabulary. It therefore seems necessary for Shibboleth clubs to conform to a particular schema for communication. This is covered in more detail in following sections.

Let us compare the Shibboleth architecture against the ten fair information practice principles laid out by Hansen et al. [2008].

Given the centralised nature of Shibboleth, openness is an easy principle to comply with. There is generally a single store of user information.

The principle of individual participation is difficult to gauge as Shibboleth does not place any restrictions or specifications on how user data is managed at an IdP. To allow a user to view all their details as well as manage their details is a decision left up to individual IdPs. Typically in a university institution, much of the available information about a user is not modifiable by said user.

Collection limitation again is a principle that is linked to the management of a particular IdP and as such not governed by any Shibboleth requirements or specifications. The same can be applied to the principle of data quality.

The principle of finality is one that Shibboleth strictly follows. Personal data is disclosed only with user consent. The user has complete control over who sees what data about them through the use of attribute release policies.

The principle of security is a shared domain between both the Shibboleth architecture and the IdP. Each is responsible in part for ensuring that the data is secure against loss, and unauthorised activity. All communication between parties in the Shibboleth architecture is done over secure channels using Secure HTTP and PKI. All parties require valid certificates and trust policies need to be in place within federations (or clubs).

While the principle of accountability should pervade, the fallout from Shibboleth within the education sector is perhaps not as large as could occur within Liberty in the business sector. While United States legislation in the area currently focuses more on health and finance than on education [Peyton 2007], the trend would indicate that it will not be long before legislative accountability would apply universally.

Diversity and decentralisation is perhaps a principle that is ill-suited to architecture such as Shibboleth. Given the domain that the architecture targets, the assumption upon which it was founded and the way the universities manage users, it would seem that centralisation is the better option.

Proportionality again has similar connotations to the principles of collection limitation and data quality and also falls under the domain of the IdP.

Finally we have privacy by design. Shibboleth was designed specifically around the concept of privacy. The protocol interaction maintains an even higher level of privacy than in the Liberty architecture and there are no requirements at SPs for local accounts.

2.3 ANALYSIS

Evaluating both Shibboleth and Liberty against a set of design principles for good identity management has highlighted two key points. The first is that the current solutions cater for particular domains, that is, the education sector and the business sector. The design decisions for each of the architectures are greatly influenced by assumptions that are specific to each domain. The second point is that while these architectures provide specifications for secure, private communication between entities within the architecture, they fail to provide specifications or guidelines for the privacy and management of data within the entities themselves. That is, they fail to provide adequate principles and guidelines for Identity and Service providers for the secure management of data residing on their systems. The lack of these guidelines or specifications means that the Federated Identity Architectures of both these systems is incomplete.

Identity management has progressed significantly with regards to privacy and manageability since the Bhargav-Spantzel [2006] silo model. With multiple models evolving and targeting specific use cases/sectors, we need to be careful not to create the same problem by simply replacing silos of simple authentication systems with silos of complex federated identity management systems. In this respect there is still a long way to go.

Identity management architectures need to evolve towards either a single architecture, or architectures that can seamlessly interoperate with each other to sustain any sort of long term viability. As a user, I should not have to have one identity for me being a student, and one for me to use as a consumer. This path would lead right back to where we currently are; soon, I might need another identity for use with my hobby services and another for news and blogging services. Each of these are merely aspects of a single identity and we need to move towards systems that treats it as such.

Dhamija and Dusseault [2008] states that Identity Management is not a goal in itself. Users are not overly concerned with the details of managing their identities so long as it is simple to do and they are assured of their privacy and that it is safe and secure. Currently, one of the largest barriers to entry for these FIAs is the level of complexity involved in setting up and managing such a system. Such complexity has significant costs associated with it. While Shibboleth, which targets the education sector, might not feel such an impact as universities tend to be resourced enough to handle that level of complexity, it would more than likely be a deterrent for small and medium sized businesses [Poon and Swatman 1999] where growth is strongest in the online domain.

To conclude, the approaches we are seeing in the way privacy and security is handled is significantly improved over earlier models. However, we need to see simpler systems for both users and service providers that still adhere to the principles of good identity management. These systems need to cover the whole identity management process and not just the communication between parties.

3 FEDERATION FRAMEWORKS

This section outlines the heritage and common components of the two Federated Identity Management frameworks we are dealing with. The first is Shibboleth, an education sector focused framework initially established to facilitate resource sharing amongst higher education institutions without the overhead of cross-institutional identity management.

The second framework is Liberty Alliance, which has more of a commercial sector focus. It was founded with the aim of providing an open and secure standard for decentralised authentication and Single Sign On.

Both of these frameworks sought to place the control of personal information into the hands of the user, allowing them to decide which services see specific information.

Both frameworks are remarkably similar and the reason is that they both are based on the Security Assertion Markup Language (SAML) specification. SAML is an open specification put forward by the Organisation for the Advancement of Structured Information Standards (OASIS), which is also responsible for several other well known specifications in this domain such as SOAP, WSDL, XACML and many more.

3.1 SAML

The SAML specification [Cantor et al. 2005a] was based on two prior security initiatives. These were the Security Services Markup Language (S2ML) and the Authorisation Markup Language (AuthXML). The SAML specification is designed to allow interchanging of security assertion information about principals (users or systems) between interested domains (typically the Service Provider and the Identity Provider). It is an open standard that was introduced as a mechanism to facilitate single sign on and aid in identity management without having to resort to the multitude of proprietary solutions that were currently available but which typically resulted in interoperability issues [Steel et al. 2005].

The core of SAML comprises of a set of schemas that are used for describing security assertions as well as a set of recommended protocols and profiles which are used to facilitate the transmission of these assertions between parties.

In order for one party to communicate with another party, SAML requires that both parties share a certain set of metadata about each other. This metadata includes the relevant URL endpoints for communication, the identifiers of each provider, their public keys and certificates for enabling secure communication and various other bits and pieces of information. The schema describing the information that is required to be shared is covered in the SAML Metadata specification [Cantor et al. 2005b]. This sharing of metadata is the trust foundation upon which both frameworks that we are discussing is based. These form the basis of the Shibboleth "Club" and the Liberty Alliance "Circle of Trust". Although we are discussing the technical requirements of integration, it is important to note that in order for federations to be established there are many non-trivial policy and process requirements that need to be addressed between the organisations involved in forming these federations.

3.2 COMMON COMPONENTS

As both frameworks are strongly based on the SAML profiles, there are many aspects that they have in common. With regards to components, both frameworks share the concept of the Service Provider (SP) and the Identity Provider (IdP). While the high level conceptualisation of these components is the same for both frameworks, their implementations do differ. These will be covered in more detail in specific framework sections.

3.2.1 IDENTITY PROVIDER

The IdP is the source of identity information for a principal (person or system). This information is typically a set of credentials and attributes for a principal that is asserted by the IdP. A Service Provider (SP) has to trust that the identities asserted by the IdP are correct and legitimate. This type of trust is fundamental to any Federated Identity system.

Some of the activities that the IdP is responsible for include authenticating users, providing assertion statements for SPs and managing Single Sign On (SSO) states for users logged onto multiple services.

3.2.2 SERVICE PROVIDER

The purpose of the Service Provider (SP) is to secure access to resources and services. Accesses by a principal to resources/services that are protected are governed by the interaction between the SP and IdP. The SP is able to make requests to an IdP for both attributes and security assertions.

The SP is required to trust that the statements the IdP issues are correct and legitimate. As stated above, the metadata stored and configured by all SPs and IdPs within a federation form the basis of this trust.

3.2.3 THE USER AGENT

The User Agent is the entity requesting resources and to whom an identity pertains. Typically, the User Agent is the web browser but can be an application or system performing actions on its own behalf. The User Agent authenticates at an IdP in order to request resources at a SP.

4 SHIBBOLETH

This section intends to establish the Shibboleth architecture by examining the components of which it is comprised, the profiles that regulate the component inter-communication and the protocols that these profiles use [Cantor 2005c] [Scavo and Cantor 2005].

The Shibboleth specification was introduced by the Internet2 Middleware Initiative as a solution to provide federated resource access and single sign on capabilities across the higher education domain and is designed to work interactively with a user and a web browser. There have been some attempts to migrate Shibboleth functionality to the desktop [Spence et al. 2006] but these variations are out of scope for this project, which is concerned primarily with the User/Browser interaction with a federation.

4.1 COMPONENTS

Apart from the User Agent, there are three components comprising the Shibboleth architecture. These are the Identity Provider (IdP), the Service Provider (SP) and the Where Are You From (WAYF). The WAYF is an optional component in the architecture. Almost all communication between components occur using SAML.

4.1.1 IDENTITY PROVIDER

The Identity Provider (IdP) component is the first of the common components to both frameworks. Within Shibboleth, this component is broken down into various logical components, each responsible for specific tasks. The rest of this section describes each of these logical subcomponents and how they contribute to the component as a whole.

4.1.1.1 AUTHENTICATION AUTHORITY

The authentication authority is tied to the authentication service of the IdP. The form of authentication used is not relevant to the specification as any form of authentication is allowed - the most common being the username and password. Once a user is authenticated at the IdP, the authentication authority can issue authentication statements regarding that user to other components. There is a vocabulary specification for describing how a user was authenticated (e.g. whether by username/password, certificate, biometric etc.). This value is relevant for assertions as when sent to a SP, the SP is able to make a decision as to whether the form of authentication was sufficiently strong. Some services, such as a bank, might require stronger forms of authentication than just a username and password.

4.1.1.2 SINGLE SIGN ON SERVICE

The single sign on service is typically the first point of contact to the IdP and is responsible for initiating the authentication process. It is also responsible for creating required authentication assertions by interacting with the authentication authority and wrapping these responses in the appropriate HTTP responses that conform to the appropriate profile that is being used.

4.1.1.3 ARTIFACT RESOLUTION SERVICE

For certain profiles (e.g. Browser/Artifact), authentication assertions are not carried across by the user from the IdP to the SP. Instead the user transfers an ‘artifact’ which is a reference to a security assertion. The SP, on receiving the artifact, can use it to obtain the authentication assertion from the IdPs artifact resolution service via a back-channel communication exchange.

4.1.1.4 ATTRIBUTE AUTHORITY

The attribute authority is responsible for issuing attribute assertions. All queries are authenticated and authorised before processing. The attribute authority will only release attributes that are permitted to be released by the Attribute Release Policy, which is user customisable.

4.1.2 SERVICE PROVIDER

The Service Provider role in the Shibboleth is governed by two logical components, the Assertion Consumer Service (ACS) and the Attribute Requester (AR).

4.1.2.1 ASSERTION CONSUMER SERVICE

The assertion consumer service processes authentication assertions that are either provided by the user (browser/post profile) or obtained via the artifact resolution service (browser/artifact profile). It is also able to request further attributes (via the attribute requester) to evaluate any authorisation decisions that need to be made. This service also establishes a security context at the SP and redirects users to their requested resource.

4.1.2.2 ATTRIBUTE REQUESTER

The attribute requester communicates directly with the attribute authority at the IdP via back-channel communication, bypassing the user/browser.

4.1.3 WHERE ARE YOU FROM

The Where Are You From (WAYF) service is independent of both the SP and the IdP. It is fundamentally a registry of IdPs within the federation that it supports where SPs can point their users to for beginning the authentication process. This component is optional, as it is possible for users to authenticate at their IdPs first and arrive at the SP already carrying Shibboleth authentication tokens.

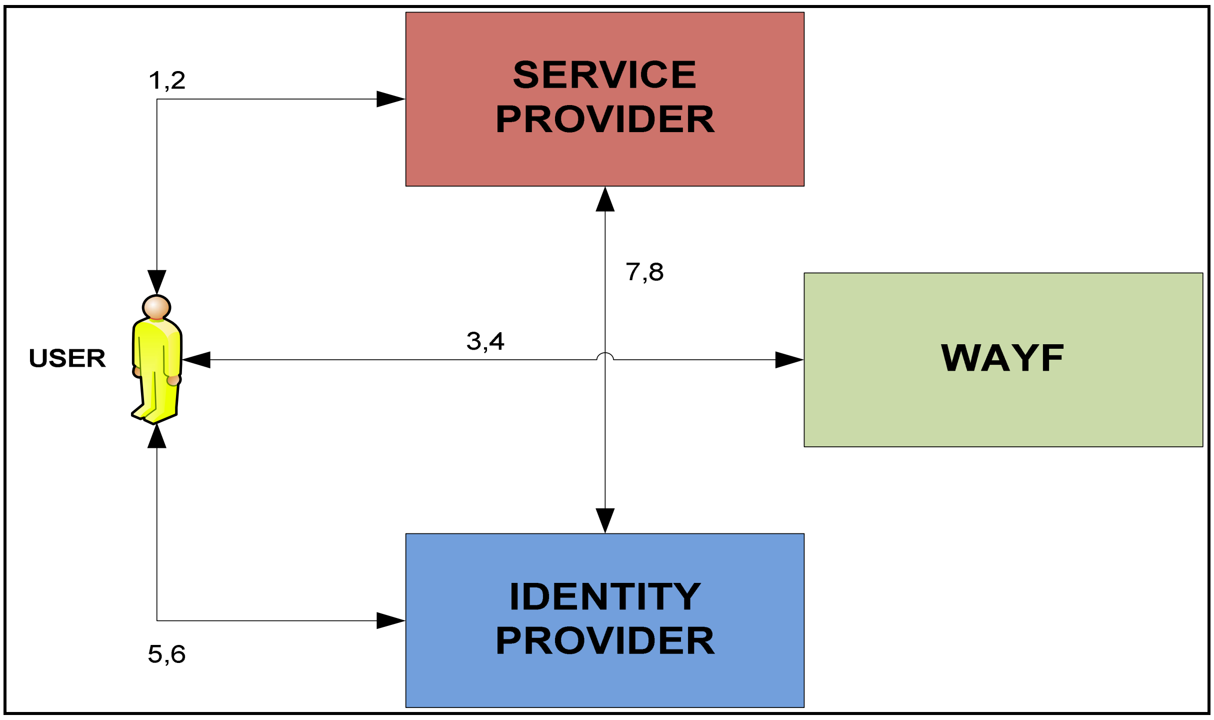
4.2 HIGH LEVEL COMPONENT INTERACTION

In the Shibboleth framework, the process is triggered by a user requesting a resource. Once this occurs, the sequence and components come in to play as follows:

1) User requests resource at the SP.
2) SP checks if the user is carrying a Shibboleth token. If they are, then the token is verified with the IdP that is associated with the token and the user is allowed access to the resource. If not, then we go to step 3.
3) SP returns a redirect tag to client and sends them to the WAYF, which provides a list of its registered IdPs within this particular Shibboleth federation.
4) The user is then able to select their IdP to which they are then sent.
5) At the IdP, they are challenged to authenticate themselves. The mechanism by which an IdP can authenticate a user is not part of the specification. An IdP can implement anything from a username/password to a biometric based challenge.
6) Once the user is authenticated, the authentication token is created and sent along with the user back to the resource at the SP where the process begins again.

7) In some profiles, this back-channel communication pathway is also used for the SP to verify that the user did in fact authenticate with the IdP and can optionally query the IdP for additional attributes that might be required for a SP to evaluate an authorisation decision.
8) The IdP can respond to queries about user attributes as long as the ARP (Attribute Release Policy) allows this information to be shared with the specific requesting SP.

FIGURE 2 - SHIBBOLETH COMMUNICATION PATHWAYS

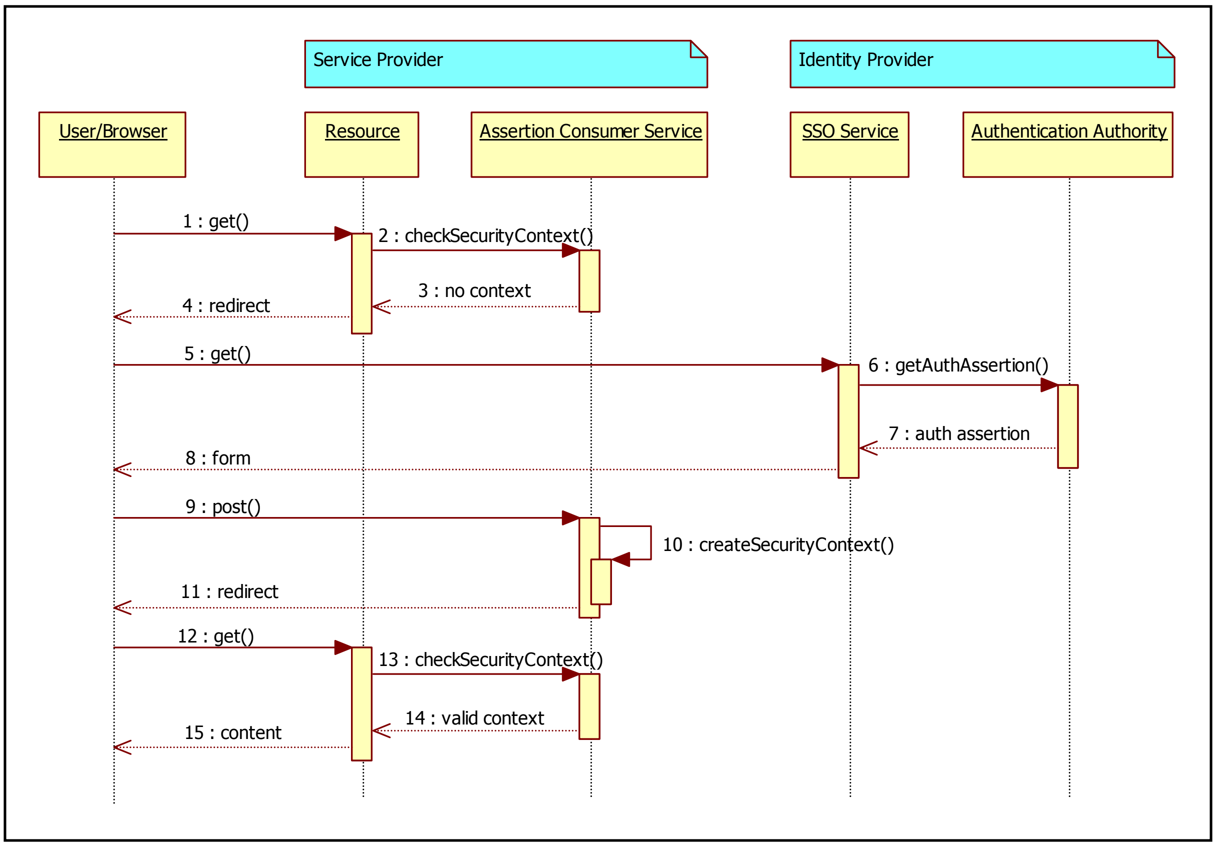


4.3 SHIBBOLETH SSO PROFILES

The Shibboleth SSO profiles [Cantor 2005c] govern the interaction between the Shibboleth components to provide access to resources for eligible users. These profiles allow a user to authenticate once only and then access services from service providers throughout the federation without having to authenticate again. There are multiple profiles, each catering for particular requirements. This section will outline each of the profiles used within Shibboleth processes and discuss the differences between them. There are primarily two main profiles, the Browser/POST profile and the Browser/Artifact profile. In addition to these basic profiles, each has slight variations encompassing the use of the WAYF and attribute exchange resulting in a total of eight possible profiles. These profiles are based on the SAML profiles (Hughes et al. 2005).

4.3.1 BROWSER/POST PROFILE

FIGURE 3 - SHIBBOLETH BROWSER/POST PROFILE SEQUENCE



The Browser/POST profile seems to be the most common profile in use. The reason for this could be that it is the simplest profile to support as there are no assumed communication pathways between the SP and the IdP. Instead, the User, via the web browser, is responsible for facilitating the communication of messages between these components primarily through the use of HTTP redirection codes [Fielding et al. 1999]. In this particular profile, we are not using a WAYF, which means that there has to be some way for the SP to identify which IdP to send the user to for an authentication request. In this scenario, it is essential that the user has already authenticated with their IdP and is carrying a client side cookie with the IdPs identity. If this is not available then there is no way for the SP to redirect a user to the IdP and the request will fail.

As with all the Shibboleth profiles we will be discussing, the processing begins with a request from a user for a resource at the SP – in this specific case, the user having, at some point prior in their browsing session, authenticated at their IdP. The table below describes the steps involved in this process.

TABLE 1 - BROWSER/POST PROFILE SEQUENCE

SEQ.	DESCRIPTION
1	Using a web browser, the User attempts to retrieve a resource from a particular SP. The request can be any form of HTTP request, e.g. http://sp.example.org/resource.

SEQ.	DESCRIPTION
2	The SP checks whether the browser is carrying any tokens (or cookies) that relate to a security context within the Assertion Consumer Service (ACS). If such a token is found then the ACS is queried for a valid context. If one exists we skip to step 14.
3	The SP is informed that no valid context is found.
4	As mentioned above, it is essential the User already be authenticated and be carrying a cookie bearing the endpoint of the IdP SSO endpoint. It is here that the SP redirects the browser to the endpoint where the Users identity can be established. The SP does this by returning a HTTP 302 status code indicating a temporary redirection and a 'Location' header that is the URL of the IdP SSO endpoint. In addition, there are several parameters that are appended to this URL as a HTTP query string. These are 'target', which is the URL of the resource that the user initially attempted to acquire; 'shire', which is the ACS at the SP, and 'providerId', which is the unique identifier of the SP (typically a URI). There is also an optional 'time' parameter that is the number of seconds past the Unix Time Epoch.

5	The client is redirected to the IdP SSO endpoint with the authentication request. The request carries that 'target', 'shire' and 'providerId' elements, with an optional 'time' element. E.g.
	https://idp.example.org/shibboleth/SSO?
	   target=https://sp.example.org/resource&
	   shire=https://sp.example.org/shibboleth/SSO/POST&
	   providerId=https://sp.example.org/shibboleth

6	The IdP SSO receives the authentication request and attempts to identify whether the user already has a current security context. In this particular profile, as it is a requirement for the user to have been already authenticated, the user will in fact have an associated security context. In other profiles, however, a user might need to be authenticated at the IdP to establish a security context if they do not already have one. The SSO endpoint sends a request to the Authentication Authority which is responsible for authenticating users (if required) and providing authentication assertions. The mechanism for authentication is external to the scope of the Shibboleth specification and it is up to the IdP to manage the processes to perform this task. Any form of authentication can be supported, be it username/password, biometric, secure token, multi-factor etc.
7	An authentication assertion is returned to the IdP SSO Service which then, if necessary, creates a security context for the User.
8	The IdP SSO Service returns a HTML form to the User/Browser. The action of this form is a POST to the SP Assertion Consumer Service and the payload of the POST is the SAML authentication response which itself contains the authentication assertion. The action is derived from the preservation of the 'shire' attribute from step 5. The SAML authentication response is digitally signed by the IdP and the digital signature information is also a part of the SAML response. The digital signature ensures authenticity of the authentication assertion as this piece of information is carried between security entities via a potentially untrusted entity (the User/Browser). In addition, the URI of the initial resource that was requested at the SP is also added to this form. This information was also provided in step 5.

SEQ.	DESCRIPTION
9	The form that the User receives needs to be submitted. In many cases this can be achieved via JavaScript so that this step occurs automatically, with no user intervention required. The form is submitted to the ACS at the SP. Again, this value was obtained from the 'shire' attribute in step 5 and the form contains the SAML authentication response, the authentication assertion and the digital signature information of the IdP.
10	The ACS at the SP processes the SAML authentication response and creates a security context for the User at the SP. The security context session is keyed to cookies held by the User.
11	Once the security context is created, the SP then redirects the User to the initially requested resource via a HTTP redirect and Location header.
12	The User makes a request to the SP for the resource in exactly the same way as step 1.
13	The SP checks whether the browser is carrying any tokens (or cookies) that relate to a security context within the Assertion Consumer Service (ACS). In this instance there is such a context and the User is validated against it.
14	The ACS returns the valid context to the SP.
15	The resource is provided to the User.

The Browser/POST profile is the simplest of the profiles from the implementation perspective. It requires no backchannel communication between parties and relies wholly on the User facilitating communication. Some of the mechanisms of this interaction seem crude, such as the JavaScript interaction for the form submission to the SP in step 9, as care has to be taken to ensure Browser compatibility with the JavaScript being employed.

4.3.2 BROWSER/POST + WAYF PROFILE

The second profile is a derivation of the first with the addition of the Where Are You From (WAYF) component. As we stated for the Browser/POST profile, it was imperative that a User authenticate with an IdP before accessing a resource at the SP so that the SP would be able to ascertain which IdP to direct users to for obtaining authentication assertions. With the addition of the WAYF to this profile, however, this imperative is relaxed. The SP is configured with a WAYF URI to which it can direct a User so that they can identify their IdP themselves.

The WAYF module in this case is fundamentally a registry of IdPs that support the particular 'Federation' that the User is in, i.e. a range of service and identity providers that share a level of trust with each other. When a user is directed to the WAYF they are presented with this registry and are required to select their personal IdP, which they are then sent to for identity validation.

Other than this step, this profile is similar to the first and the exact sequence is provided in the diagram and table below.

FIGURE 4 - SHIBBOLETH BROWSER/POST + WAYF PROFILE SEQUENCE

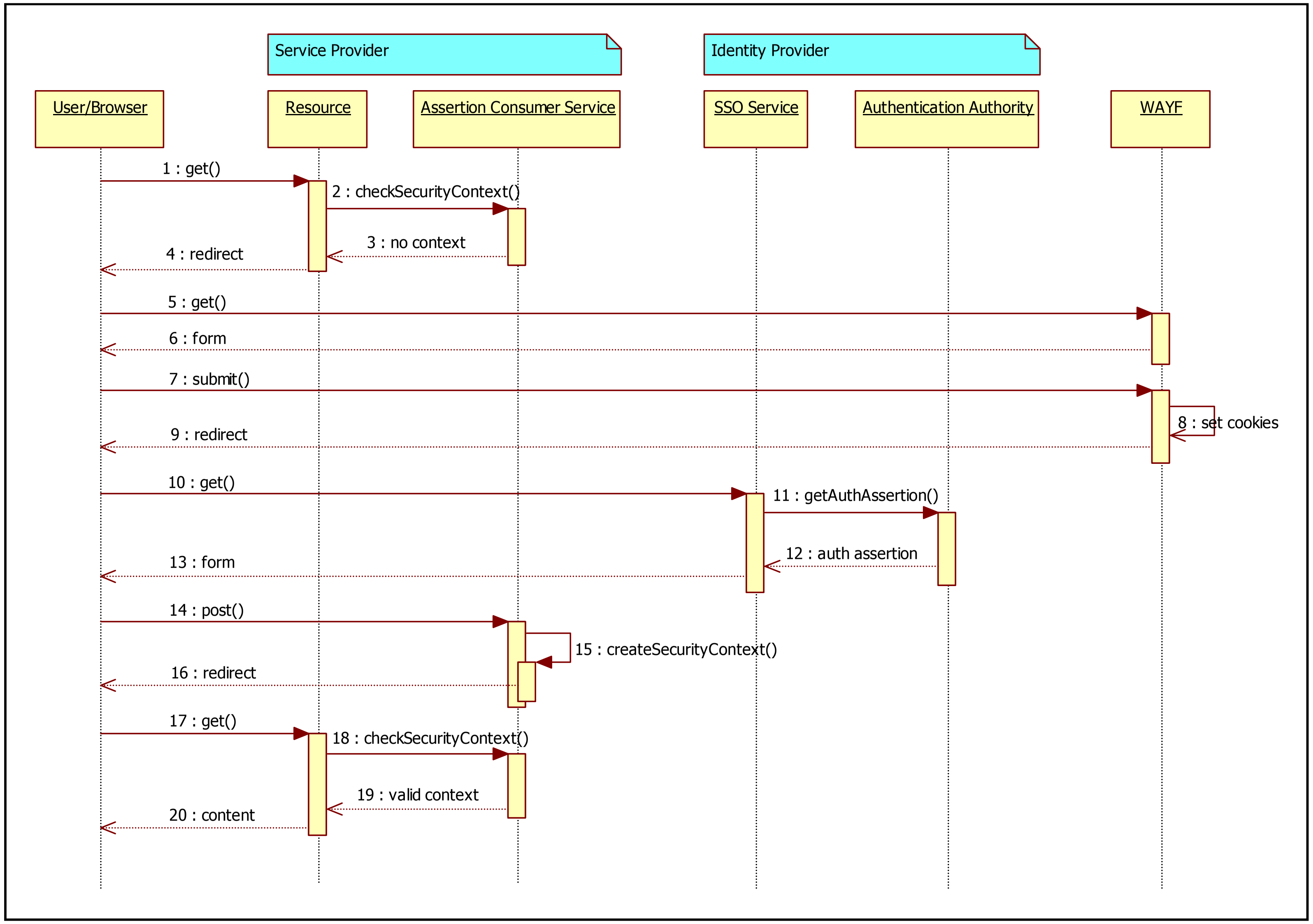


TABLE 2 - BROWSER/POST + WAYF PROFILE SEQUENCE

SEQ.	DESCRIPTION
1	Using a web browser, the User attempts to retrieve a resource from a particular SP. The request can be any form of HTTP request, e.g. http://sp.example.org/resource.
2	The SP checks whether the browser is carrying any tokens (or cookies) that relate to a security context within the Assertion Consumer Service (ACS). If such a token is found then the ACS is queried for a valid context. If one exists we skip to step 14.
3	The SP is informed that no valid context is found.
4	The SP then checks for a cookie containing the IdP of the User. If no cookie is found (implying that the User has not authenticated), the SP sends the User to the WAYF using a HTTP redirect. The WAYF is pre-registered with the SP. In addition, there are several parameters that are appended to this redirect URL as a HTTP query string. These are 'target', which is the URL of the resource that the user initially attempted to acquire; 'shire', which is the ACS at the SP, and 'providerId', which is the unique identifier of the SP (typically a URI). There is also an optional 'time' parameter that is the number of seconds past the Unix Time Epoch.

5	The client is redirected to the WAYF to perform identity provider discovery. The request carries the 'target', 'shire' and 'providerId' elements, with an optional 'time' element. E.g.
	https://idp.example.org/shibboleth/SSO?
	target=https://sp.example.org/resource&
	shire=https://sp.example.org/shibboleth/SSO/POST&
	providerId=https://sp.example.org/shibboleth

6	The WAYF responds to the User with a form. Fields in the form correspond with the parameters from step 5 in order to preserve these across multiple requests. In addition to these parameters, the WAYF provides some method for the User to select their IdP as part of the form. The exact method for doing this is discretionary for the WAYF.
7	The User selects their IdP and submits the form to the WAYF.
8	The WAYF sets a cookie in the User browser identifying their IdP of choice. This cookie can be used by SPs to identify the User IdP without having to repeat the WAYF process in this session.
9	The WAYF redirects the user to their IdP and preserving the parameters which have been carried by the User from steps 5-8.

10	The client is redirected to the IdP SSO endpoint with the authentication request. The request carries that 'target', 'shire' and 'providerId' elements, with an optional 'time' element. E.g.
	https://idp.example.org/shibboleth/SSO?
	target=https://sp.example.org/resource&
	shire=https://sp.example.org/shibboleth/SSO/POST&
	providerId=https://sp.example.org/shibboleth

SEQ.	DESCRIPTION
11	The IdP SSO receives the authentication request and attempts to identify whether the user already has a current security context. In this particular profile, as it is a requirement for the user to have been already authenticated, the user will in fact have an associated security context. In other profiles, however, a user might need to be authenticated at the IdP to establish a security context if they do not already have one. The SSO endpoint sends a request to the Authentication Authority which is responsible for authenticating users (if required) and providing authentication assertions. The mechanism for authentication is external to the scope of the Shibboleth specification and it is up to the IdP to manage the processes to perform this task. Any form of authentication can be supported, be it username/password, biometric, secure token, multi-factor etc.
12	An authentication assertion is returned to the IdP SSO Service which then, if necessary, creates a security context for the User.
13	The IdP SSO Service returns a HTML form to the User/Browser. The action of this form is a POST to the SP Assertion Consumer Service and the payload of the POST is the SAML authentication response which itself contains the authentication assertion. The action is derived from the preservation of the 'shire' attribute from step 5. The SAML authentication response is digitally signed by the IdP and the digital signature information is also a part of the SAML response. The digital signature ensures authenticity of the authentication assertion as this piece of information is carried between security entities via a potentially untrusted entity (the User/Browser). In addition, the URI of the initial resource that was requested at the SP is also added to this form. This information was also provided in step 5.
14	The form that the User receives needs to be submitted. In many cases this can be achieved via JavaScript so that this step occurs automatically, with no user intervention required. The form is submitted to the ACS at the SP. Again, this value was obtained from the 'shire' attribute in step 5 and the form contains the SAML authentication response, the authentication assertion and the digital signature information of the IdP.
15	The ACS at the SP processes the SAML authentication response and creates a security context for the User at the SP. The security context session is keyed to cookies held by the User.
16	Once the security context is created, the SP then redirects the User to the initially requested resource via a HTTP redirect and Location header.
17	The User makes a request to the SP for the resource in exactly the same way as step 1.
18	The SP checks whether the browser is carrying any tokens (or cookies) that relate to a security context within the Assertion Consumer Service (ACS). In this instance there is such a context and the User is validated against it.
19	The ACS returns the valid context to the SP.
20	The resource is provided to the User.

4.3.3 BROWSER/ARTIFACT PROFILE

The Browser/Artifact profile of Shibboleth is distinguished by the use of a backchannel communication with an IdP. This particular profile is more complex than the Browser/POST variants as it performs an out of band communication step with the Users IdP to resolve an 'artifact'. The 'artifact' is an identifier for an authentication assertion. In the Browser/POST profile, the authentication assertion is sent to the SP via the User browser. In this profile, the artifact is sent to the SP via the User browser instead of the authentication assertion itself. The artifact is received by the ACS which then sends a request to the IdP Artifact Resolution Service which in turn responds with the authentication assertion. This additional step ensures that the authentication assertion is not passed around by the User and thus increases the level of security in the process.

The steps in this sequence are outlined by the diagram and table below.

TABLE 3 - BROWSER/ARTIFACT PROFILE SEQUENCE

SEQ.	DESCRIPTION
1	Using a web browser, the User attempts to retrieve a resource from a particular SP. The request can be any form of HTTP request, e.g. http://sp.example.org/resource.
2	The SP checks whether the browser is carrying any tokens (or cookies) that relate to a security context within the Assertion Consumer Service (ACS). If such a token is found then the ACS is queried for a valid context. If one exists we skip to step 14.
3	The SP is informed that no valid context is found.
4	Without the WAYF in the process, the requirement of the User already having being authenticated is reinstated. This means that the IdP SSO endpoint is available to the SP via a User cookie. The SP redirects the User to this IdP SSO endpoint. Again, the target, shire and providerId attributes are added to the redirection URL.

5	The client is redirected to the IdP SSO endpoint to with the authentication request. The request carries that 'target', 'shire' and 'providerId' elements, with an optional 'time' element. E.g.
	https://idp.example.org/shibboleth/SSO?
	target=https://sp.example.org/resource&
	shire=https://sp.example.org/shibboleth/SSO/POST&
	providerId=https://sp.example.org/shibboleth

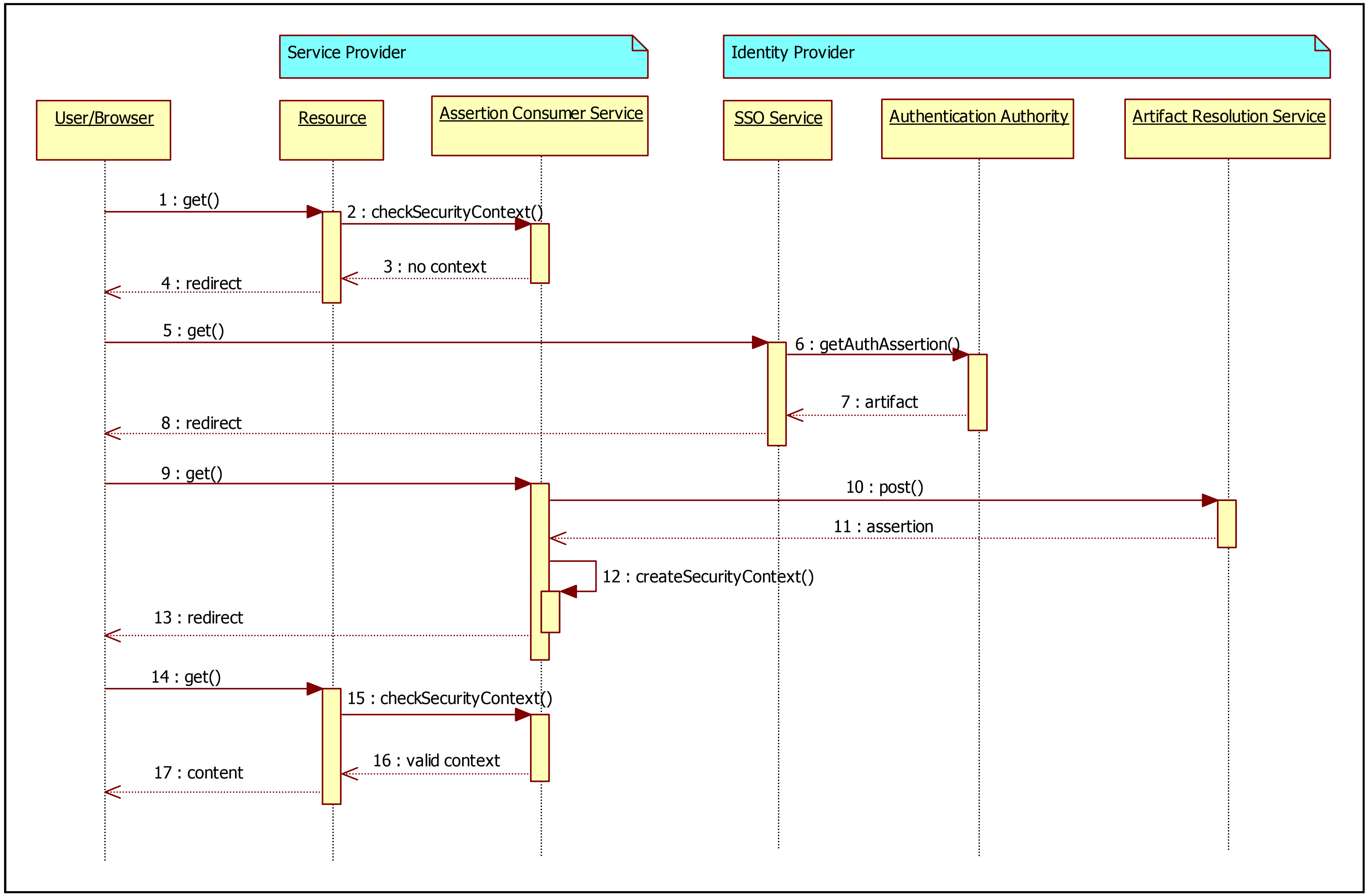
6	The IdP SSO receives the authentication request and attempts to identify whether the user already has a current security context. In this particular profile, as it is a requirement for the user to have been already authenticated, the user will in fact have an associated security context. In other profiles, however, a user might need to be authenticated at the IdP to establish a security context if they do not already have one. The SSO endpoint sends a request to the Authentication Authority which is responsible for authenticating users (f required) and providing authentication assertions. The mechanism for authentication is external to the scope of the Shibboleth specification and it is up to the IdP to manage the processes to perform this task. Any form of authentication can be supported, be it username/password, biometric, secure token, multi-factor etc.

SEQ.	DESCRIPTION
7	An authentication assertion is created and stored at the Authentication Authority and an 'artifact' representing this assertion is returned to the IdP SSO Service.
8	The IdP SSO Service returns a HTML form to the User/Browser. The action of this form is a POST to the SP Assertion Consumer Service and the payload of the POST is the SAML artifact. The action is derived from the preservation of the 'shire' attribute from step 5. In addition, the URI of the initial resource that was requested at the SP is also added to this form. This information was also provided in step 5.
9	The form that the User receives needs to be submitted. In many cases this can be achieved via JavaScript so that this step occurs automatically, with no user intervention required. The form is submitted to the ACS at the SP. Again, this value was obtained from the 'shire' attribute in step 5 and the form contains the SAML artifact.
10	The ACS processes the SAML response and extracts the Artifact. Based again on the User cookie, the IDP Artifact Resolution Service is contacted with a request containing the artifact.
11	The Artifact Resolution Service resolves the artifact to an authentication assertion and returns a SAML authentication response containing the resolved authentication assertion to the ACS.
12	The ACS at the SP processes the SAML authentication response and creates a security context for the User at the SP. The security context session is keyed to cookies held by the User.
13	Once the security context is created, the SP then redirects the User to the initially requested resource via a HTTP redirect and Location header.
14	The User makes a request to the SP for the resource in exactly the same way as step 1.
15	The SP checks whether the browser is carrying any tokens (or cookies) that relate to a security context within the Assertion Consumer Service (ACS). In this instance there is such a context and the User is validated against it.
16	The ACS returns the valid context to the SP.
17	The resource is provided to the User.

4.3.4 BROWSER/ARTIFACT + WAYF

This profile is analogous to the Browser/POST profile with the WAYF. The user is not required to have previously authenticated at the IdP as they are redirected to the WAYF component for IdP discovery.

FIGURE 5 - SHIBBOLETH BROWSER/ARTIFACT PROFILE



4.3.5 ATTRIBUTE EXCHANGE PROFILE VARIATION

Both the Browser/POST and the Browser/Artifact profiles (with or without the WAYF) are also able to perform an attribute exchange. This step is for the ACS to obtain more information about the Principal.

During the processing of the Authentication Assertion by the ACS, if more information is required, the ACS is able to tell the Attribute Requester component to contact the Attribute Authority at the Principal's IdP. This is a SOAP request/response which will result in a SAML Attribute Assertion if the information is available for the SP. Once this information is retrieved, the ACS can continue processing the Authentication Assertion.

4.3.6 SINGLE LOGOUT

Converse to single sign on, is single logout. This profile allows a user to logout of the IdP and simultaneously terminate all sessions at all service providers that they authenticated at, effectively logging them out of those services. This profile is difficult to implement and is implementation specific.

4.4 PRIVACY

As with most Federated Identity Management systems, privacy is an important concern. The Shibboleth view is that the SP knows only the minimum necessary amount of information about a user. To accomplish this Shibboleth makes use of a special unique identifier, and attribute release policies.

4.4.1 UNIQUE IDENTIFIERS

In the interest of privacy and to prevent service providers from colluding and generating behavioural profiles of users, the Shibboleth framework provides to the SP a 'targeted identifier', which is an identifier that is derived from the user id, and the id of the SP to ensure that the SP will never know the IdP based ID of a user and that the targeted ID is unique. This allows the SP to create local accounts and preference information for the user while still allowing the user to maintain their privacy. This targeted identifier is passed in assertions as the 'saml:NameIdentifier' attribute which is part of the SAML specification.

4.4.2 ATTRIBUTE RELEASE POLICY

The Attribute Release Policy is a user configurable policy at the IdP. This policy allows the user to configure which attributes they are willing to release to particular SPs. In order to use a particular service, however, the SP might require a certain amount of information about the user. If a user does not allow that information to be released to the SP, they will not be able to use the service. For example, John wants to use a Stock Watch service which alerts him when stocks he has configured reach levels that he has specified. In order to alert John, the service requires that he either provides an email address or a mobile phone number for alerts. If John does not want to release this information to the service, he will not be able to use it. To use this service, John will have to configure his Attribute Release Policy (ARP) at his IdP, to allow either his email address or mobile phone number to be released to the SP.

5 LIBERTY ALLIANCE

In this section we will discuss the Liberty Alliance architecture [Wason 2003], examining the components and profiles that it defines.

Liberty Alliance is a project sponsored by over 150 different commercial organisations, including many of the influential Information Technology companies such as IBM, Sun, Microsoft, Intel and more. As the primary focus of these organisations is the commercial sector, the strategies and business requirements that drive Liberty are aligned with respect to these types of organisations. This is in contrast to the Shibboleth architecture, which is primarily focussed on the education sector.

Given the focus and the broader range of requirements that guide Liberty, the specification is a lot larger and more comprehensive than that of Shibboleth. The complete Liberty Alliance architecture encompasses not just identity management and single sign on profiles for a user using a browser (ID-FF, Identity Federation Framework), but also for web services (ID-WSF, Identity Web Services Framework) and an interface specification for the development of identity profile services (ID-SIS, Identity Services Interface Specification). As the Shibboleth architecture deals only with the identity management and single sign on for users using a browser, we will confine our discussion and analysis to the relevant sections of Liberty only – the Identity Federation Framework (ID-FF), which is the foundation of the Liberty Alliance Project.

5.1 COMPONENTS

The ID-FF platform of Liberty recognises both the Identity and Service Provider components. These components use the SAML specification to facilitate communication between themselves and the User.

5.1.1 IDENTITY PROVIDER

The Liberty specifications do not break down the IdP component into logical subcomponents as it is left to implementers of this specification to handle those aspects as they see fit. Instead, the IdP holds metadata about itself with information such as public keys, providerId and URL endpoints for the various services at the IdP. This metadata [Davis 2005] is shared between all members of the Circle of Trust to which the IdP belongs. Multiple Circles of Trust are also possible.

Again, the IdP is responsible for authenticating principals and providing security assertions about them to relying parties for the purposes of single sign on. An additional responsibility of the IdP is to facilitate identity federation with SPs as well as other IdPs and conversely identity de-federation. Identity federation is explained in a following section. As part of the single sign on process, a user is also able to perform a single logout. Single logout effectively logs a user out of all sessions that they are currently connected to within the Circle of Trust.

5.1.2 SERVICE PROVIDER

Similar to the IdP, the SP is not broken down into logical sub components as Liberty views these as implementation specific. The SP is also required to have a metadata profile [Davis

2005] that provides members of the Circle of Trust with information regarding the various service endpoints that the SP contains.

Apart from the user services that it provides, the Liberty based responsibilities of the SP include adherence to the Single Sign On and Federation profile to provide single sign on, single logout, federation and de-federation services. In conjunction with the IdP, the SP also has to support the Name Registration and Name Identifier Mapping profiles. These profiles are used for the generation and discovery of a pseudonym (an opaque identifier) that a SP can use when referring to a principal at an IdP. This pseudonym is different for each SP and provides a higher level of privacy for a user by hiding their IdP based unique identifier. This prevents SPs from colluding and profiling user behaviour based on a common identifier.

5.2 IDENTITY FEDERATION

The concept of Identity Federation is referenced many times throughout the Liberty Alliance specification set. Federation is the linking of an account at an IdP to an account at the SP. This account is linked using the pseudonym that is agreed upon by both parties. A user would have an identity at the IdP which would be referenced by a unique identifier. This identifier is never released to any other IdPs or SPs. Identity federation links this identifier with an identifier that is specific to that user at the SP. The SP uses this pseudonym when communicating with the IdP and referencing the user. Using this pseudonym, the SP is also able to have a local account for the user along with any preferences and settings that are required for a user to customise and use that service.

5.3 LIBERTY SSO AND FEDERATION PROFILES

For components within Liberty to communicate, they need to adhere to the Single Sign On and Federation profiles. Each of these profiles governs the required sequence of events that need to occur to perform their function. The profiles for single sign on are derived from the profiles defined by the SAML specification [Cantor et al. 2005a].

There are three profiles in this category, the Liberty Browser POST Profile, the Liberty Artifact Profile and the Liberty Enabled Client Proxy Profile. At this time we will only be discussing the first two profiles.

5.3.1 LIBERTY BROWSER POST PROFILE

The Liberty Browser Profile is based on the SAML Browser Profile. It defines the use case of a user accessing a resource at a SP. The user need not be authenticated at an IdP. The following sequence diagram and table describe the profile.

It is important to note that in steps 5 and 6, the process of Identity Provider discovery is not specified. From the Liberty perspective, it is up to the service providers to develop solutions to this issue. Liberty has provided a solution called the Identity Provider Introduction profile. This profile suggests some solutions that a SP might look at or use as a starting point for. We will describe this profile in a following section.

The diagram and table below detail the sequence of this profile. As with all the SSO profiles, it begins with a request to a resource or service by the User Agent.

FIGURE 6 - LIBERTY BROWSER POST PROFILE SEQUENCE

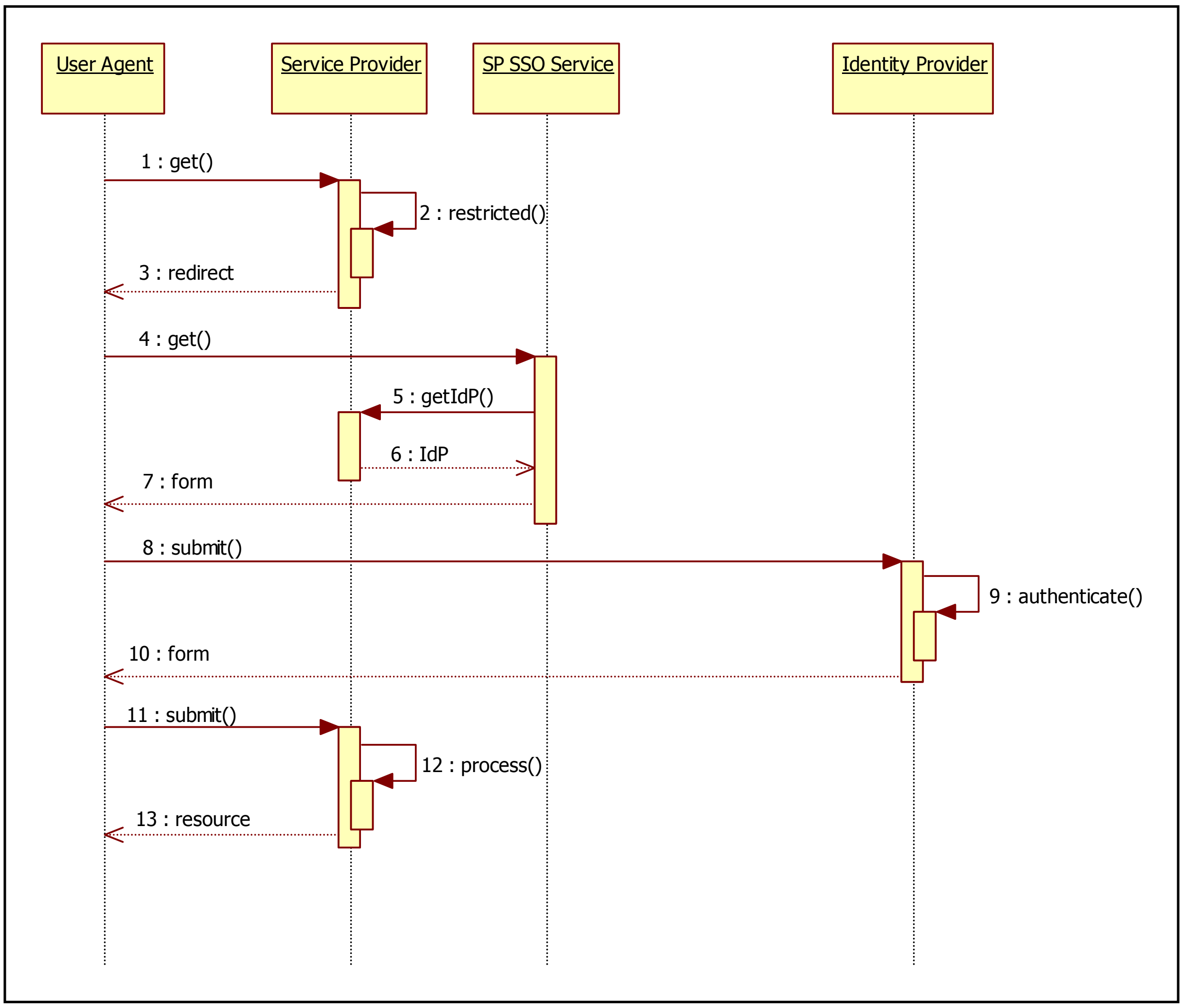


TABLE 4 - LIBERTY BROWSER POST PROFILE SEQUENCE

SEQ.	DESCRIPTION
1	The User Agent makes a request for a resource at the SP.
2	The SP identifies that this is a secured resource and requires authentication. It also identifies that the User Agent is not authenticated.
3	The SP responds with a HTTP redirect (302 temporary relocation) that points to the SPs SSO service. The redirection URL preserves the initial resource request URL in the RelayState parameter.
4	The User Agent sends a GET request to the SSO Service with the RelayState parameter attached to the URL preserving the initial resource request.
5	The SSO Service gets the SP to identify the IdP to be used. This process is implementation specific, although there are some suggestions provided by the Liberty specification on some methods that could be used to do so.
6	The SP responds with the IdP to be used.

SEQ.	DESCRIPTION
7	The SSO Service responds to the client with a form. The action of the form points to the IdP SSO service (using https) and the contents of the form contain a Liberty Authentication Request (LAREQ) in a base64 encoded format within a field called LAREQ. Within the LAREQ is an actual SAML Authentication Request. The RelayState parameter containing the initial resource request is also preserved within the form. Alternatively, the SSO Service can respond with a HTTP redirect with the Liberty Authentication Request encoded into URL query parameters [Cantor et al. 2005d].
8	The form is submitted to the IdP.
9	The IdP processes the authentication request and, if necessary, is able to authenticate the User at this point.
10	The IdP responds with a form. The action of the form points to the URL at the SP that processes security assertions (Assertion Consumer). The form also contains a field called LARES which itself contains a base64 encoded Liberty Authentication Response. Within the LARES is an actual SAML Authentication Response. In addition, the RelayState is present containing the initial resource URL.
11	The User Agent submits the form to the SP Assertion Consumer URL.
12	The Liberty Authentication Response is processed.
13	The User Agent is provided with the resource.

5.3.2 LIBERTY ARTIFACT PROFILE

The Liberty Artifact Profile is based on the SAML Artifact Profile from the SAML specification. This profile is identical to the Browser POST profile except that it does not rely on the User Agent to carry authentication assertions between the IdP and the SP. Instead, a backchannel communication between the SP and the IdP occurs with the User Agent carrying just a reference, known as a SAML Artifact, to the assertion.

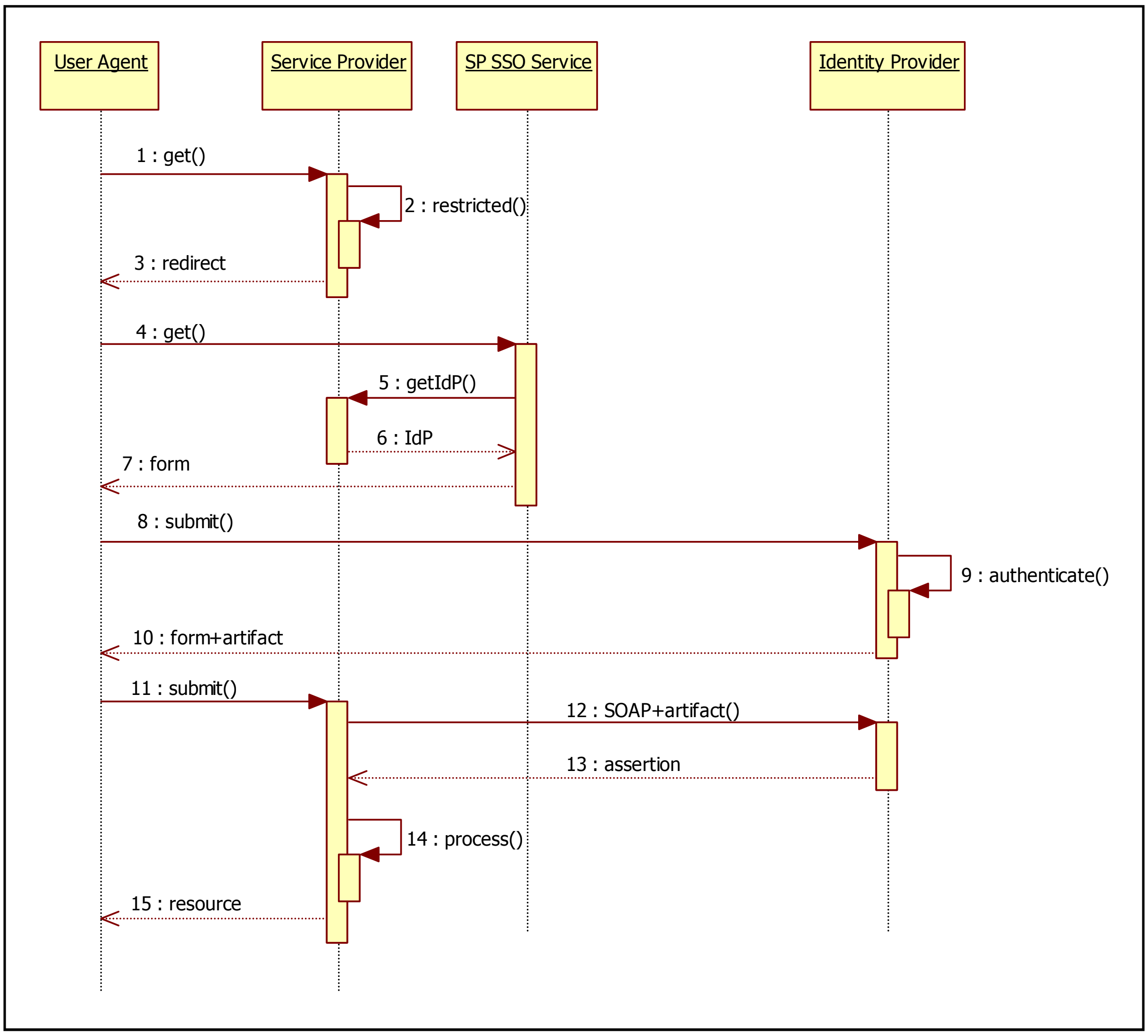
It should also be noted that in many cases within the Liberty framework, the use of SOAP messages and GET requests are both supported.

TABLE 5 - LIBERTY ARTIFACT PROFILE SEQUENCE

SEQ.	DESCRIPTION
1	The User Agent makes a request for a resource at the SP.
2	The SP identifies that this is a secured resource and requires authentication. It also identifies that the User Agent is not authenticated.
3	The SP responds with a HTTP redirect (302 temporary relocation) that points to the SPs SSO service. The redirection URL preserves the initial resource request URL in the RelayState parameter.

SEQ.	DESCRIPTION
4	The User Agent sends a GET request to the SSO Service with the RelayState parameter attached to the URL preserving the initial resource request.
5	The SSO Service gets the SP to identify the IdP to be used. This process is implementation specific, although there are some suggestions provided by the Liberty specification on some methods that could be used to do so.
6	The SP responds with the IdP to be used.
7	The SSO Service responds to the client with a form. The action of the form points to the IdP SSO service (using https) and the contents of the form contain a Liberty Authentication Request (LAREQ) in a base64 encoded format within a field called LAREQ. Within the LAREQ is an actual SAML Authentication Request. The RelayState parameter containing the initial resource request is also preserved within the form. Alternatively, the SSO Service can respond with a HTTP redirect with the Liberty Authentication Request encoded into URL query parameters [Cantor et al. 2005d].
8	The form is submitted to the IdP.
9	The IdP processes the authentication request and, if necessary, is able to authenticate the User at this point.
10	The IdP can respond with either a redirect or a form. If using a Form, the action points to the URL at the SP that processes security assertions (Assertion Consumer). The form also contains a field called LARES which itself contains a base64 encoded SAML Artifact. In addition, the RelayState is present containing the initial resource URL. If using the redirect method, the IdP redirects the User Agent using a HTTP redirect with a Location header sending it directly to the Assertion Consumer at the SP. The URL will have the RelayState parameter appended and also have the parameter 'SAMLart' present with the SAML Artifact.
11	The User Agent submits the form to the SP Assertion Consumer URL or has been redirected with the appropriate parameters within the URL.
12	The SP sends a SAML request over SOAP to the IdPs SOAP endpoint for artifact resolution. The message is a standard SAML request containing the 'AssertionArtifact' element.
13	The IdP processes the SAML request and resolves the AssertionArtifact element to the appropriate SAML authentication assertion, with which it responds to the SP in the form of a SAML response.
14	The SAML Response is processed.
15	The User Agent is provided with the resource.

FIGURE 7 - LIBERTY ARTIFACT PROFILE SEQUENCE



5.4 OTHER PROFILES

In addition to the Single Sign On and Federation Profiles of Liberty, there are several other profiles that aid in the management of federated identities; these include Single Logout, Federation Termination Notification, Identity Provider Introduction and Name Registration. We will discuss these briefly.

5.4.1 SINGLE LOGOUT

The Single Logout profile is derived from the SAML specification as well. The purpose of this Profile is to guide a use case for the termination of all currently active sessions at SPs for a particular user across the Circle of Trust. This ensures that any session specific data held at any of the SPs the user has visited is destroyed and the session terminated.

Typically, this would happen automatically once a user has not been active at a particular service for the session timeout duration – a common concept across web based services.

Alternatively, a user would have to logout of each service individually.

5.4.2 FEDERATION TERMINATION NOTIFICATION

This profile severs the connection for a principal between the IdP and the SP. The IdP informs the SP that it will no longer be servicing identity based requests for the SP for the particular principal that is being de-federated.

5.4.3 IDENTITY PROVIDER INTRODUCTION

This profile intends to assist in Identity Provider Discovery. There are no concrete implementations specified for this profile; however, there are several suggestions put forward. This profile is largely left as an implementation problem.

The suggestions involve using a common domain for the purposes of sharing cookies. Using a common domain, both an IdP and a SP are able to view cookies for a particular domain. This suggestion recommends that the IdP set a specific cookie, _liberty_ipd, with the value being the providerId of the IdP. This cookie will then contain a list of all IdPs that the User is associated with and will provide a list for SPs to use for authentication.

Again, this is merely a suggestion, and more suitable methods are encouraged within the specification.

5.4.4 REGISTER NAME IDENTIFIER

During the process of Identity Federation a pseudonym representing the principal to the SP is generated that is unique to the SP. This pseudonym prevents the SP from knowing what the identifier for the principal at the IdP is, preserving privacy. The pseudonym is linked to the IdP identifier at the IdP.

This profile is used both at the time of federation for the establishment of the pseudonym and at any subsequent time for changing that pseudonym should the need arise. This profile is optional.

6 ARCHITECTURE ANALYSIS

Now that we have examined each of the architectures and deconstructed their communication profiles, we can perform a comparative analysis and establish their commonalities and differences. This will enable us to identify where opportunities for integration might exist and what the requirements would be for exploiting them.

6.1 INITIAL ANALYSIS

We can easily see that the two frameworks have much in common. The reason for this is they are both based on the same SAML specification. From the profiles that both frameworks present, we can still see subtle differences in their focus.

The Liberty architecture, for instance, has the view that SPs each maintain local accounts for principals, along with attribute information, and use the IdP for the purposes of authentication and SSO only. This is not a technical restriction placed on the framework by any means, but does suggest a slightly different viewpoint of Federated Identity than that of Shibboleth.

Shibboleth, on the other hand, does not advocate the management of accounts locally at the SP and the lack of profile support for the management of such accounts suggest this. Again, this is not a technical restriction, but a differing viewpoint on Federated Identity. Shibboleth was primarily designed for resource sharing, and not typically service sharing which is where the distinction lies. When sharing resources, it is often not necessary for any local accounts to exist. You only care that the principal is who they say they are, and has the attributes that support authorisation to the requested resource. Once this is established you simply provide them the resource. Attributes that are required are stored at the IdP and can be accessed by the SP at the time a request is made to evaluate the authorisation.

With Liberty, the view is that you are not just sharing a resource, but a service. With a service comes additional information, service related metadata specific to principals, preference related data and more. This data has to be stored somewhere, and the simplest solution, and arguably the correct solution, is to place the data at the point at which it is relevant – the SP.

From this we can see that SPs within each framework differ slightly in their requirements, but there are no technical differences from a required functionality perspective. SPs from both frameworks have the ability to issue both authentication and attribute assertions to IdPs. What they do with the information and whether they choose to create local accounts or not is entirely up to them. In both instances they are not privy to any information that the principal has not authorised them to see, and their privacy is further ensured by the use of pseudonyms (targeted identifiers in Shibboleth terms). The only difference is in the support profiles provided for the management of local accounts. Liberty provides these as part of its specification while Shibboleth leaves much of this as implementation level concerns.

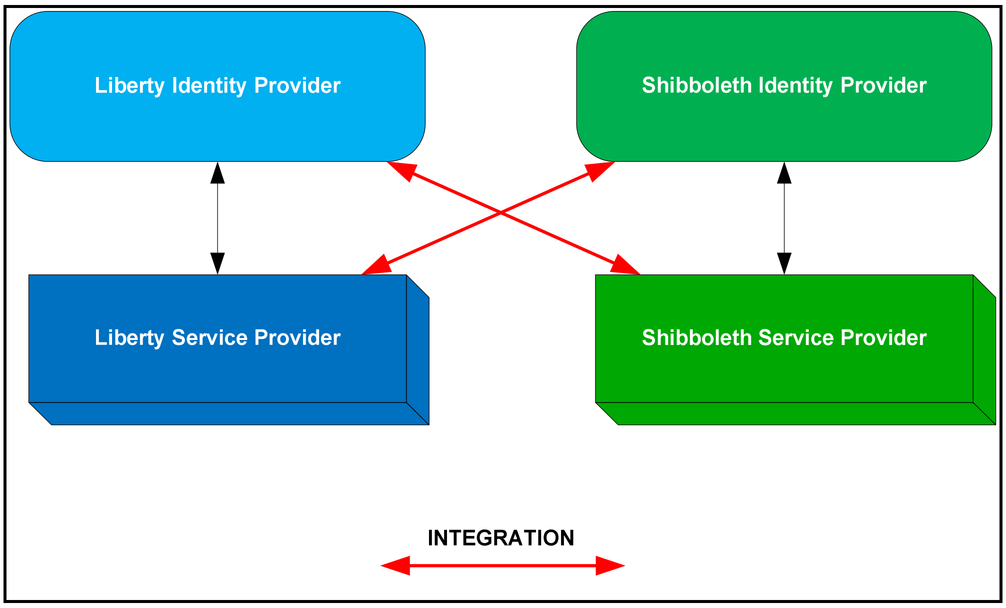
On the other hand, with regards to the roles of the IdP in each framework, both provide virtually identical roles. Ultimately, they provide mechanisms for authenticating users and respond to authentication and attribute assertions from SPs.

6.2 INTEGRATION CONCEPT

We now need to decide what exactly we mean by integration in this context. What are we trying to achieve?

The end goal is to be able to extend a federation beyond the bounds that are defined by its framework – to be able to use different technologies within the same federation. In its simplest form, we would like to have service providers from either framework use identity providers from the other framework.

FIGURE 8 - INTEGRATION CONCEPT


Liberty Identity Provider
Shibboleth Identity Provider
Liberty Service Provider
Shibboleth Service Provider
INTEGRATION

From what we have discerned, both frameworks employ profiles that are remarkably similar. As stated, this is due to their derivation from the same SAML specification.

At the core, the security language used by both frameworks to represent assertions is identical. They both use the SAML Assertion specifications. The SAML specification also guides the protocol interaction between parties, i.e. the request/response procedure. So, for instance, if a SP sends an IdP a SAML Authentication Request, the IdP will respond with a SAML Authentication Response, likewise with an Attribute Assertion Request, the IdP will respond with an Attribute Assertion Response. Both frameworks adhere to these specifications. The profiles that these frameworks use are the collection of protocols, assertions and bindings that represent a particular use case. . Within each framework we have two common profiles, the Browser/POST and the Browser/Artifact profiles. The problem and the main difference between both these architectures are the protocol bindings for these profiles. The bindings are how the messages are sent using particular delivery channels. In this case we are looking specifically at HTTP as the delivery channel, but the messages sent between IdP and SP over HTTP are different, using different wrappings for Assertions, different parameter names for the same information and so on. To an extent, such subtle differences makes one wonder as to the existence of two completely separate frameworks that are, in fact, so similar.

6.3 COMPARING THE BROWSER/POST PROFILES

To begin with, we need to establish the messaging sequence between IdPs and SPs in this profile for both frameworks and compare them. This way we can analyse and isolate the communications that we are interested in, namely the communication between SP and IdP through the User Agent.

It is important to remember that the URL endpoints of the various services are configured using the SAML Metadata specification. This, as mentioned before, is the basis of the federations and contains information such as providerId, public keys and URL endpoint information for members of the federation, both IdPs and SPs.

Let us first look at the Shibboleth sequences. The entire profile is driven by the User Agent. The following table gives us an overview of what occurs.

`U = User Agent, S = Service Provider, I = Identity Provider`

TABLE 6 - SHIBBOLETH BROWSER/POST INTERACTION

#	DIRECTION	MESSAGE

1	U -> S	`Resource request - HTTP GET:`
		`http://sp.com/some-resource`

2	S -> U	`HTTP Redirection [Status 302 Temporary Relocation]:`
		`Location: https://idp.com/shibboleth/SSO?`
		`  target=https://sp.com/some-resource&`
		`  shire=https://sp.com/shibboleth/SSO/POST&`
		`  providerId=https://sp.com/shibboleth/1234567890`

3	U -> I	`Authentication Request - HTTP GET:`
		`https://idp.com/shibboleth/SSO?`
		`  target=http://sp.com/some-resource&`
		`  shire=https://sp.com/shibboleth/SSO/POST&`
		`  providerId=https://sp.com/shibboleth/1234567890`

4	I -> U	`Form based response:`
		`<form`
		`  action="https://sp.com/shibboleth/SSO/POST"`
		`  method="POST">`
		`  <input name="target" type="hidden"`
		`    value="http://sp.com/some-resource" />`
		`  <input name="SAMLResponse" type="hidden"`
		`    value="_BASE64 SAML AUTHENTICATION RESPONSE_" />`
		`  <input type="submit" value="Submit" />`
		`</form>`

5	U -> S	`Submit form, POST to ACS.`

6	S -> U	`HTTP Redirection [Status 302 Temporary Relocation]`
		`Location: http:/sp.com/some-resource`

From the table, Steps 2 to 6 represents the authentication process showing the communication between the SP and the IdP via the User Agent. In step 4, the SAMLResponse parameter is populated with a base64 encoded standard SAML authentication response.

Let us compare this with the Liberty equivalent. The sequence numbers in both interactions have been aligned, and within each profile, with the exception of 1a and 1b from Liberty which is a small step that does not affect the interaction, they are fundamentally the same.

TABLE 7 - LIBERTY BROWSER/POST INTERACTION

#	DIRECTION	MESSAGE
1	U -> S	`Resource request - HTTP GET: http://sp.com/some-resource`
1a	S -> U	`HTTP Redirection [Status 30x No specified code] Location: http://sp.com/ISTS? RelayState=http:/sp.com/some-resource`
1b	U -> S	`HTTP GET: http://sp.com/ISTS?RelayState=http:/sp.com/some-resource`
2	S -> U	`HTTP Redirection [Status 302 Temporary Relocation]: https://idp.com/authn? ProviderID=http://sp.com/liberty/& RelayState=03mhakSms5tMQ0WRDCEzpF7BNcywZa75FwIcSSEPvbko FxaQHCuNnc5yChIdDlWc7JBV9Xbw3avRBK7VFsPl2X& AssertionConsumerServiceID=http://sp.com/liberty/`
3	U -> I	`HTTP GET with above request`
4	I -> U	`Form based response: <form action="https://sp.com/assertion" method="POST"> <input name="RelayState" type="hidden" value="http://sp.com/some-resource" /> <input name="LARES" type="hidden" value="_BASE64 LIB AUTH RESPONSE" /> <input type="submit" value="Submit" /> </form>`
5	U -> S	`Submit form, POST to Assertion Consumer`
6	S -> U	`HTTP Redirection [Status 302 Temporary Relocation] Location: http:/sp.com/some-resource`

It should be noted that Liberty supports form/POST and GET methods for the sequences 1a+1b and 2+3. We have opted to demonstrate just the GET sequence. In step 2, the RelayState parameter is a base64 encoding of the URL to the initially requested resource. The LARES parameter in step 4 is a slight extension of a standard SAML Authentication Response. The extension adds 3 parameters to standard response but these parameters are optional. Complete specifications can be found in [Cantor et al. 2005d].

Comparing Table 6 and Table 7, we can see that the interactions in both profiles follow the same sequence when communicating between the SP and the IdP through the User Agent.

The first communication we are interested in occurs in step 3. The User Agent makes a request to the IdP (based on the redirect from step 2). There are two differences here. The first is that the ‘shire’ attribute in Shibboleth is a URL that points directly to the Assertion Consumer Service at the SP. The Liberty equivalent is AssertionConsumerServiceID, which is an URI ID that references the actual URL within the federation metadata.

The second difference is that within the Shibboleth request, the initially requested resource is mapped directly to the 'target' parameter. Within Liberty the equivalent is the RelayState parameter which is a base64 encoding of the URL for the initially requested resource.

The second interaction of concern is step 5, which is a direct result of the response in step 4. This is where the User Agent submits a form to the SP Assertion Consumer Service. In both instances the form action points directly to the Assertion Consumer URL. There are, again, two differences to be noted. The first is the 'RelayState' and 'target' parameter mismatch, and the second is the parameter name of the response. Shibboleth calls this parameter 'SAMLResponse' while Liberty refers to it as LARES which stands for Liberty Authentication RESponse. As mentioned, the LARES schema is an extension of the SAML Authentication Response schema and should therefore be compatible with the Shibboleth version.

From this we can see that the biggest difference between the frameworks is the protocol bindings. In theory, if we can overcome this difference, it should be possible to integrate these two frameworks.

6.4 ARTIFACT PROFILES

The Artifact Profiles for both frameworks are identical to the Browser/POST profiles with two differences. The first is that instead of a complete authentication response assertion delivered in step 4, a SAML artifact is returned instead.

When using this profile, instead of a form being returned in step 4, a HTTP redirection is used instead. This applies to both frameworks.

The Liberty redirection looks like this:

```
https://sp.com/assertion?SAMLart=[SAML_ARTIFACT]&RelayState=[base64_encoded
_url_to_resource]
```

The Shibboleth equivalent:

```
https://sp.com/shibboleth/SSO/Artifact?SAMLart=[SAML_ARTIFACT]&TARGET=[url_
to_resource]
```

As we can see, they are virtually identical with the exception of the Assertion Consumer Service URL (which is a metadata configuration) and the RelayState/TARGET parameters which have the same purpose yet different names.

Once the artifact is obtained at the Assertion Consumer, the Assertion Consumer makes a request to the IdP to resolve the artifact into an assertion. The good news here is that both frameworks follow the appropriate SAML SOAP specification for making this request. This process is a SOAP message to the IdP containing a standard SAML Authentication Assertion request using an artifact. This means that a SP from either framework can make this kind of a request to and IdP from either framework.

6.5 IDENTITY PROVIDER DISCOVERY

From the Browser/POST profiles of both frameworks, we've left out the process of Identity Provider Discovery. This is a key step in the process and is achieved by the addition of the WAYF in Shibboleth. The Liberty framework has an equivalent profile, the Identity Provider Introduction Profile. Both the WAYF and the Introduction profile are optional components in their respective frameworks and both frameworks allow for the possibility of customised discovery methods.

We should note, however, that while these customised methods need to operate in part at the SP level, they also need to be supported and implemented at the Federation level. Discovery is not specific to individual SPs, but all SPs and IdPs within a federation.

Due to the variability of implementations it would be difficult to begin discussing methods of integration for Identity Provider Discovery. It would have to be assumed that whatever method is employed by a particular Federation, there would be a way to add that facility across both frameworks.

6.6 INTEGRATION OPPORTUNITIES

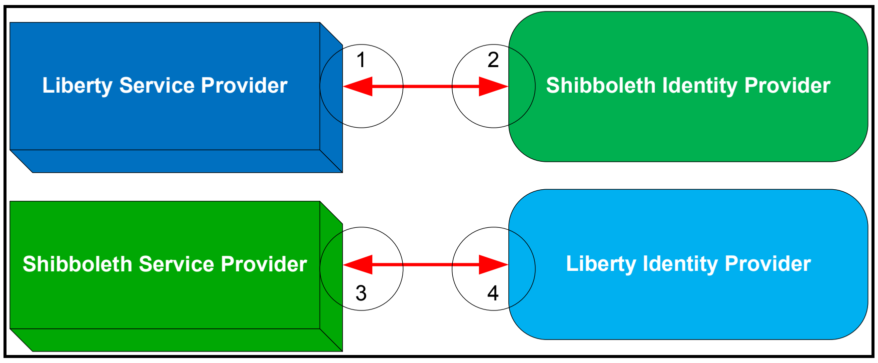
From the comparison of the two profiles, we can see that there are remarkable similarities that can be exploited for the purposes of integration. The message formats between the IdPs and the SPs within each framework are different, but contain essentially the same information that is packaged or represented differently.

An ideal situation would be if both frameworks simply supported the others message binding formats. Once the federation metadata was configured, cross framework federations would be trivial.

In the absence of this, we would have to examine where in the frameworks the least intrusive modifications need to made in order to facilitate message passing between components of different frameworks.

To do this we need only support two additional use cases. The first is that of a Liberty SP using a Shibboleth IdP for authentication and the second is a Shibboleth SP using a Liberty IdP for authentication.

FIGURE 9 - INTEGRATION POINTS


Liberty Service Provider
1
2
Shibboleth Identity Provider
Shibboleth Service Provider
3
4
Liberty Identity Provider

For the first use case, that of a Liberty SP messaging a Shibboleth IdP, there are two message translations that need to take place. The first is the request leaving the SP to the IdP and the second is the response that leaves the IdP to the SP.

Figure 9 identifies four possible locations where message translations can occur. For the first use case, a Liberty SP to Shibboleth IdP, we can translate outgoing messages from the SP at point 1 or point 2. Point 1 would indicate modifying the Liberty SP to translate outgoing messages for compatibility with a Shibboleth IdP. Point 2, would be to have outgoing messages from the SP translated at the just before hitting the IdP and would indicate modifications to the Shibboleth IdP.

Response messages from the IdP to the SP would also require translation. These translations can also occur at either point 1 or point 2. From a design perspective it would make better sense to keep these two functions in the same location, i.e. implement both at either point 1 or point 2. However, given vast range of service types, interactions and implementations SPs could have, it would make more sense to makes these kinds of modifications at the IdP side of the framework. The IdP has a limited range of services and scenarios to cater for and is arguably the simpler of the two components.

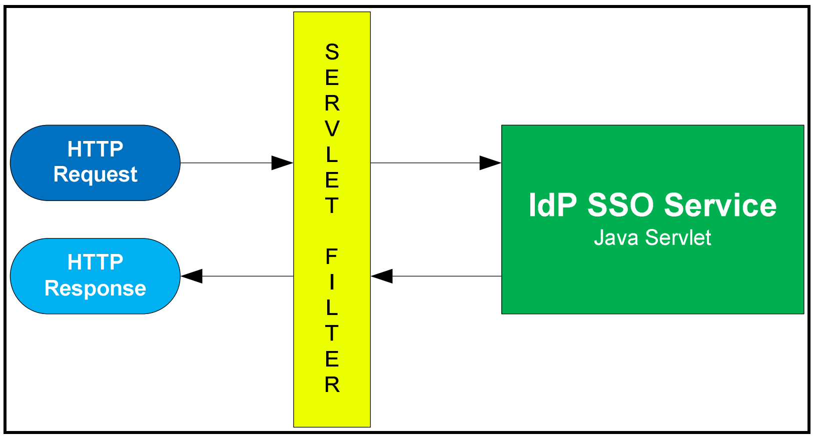
The second use case has the same message translation requirements, except we are using a Shibboleth SP against a Liberty IdP. The same scenarios from the first use case apply here. There are two points where messages can be translated, points 3 and 4. Again, it would appear to make more sense to localise where translations occur and given the relative complexity of each component, it makes more sense to have these occur at the IdP end.

Performing these translations would be a technology concern as, depending on which implementation of these specifications you are using, the range of solutions available will differ substantially.

As a theoretical example however, we can take a brief look at the Internet2 implementation of the Shibboleth IdP. The IdP is a Java Web Application that has a range of Java Servlets that perform the various functionality associated with the IdP, such as handing SAML Authentication Requests, SAML Authentication responses, Attribute Requests and Responses etc..

This particular configuration provides us with some interesting capabilities to handle our translations, namely the use of the interceptor pattern via the use of Java Servlet Filters.

FIGURE 10 - INTERCEPTOR PATTERN FOR TRANSLATION



The servlet filter has the ability to intercept HTTP requests as they enter the servlet and the response just as it leaves the servlet. The filter has complete access to the entire HTTP request and has the ability to alter the incoming request body, parameters and headers. After the servlet has processed the incoming requests and sends the response, the filter has the ability to alter the response body and headers.

Servlet filters are easy modules to add to a request processing pathway. Given this functionality, it provides ideal location to add functionality to rewrite both responses and requests to cater for the differing message bindings of the two SPs.

7 CONCLUSION

We have explained the importance of the growth of Federated Identity Architectures and the reasoning that supports why limiting federations to particular technologies is undesirable. To this end we have extracted and examined, in detail, the relevant specifications of each framework that we have been investigating.

We have examined both the Shibboleth and Liberty Alliance Federated Identity Architectures and deconstructed the relevant profiles, protocols and bindings that are used to transmit assertions between security domains. We have shown that having each framework based upon a common specification shares common, core interactions but that these are subtly altered to satisfy domain specific requirements.

In addition we have seen that the gaps in the specifications that have been left as implementation concerns create unknown quantities in terms of integration. This means that for different types of implementations, different integration solutions will have to be employed. As a result the integration opportunities need to be identified at the specification level with integration left as an implementation concern.

We have, however, endeavoured to provide a simple example of how such integration could be done, but the significant result of the work has been the identification of the message binding differences between both frameworks and the profile commonalities. With this information we are able to isolate exactly what needs to be done at the message binding level to enable components from the different frameworks to interact. Successfully implementing a solution would prove that the federation can extend beyond its technological boundaries.

Further work along this line could be performed by establishing a small test federation using both frameworks and selecting an easy set of implementations to work with.

REFERENCES

1. Bhargav-Spantzel, A., Camenisch, J., Gross, T., Sommer, D., (2006) “User centricity: a taxonomy and open issues.” *Proceedings of the second ACM workshop on Digital identity management*, Alexandria, Virginia, USA: ACM, pp. 1-10
2. Cantor, S. (ed.), Kemp, J. (ed.), Philpott, R. (ed.), Maler, E. (ed.) (2005a) “Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0” *Organisation for the Advancement of Structured Information Standards*, URL: http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf (accessed on 10 September 2009)
3. Cantor, S. (ed.), Moreh, J. (ed.), Philpott, R. (ed.), Maler, E. (ed.) (2005b) “Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0” *Organisation for the Advancement of Structured Information Standards*, URL: http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf (accessed on 10 September 2009)
4. Cantor, S. (ed.), (2005c) “Shibboleth Architecture Protocols and Profiles.” *Internet2 Middleware Initiative* URL: http://shibboleth.internet2.edu/docs/internet2-mace-shibbolletharch-protocols-200509.pdf (accessed on 20 May 2008)
5. Cantor, S. (ed.), Kemp, J. (ed.), Champagne, D. (ed.), (2005d) “Liberty ID-FF Bindings and Profiles Specification” *Liberty Alliance Project* URL: http://www.projectliberty.org/liberty/content/download/319/2369/file/draft-liberty-idff-bindings-profiles-1.2-errata-v2.0.pdf (accessed at 24 September 2009)
6. Davis, P. (ed.), (2005) “Liberty Metadata Description and Discovery Specification 1.1” *Liberty Alliance Project* URL: http://www.projectliberty.org/liberty/content/download/1224/7973/file/liberty-metadata-v1.1.pdf (accessed at 12 September 2009)
7. Dhamija, R., Dusseault, L., (2008) “The Seven Flaws of Identity Management: Usability and Security Challenges.” *Security & Privacy, IEEE* Vol. 6, Iss. 2 pp. 24-29
8. Eap, T.M., Hatala, M., Gasevic, D., (2007) “Enabling User Control with Personal Identity Management.” *Services Computing, 2007. SCC 2007. IEEE International Conference on*, pp. 60-67
9. El Maliki, T., Seigneur, J., (2007), “A Survey of User-centric Identity Management Technologies.” *Emerging Security Information, Systems, and Technologies, 2007. SecureWare 2007. The International Conference on,* pp. 12-17
10. Fielding, R., Irvine, Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., (1999) “RFC2616 - Hypertext Transfer Protocol – HTTP/1.1” *The Internet Engineering Task Force (IETF)* URL: http://www.ietf.org/rfc/rfc2616.txt (accessed on 16 October 2009)

11. Fragoso-Rodriguez, U., Laurent-Maknavicius M., Incera-Dieguez J., (2006) "Federated Identity Architectures." *Proceedings of the 1st Mexican Conference on Informatics Security 2006* URL: http://www-lor.int-evry.fr/~maknavic/articles/mlaurent-mcis06.pdf (accessed on 15 May 2008)
12. Fujiwara, S., Komura, T., Okabe, Y., (2007) "A Privacy Oriented Extension of Attribute Exchange in Shibboleth." *Applications and the Internet Workshops, 2007. SAINT Workshops 2007. International Symposium on,* pp. 28-32
13. Hansen, M., Schwartz, A., Cooper, A., (2008) "Privacy and Identity Management." *Security & Privacy, IEEE* Vol. 6, Iss. 2 pp. 38-45
14. Hughes, J., Cantor, S., Hodges, J., Hirsch, F., Mishra, P., Phipott, R., Maler, E., (2005) "Profiles for the OASIS Security Assertion Markup Language" *Organisation for the Advancement of Structured Information Standards* URL: http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf (accessed on 16 October 2009)
15. Landau, S., eds., (2007) "Liberty ID-WSF Web Services Security and Privacy Overview" *Liberty Alliance Project* URL: http://www.projectliberty.org/resource_center/specifications/liberty_alliance_id_wsf_2_0_specifications_including_errata_v1_0_updates (accessed on 15 May 2008)
16. Liberty Project Contributors, (2003) "Liberty and 3rd Party Identity Systems White Paper" *The Liberty Project* URL: http://www.projectliberty.org/liberty/files/whitepapers/liberty_and_3rd_party_identity_systems_white_paper_pdf (accessed on 1 June 2008)
17. Mont, M. et al., (2002), "Identity Management: a Key e-Business Enabler" *Trusted Eservices Laboratory, HP Laboratories* Bristol URL: http://www.hpl.hp.com/techreports/2002/HPL-2002-164.pdf (accessed on 12 May 2008)
18. Needleman, M., (2004) "The Shibboleth Authentication/Authorization System." *Serials Review* Vol. 30, Iss. 3 pp. 252-253
19. Peyton, L., Hu, J., Doshi, C., Seguin, P., (2007) "Addressing Privacy in a Federated Identity Management Network for EHealth." *Management of eBusiness, 2007. WCMeB 2007. Eighth World Congress on the*
20. Poon, S., Swatman, P., (1999) "An exploratory study of small business Internet commerce issues." *Information & Management* Vol. 35, Iss.1, pp. 9-18
21. Scavo, T. (ed.), Cantor, S. (ed.), (2005) "Shibboleth Architecture – Technical Overview" *Internet2 Middleware Initiative* URL: http://shibboleth.internet2.edu/docs/draft-mace-shibboleth-tech-overview-latest.pdf (accessed on 8 Sep 2009)
22. Shim, S.S.Y., Bhalla G., Pendyala, V., (2005) "Federated identity management." *Computer* Vol. 38, Iss. 12 pp. 120-122

23. Spence, D., Geddes, N., Jensen, J., Richards, A., Viljoen, M., Martin, A., Dovey, M., Norman, M., Tang, K., Trefethen, A., Wallom, D., Allan, D., Meredith, D., (2006) "ShibGrid: Shibboleth Access for the UK National Grid Service," *Second IEEE International Conference on e-Science and Grid Computing* e-science, pp.75, 2006

24. Steel, C., Nagappan, R., Lai, R., (2005) "*Core security patterns : best practice and strategies for J2EE, Web Services and identity management*", Prentice Hall PTR, New Jersey

25. Wason, T., eds., (2003) "Liberty ID-FF Architecture Overview version 1.2" *Liberty Alliance Project* URL: http://www.projectliberty.org/resource_center/specifications/liberty_alliance_id_ff_1_2_specifications (accessed on 15 May 2008)

APPENDICES

A. SAML Authentication Assertion

```
<saml:Assertion
  xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
  MajorVersion="1" MinorVersion="1"
  AssertionID="a75adf55-01d7-40cc-929f-dbd8372ebdfc"
  IssueInstant="2004-12-05T09:22:02Z"
  Issuer="https://idp.example.org/shibboleth">
  <saml:Conditions
    NotBefore="2004-12-05T09:17:02Z"
    NotOnOrAfter="2004-12-05T09:27:02Z">
    <saml:AudienceRestrictionCondition>
      <saml:Audience>http://sp.example.org/shibboleth</saml:Audience>
    </saml:AudienceRestrictionCondition>
  </saml:Conditions>
  <saml:AuthenticationStatement
    AuthenticationInstant="2004-12-05T09:22:00Z"
    AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
    <saml:Subject>
      <saml:NameIdentifier
        Format="urn:mace:shibboleth:1.0:nameIdentifier"
        NameQualifier="https://idp.example.org/shibboleth">
        3f7b3dcf-1674-4ecd-92c8-1544f346baf8
      </saml:NameIdentifier>
      <saml:SubjectConfirmation>
        <saml:ConfirmationMethod>
          urn:oasis:names:tc:SAML:1.0:cm:bearer
        </saml:ConfirmationMethod>
      </saml:SubjectConfirmation>
    </saml:Subject>
  </saml:AuthenticationStatement>
</saml:Assertion>
```

B. SAML Authentication Assertion with Artifact

```
<saml:Assertion
  xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
  MajorVersion="1" MinorVersion="1"
  AssertionID="003c6cc1-9ff8-10f9-990f-004005b13a2b"
  IssueInstant="2004-12-05T09:22:05Z"
  Issuer="https://idp.example.org/shibboleth">
  <saml:Conditions
    NotBefore="2004-12-05T09:17:05Z"
    NotOnOrAfter="2004-12-05T09:27:05Z">
    <saml:AudienceRestrictionCondition>
     <saml:Audience>http://sp.example.org/shibboleth</saml:Audience>
    </saml:AudienceRestrictionCondition>
  </saml:Conditions>
  <saml:AuthenticationStatement
    AuthenticationInstant="2004-12-05T09:22:00Z"
    AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
    <saml:Subject>
      <saml:NameIdentifier
        Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress"
        NameQualifier="https://idp.example.org/shibboleth">
        user@idp.example.org
      </saml:NameIdentifier>
      <saml:SubjectConfirmation>
        <saml:ConfirmationMethod>
          urn:oasis:names:tc:SAML:1.0:cm:artifact
        </saml:ConfirmationMethod>
      </saml:SubjectConfirmation>
    </saml:Subject>
  </saml:AuthenticationStatement>
</saml:Assertion>
```

C. SAML Attribute Assertion

```
<saml:Assertion
  xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
  MajorVersion="1" MinorVersion="1"
  AssertionID="a144e8f3-adad-594a-9649-924517abe933"
  IssueInstant="2004-12-05T09:22:05Z"
  Issuer="https://idp.example.org/shibboleth">
  <saml:Conditions
    NotBefore="2004-12-05T09:17:05Z"
    NotOnOrAfter="2004-12-05T09:52:05Z">
    <saml:AudienceRestrictionCondition>
      <saml:Audience>http://sp.example.org/shibboleth</saml:Audience>
    </saml:AudienceRestrictionCondition>
  </saml:Conditions>
  <saml:AttributeStatement>
    <saml:Subject>
      <saml:NameIdentifier
        Format="urn:mace:shibboleth:1.0:nameIdentifier"
        NameQualifier="https://idp.example.org/shibboleth">
        3f7b3dcf-1674-4ecd-92c8-1544f346baf8
      </saml:NameIdentifier>
    </saml:Subject>
    <saml:Attribute
      AttributeName="urn:mace:dir:attribute-def:eduPersonPrincipalName"
      AttributeNamespace="urn:mace:shibboleth:1.0:attributeNamespace:uri">
      <saml:AttributeValue Scope="example.org">
        userid
      </saml:AttributeValue>
    </saml:Attribute>
  </saml:AttributeStatement>
</saml:Assertion>
```

D. SAML Signed Assertion

```
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <ds:SignedInfo>
    <ds:CanonicalizationMethod
      Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
    <ds:SignatureMethod
      Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
    <ds:Reference URI="#c7055387-af61-4fce-8b98-e2927324b306">
      <ds:Transforms>
        <ds:Transform
          Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
signature"/>
        <ds:Transform
          Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
          <InclusiveNamespaces
            PrefixList="#default saml samlp ds xsd xsi"
            xmlns="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        </ds:Transform>
    </ds:Transforms>
    <ds:DigestMethod
      Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <ds:DigestValue>TCDVSuG6grhyHbzhQFWFzGrxIPE=</ds:DigestValue>
    </ds:Reference>
  </ds:SignedInfo>
  <ds:SignatureValue>
x/GyPbzmFEe85pGD3c1aXG4Vspb9V9jGCjwcRCKrtwPS6vdVNCcY5rHaFPYWkf+5
EIYcPzx+pX1h43SmwviCqXRjRtMANWbHLhWAptaK1ywS7gFgsD01qjyen3CP+m3D
w6vKhaqledl0BYyrIzb4KkHO4ahNyBVXbJwqv5pUaE4=
  </ds:SignatureValue>
  <ds:KeyInfo>
    <ds:X509Data>
      <ds:X509Certificate>
MIICyjCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgakxCzAJBgNVBAYTAlVT
MRIwEAYDVQQIEwlXaXNjb25zaW4xEDAOBgNVBAcTB01hZGlzb24xIDAeBgNVBAoT
F1VuaXZlcnNpdHkgb2YgV2lzY29uc2luMSswKQYDVQQLEyJEaXZpc2lvbiBvZiBJ
bmZvcm1hdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgQ0Eg
LS0gMjAwMjA3MDFBMB4XDTAyMDcyNjA3Mjc1MVoXDTA2MDkwNDA3Mjc1MVowgYsx
CzAJBgNVBAYTAlVTMREwDwYDVQQIEwhNaWNoaWdhbjESMBAGA1UEBxMJQW5uIEFy
Ym9yMQ4wDAYDVQQKEwVVQ0FJRDEcMBoGA1UEAxMTc2hpYjEuaW50ZXJuZXQyLmVk
dTEnMCUGCSqGSIb3DQEJARYYcm9vdEBzaGliMS5pbnRlcm5ldDIuZWR1MIGfMA0G
CSqGSIb3DQEBAQUAA4GNADCBiQKBgQDZSAb2sxvhAXnXVIVTx8vuRay+x50z7GJj
IHRYQgIv6IqaGG04eTcyVMhoekE0b45QgvBIaOAPSZBl13R6+KYiE7x4XAWIrCP+
c2MZVeXeTgV3Yz+USLg2Y1on+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7O27rhRjE
pmqOIfGTWQIDAQABox0wGzAMBgNVHRMBAf8EAjAAMAsGA1UdDwQEAwIFoDANBgkq
hkiG9w0BAQQFAAOBgQBfDqEW+OI3jqBQHIBzhujN/PizdN7s/z4D5d3pptWDJf2n
qgi7lFV6MDkhmTvTqBtjmNk3No7v/dnP6Hr7wHxvCCRwubnmIfZ6QZAv2FU78pLX
8I3bsbmRAUg4UP9hH6ABVq4KQKMknxu1xQxLhpR1ylGPdiowMNTrEG8cCx3w/w==
      </ds:X509Certificate>
    </ds:X509Data>
  </ds:KeyInfo>
</ds:Signature>
```

E. SOAP SAML REQUEST WITH ARTIFACT

```
<?xml version="1.1" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
```

```
    <samlp:Request
      xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
      MajorVersion="1" MinorVersion="1"
      RequestID="f81d4fae-7dec-11d0-a765-00a0c91e6bf6"
      IssueInstant="2004-12-05T09:22:04Z">
      <samlp:AssertionArtifact>
        AAEwGDwd3Z7Fr1GPbM82Fk2CZbpNB1dxD+t2Prp+TDtqxVA78iMf3F23
      </samlp:AssertionArtifact>
    </samlp:Request>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

F. SOAP SAML RESPONSE

```
<?xml version="1.1" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <samlp:Response
      xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
      MajorVersion="1" MinorVersion="1"
      Recipient="https://sp.example.org/shibboleth/SSO/Artifact"
      ResponseID="00099cf1-a355-10f9-9e95-004005b13a2b"
      InResponseTo="f81d4fae-7dec-11d0-a765-00a0c91e6bf6"
      IssueInstant="2004-12-05T09:22:05Z">
      <samlp:Status>
        <samlp:StatusCode Value="samlp:Success"/>
      </samlp:Status>
      <saml:Assertion
        xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
        MajorVersion="1" MinorVersion="1"
        AssertionID="a75adf55-01d7-40cc-929f-dbd8372ebdfc"
        IssueInstant="2004-12-05T09:22:02Z"
        Issuer="https://idp.example.org/shibboleth">
        <saml:Conditions
          NotBefore="2004-12-05T09:17:02Z"
          NotOnOrAfter="2004-12-05T09:27:02Z">
          <saml:AudienceRestrictionCondition>
            <saml:Audience>http://sp.example.org/shibboleth</saml:Audience>
         </saml:AudienceRestrictionCondition>
        </saml:Conditions>
        <saml:AuthenticationStatement
          AuthenticationInstant="2004-12-05T09:22:00Z"
          AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
          <saml:Subject>
            <saml:NameIdentifier
              Format="urn:mace:shibboleth:1.0:nameIdentifier"
              NameQualifier="https://idp.example.org/shibboleth">
              3f7b3dcf-1674-4ecd-92c8-1544f346baf8
            </saml:NameIdentifier>
            <saml:SubjectConfirmation>
              <saml:ConfirmationMethod>
                urn:oasis:names:tc:SAML:1.0:cm:bearer
              </saml:ConfirmationMethod>
            </saml:SubjectConfirmation>
          </saml:Subject>
        </saml:AuthenticationStatement>
      </saml:Assertion>
    </samlp:Response>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```