Shibboleth-Architecture DRAFT v05

draft-internet2-shibboleth-architecture-05.html

May 2, 2002

Marlena Erdos (marlena@us.ibm.com) and Scott Cantor (cantor.2@osu.edu)

comments to: mace-shibboleth-comments@internet2.edu

Abstract

Shibboleth, an Internet2/MACE project with intellectual and financial support from IBM, is developing architectures, frameworks, and practical technologies to support inter-institutional sharing of resources that are subject to access controls. This paper presents the Shibboleth architecture for the secure exchange of interoperable authorization information that can be used in access control decision-making. The paper will present a high-level view of the interaction between sites and will provide a detailed behavioral description of model components and message exchange formats and protocols. One difference between Shibboleth and other efforts in the access control arena is Shibboleth's emphasis on user privacy and control over information release.

1 Introduction

This paper describes the concepts and model of Shibboleth. It also describes and specifies Shibboleth messages and protocols and the behavior or Shibboleth components. There are two intended audiences: technically-minded readers who want to get a "sense" of Shibboleth; and those who want to understand Shibboleth in detail, perhaps as a prelude to their own implementation of one or more Shibboleth components.

Thus the document has two overall topics: an overview of the problem space and the high-level architecture; and an in-depth discussion of the Shibboleth components. The first sections should, among other things, provide motivation for "why Shibboleth?" and provide a good conceptual lead-in to the component sections. This document will also provide a rationale for various aspects of the design.

Below are descriptions in brief of the sections that follow this introduction:

Section 2: Overview

In the overview, we talk about the problem space that Shibboleth is trying to address. And we present the "solution" at a high level.

Section 3: High Level Architecture

This section introduces the Shibboleth architecture first at a simplified level and then more completely. The section provides the background needed to understand the components and message flows on a technical basis.

Section 4: Relationship to Related Initiatives

This section discusses industry standards that relate to or influence Shibboleth.

Section 5: Shibboleth Components in Detail

This section will discuss each of the components of Shibboleth in some detail.

Section 6: Specification of Shibboleth Messages and Protocols

This section provides the specific formats and exchanges of messages that are used in Shibboleth.

Acknowledgements

This section lists the Shibboleth contributors and their affiliations.

2 Overview of Shibboleth

2.1 The Problem Space

Within the worlds of both academia and business, there is growing interest in collaboration and resource sharing among institutions. In the case of universities (the focus of Shibboleth), current sharing "solutions" employ fairly primitive mechanisms for identifying legitimate users to a partner site providing resources.

In some cases, the "identifier" is merely IP address. This solution suffers from being easy to defeat (IP addresses are relatively easy to spoof) and from lack of flexibility, forcing off-campus and mobile users to go 'through' the campus network to get access.

In other cases, each user is given a new name and password to be used when accessing the partner's site. If the number of eligible users is small, then it is not a large burden on the partner's system administrators to create and maintain separate identities for each 'foreign' user. But when the number of users gets large (as it often does in the university case), the burden of managing identities for foreign users can be high. The resource provider has to get information on new users (new students, faculty, and staff) and on those leaving (hopefully graduating students!).

The resource provider winds up in the role of system administrator for the university's users, without actually relieving the university of system administration it has to do anyway. This sort of administrative entanglement is not generally considered a positive feature of collaboration, and often inhibits it entirely.

Figure 1 illustrates this situation.

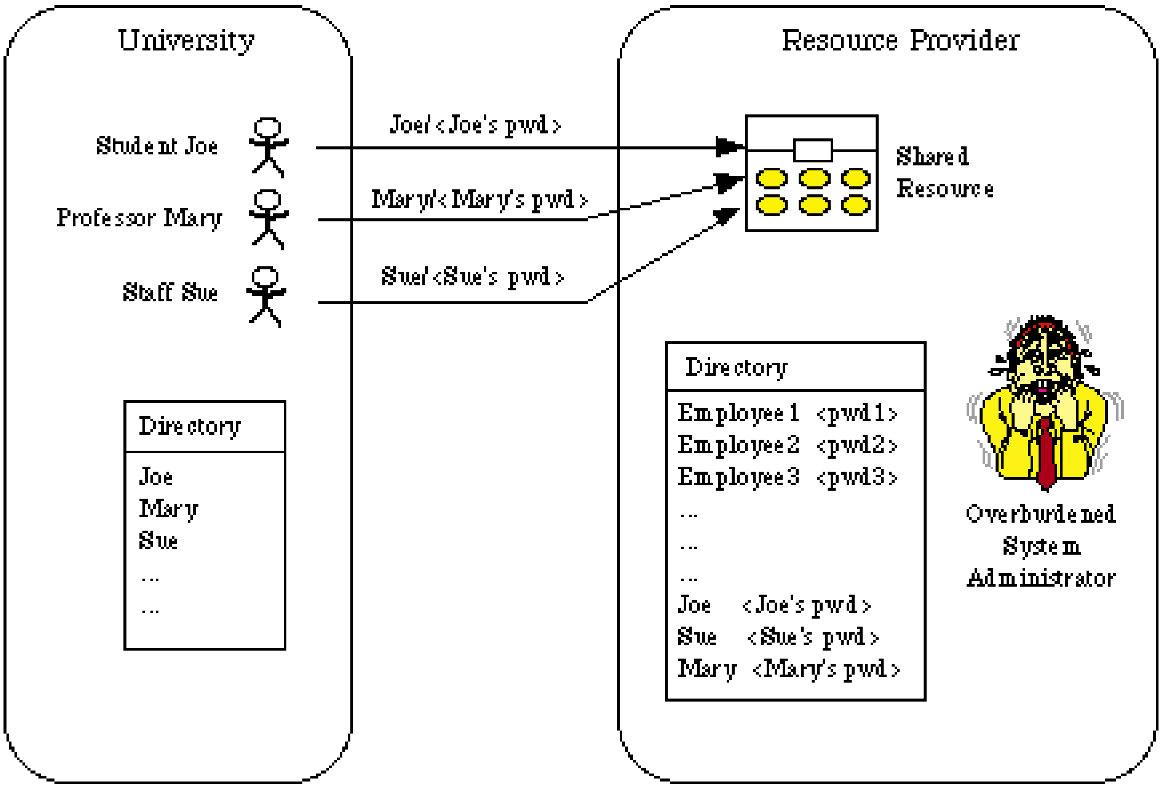

Figure 1: Users registered separately

The burden on the university user trying to access the provider's resource is also increased. He or she has yet another name and password to remember. Global sign-on solutions can help with this, but most universities (and companies) would like to reduce the number of identities that have to be managed for each user.

The other option, a single identity and password that is used by each and every user of the university when accessing the resource provider, suffers from the problem of lack of accountability. If information or a service is misused, the provider (and the university) has little means of ascertaining who specifically misused the resource. This is a very serious downside of the "one identity for all" approach.

Figure 2 illustrates this situation.

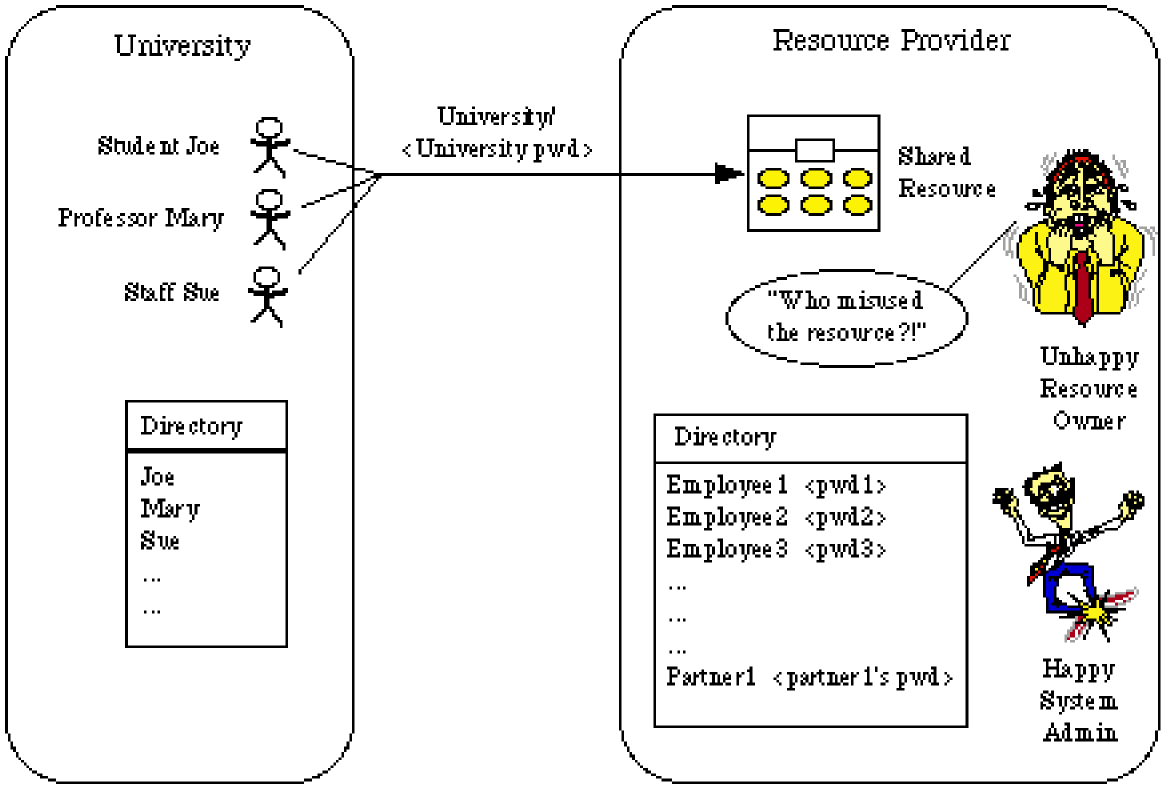

Figure 2: Single name & password for all users

2.1.1 PKI

Public key infrastructure solutions appear more sophisticated than password-based approaches, but still cause an administrative burden at the provider site. (The presumption in the following discussion is that each foreign user has a distinct PKI identity.) When a foreign user tries to access a resource, the provider must not only check the signature (not a big deal) but also check for revocation. Revocation is typically handled at the users' origin site so there is the problem of distributing revocation lists from the university to the resource provider.

Again, there is an administrative entanglement where the provider has to be aware of the personnel changes at the university (though now just deletions), and where the university may be revealing information about students, faculty, and staff not involved in the partnership.

Further, the burden on the accessing university can be high if a PKI is not already in place. Setting up and administering a PKI is not a trivial task: registering users, distributing keys, and providing education on user protection of private keys or smart cards all take up a fair bit of administrative and employee time and effort.

The real problem with PKI, however, is that the provider must still maintain lists of which users at the university are actually authorized to access resources. The university may have thousands of users that have PKI identities though only scores may be involved in any given partnership. The provider must get continuous updates on which users are joining the partnership project and which are leaving.

2.1.2 Identity and Privacy

Both PKI and individual name/password schemes inherently involve identity. This not only causes administrative entanglements and headaches, as demonstrated, but also may be contrary to users' desire for privacy in some of their online interactions. In the business community, an employee's desire for privacy is often secondary to the business's need for accountability; however universities in some countries (including the US) are legally required to protect the privacy of their students. These legal requirements sometimes dictate that a student be able to access learning resources without revealing their identity.

Clearly, an access-control scheme that is based on identity cannot serve a university's legal need to provide privacy for its students.

2.2 The High-Level Solution

Shibboleth aims to detangle the management of users at cooperating institutions by "federating" administration. In federated administration, a resource provider leaves the administration of user identities and attributes to the users' origin site. The resource provider relies on the origin site to provide attributes about a user (possibly but not necessarily including a username) that the provider can use in making an access control decision when the user attempts to use a resource. Typical attributes in Shibboleth (expressed informally) include "member of university community@foo.edu", "faculty member@foo.edu", student@foo.edu, etc. Users are registered only at their origin site, and not at each resource provider.

Shibboleth, then, is a system for securely transferring attributes about a user from the user's origin site to a resource provider site. The components and message flows described later detail this process. Shibboleth assumes that users employ browsers and that the resources are accessible via standard browser technologies. Shibboleth is also a system for allowing user choice in what information gets released about the user and to which site. Thus, the job of balancing access and privacy lies ultimately with the user, where it belongs.

2.2.1 Addressing Privacy Concerns

Since Shibboleth is concerned with user privacy, an important element of the Shibboleth architecture is the component that releases information about users. This is the Attribute Authority (AA). Each origin site (i.e. a site with administrative authority over users who access resources at remote providers) has its own AA. The AA's job is to provide attributes about a user to a resource provider. But the AA also has the responsibility of providing a means for users to specify exactly which of their allowable attributes gets sent to each site they visit.

For example, faculty member Mary Smith at Brown University may be a participant of a multi-institution research project whose documents and resources are located at Ohio State. And she may wish for personal reasons to visit a multiple sclerosis site hosted at John Hopkins. In the case of the research project, she may wish and need to send her name to get access to project resources; in the case of the multiple sclerosis site she may need only to send her affiliation (i.e. faculty member at Brown University), and she may want to exclude the release of her name.

While the Shibboleth architecture doesn't specify all of the implementation of the AA, it does specify a way for an Attribute Authority to structure "attribute release policies" so that each user can choose what attributes get released and to where they get released. We expect that AAs will offer a web page that will allow a user to control attribute release policies for the sites they customarily visit, and to choose intelligent default policies for new sites.

Of course, AAs and attribute release policies only work with resource provider sites that have implemented Shibboleth's protocols for acquiring attributes. We expect that the benefits of detangled, federated administration will interest a growing number of commercial resource providers as well as universities that themselves host resources. Our hope and expectation is that Shibboleth will address both providers' economic interests (through federated administration) and users' privacy concerns (through attribute release policies).

3 High Level Architecture

3.1 Slightly Simplified Architecture

In Shibboleth, when a user at a browser attempts to access a resource at a destination site, the "Shibbolized" web server will 'notice' that it doesn't have attributes about the user. The part of the web server that obtains and caches attributes is called the "Shibboleth Attribute Requester" or SHAR (rhymes with 'tar') for short.

The SHAR will then interact with the Attribute Authority (AA) at the origin site to get attributes about the user. The AA may store attributes directly, or more likely will obtain them from the institution's LDAP directory or other institutional database. Shibboleth doesn't specify how the AA stores or acquires user attributes. That is up to each origin site to decide on and implement. The AA must additionally have access to the user's "attribute release policy" for the destination site, in order to decide what attributes to send back to the SHAR. Again, Shibboleth doesn't specify how attribute release policies are stored and managed.

We call the attribute request that the SHAR sends to the AA an "AQM" for "attribute query message". The response that the AA sends to the SHAR is an "ARM" for "attribute response message".

The SHAR, once it has these attributes, will send them on to the manager of the resource the user is trying to access. The resource manager (RM) will then make an access control decision based on the user's attributes, and either grant or deny the user's request. If the user is simply trying to access a static web page or a typical application, this RM may be the web server itself. In the case where the user is attempting a more complex action (say updating experimental results or transferring grant money), the RM may sit "behind" the web server on a separate machine.

Figure 3 shows the user, web server, SHAR, and AA.

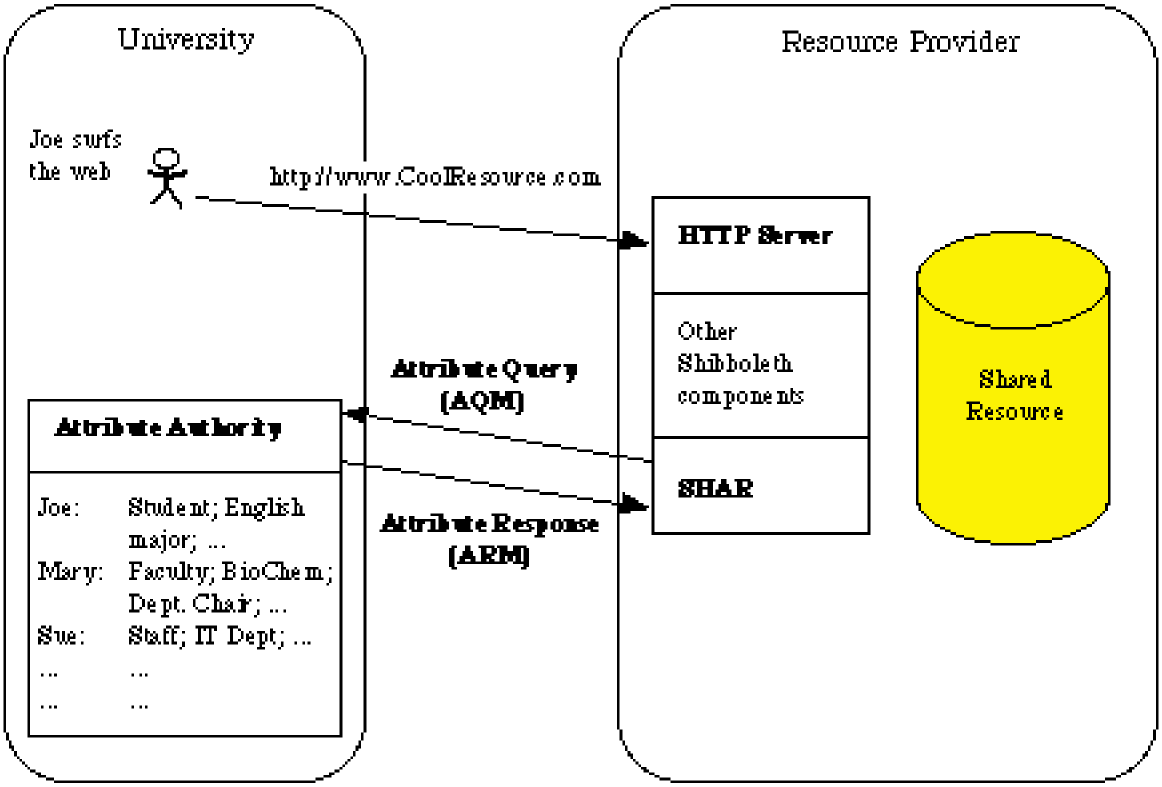

Figure 3: SHAR requests user attributes from the AA

3.2 The Architecture in More Detail

When a user contacts a destination, the SHAR would like to get attributes about the user. But on what basis can it ask? Though Shibboleth allows for the case in which a user authenticates directly to a destination with a digital signature and client certificate (thus providing a name to the SHAR), its primary emphasis is on browser users without PKI credentials.

Since Shibboleth wishes to obviate the need for a user to "log in" at each destination, and since Shibboleth is very concerned with privacy, this presents a dilemma: How can the SHAR ask for the user's attributes when it doesn't typically have the user's name?

Shibboleth solves this problem by associating a "handle" with the user. The handle lets a SHAR ask for attributes about the user, but the handle doesn't give away the user's name (or any other information that would identify the user to the destination).

Shibboleth supports two different ways for a user without PKI credentials to become securely associated with a handled transmitted from the origin site to a destination. These two approaches, and the client certificate case, are described below, each in their own sub-section.

3.2.1 Direct Access to Destination Site

Very typically, a Shibboleth user will simply surf to a site they are interested in. In this case, the destination site not only doesn't have attributes about the user, it doesn't have any identifying information. It doesn't even know what the user's origin site is.

Before the SHAR component of the web server can ask for attributes about the browser user, the "handle" for getting information about this user needs to be obtained. The part of the "Shibbolized" web server that manages the process of acquiring a handle is called the SHIRE – Shibboleth Indexical Reference Establisher.

What the heck is "Indexical Reference"?

"Indexical reference" refers to being able to "point at" a user without being able to otherwise name or describe the user. The web server, by virtue of being contacted by a browser user, has a "pointing" reference to that user. The SHIRE (which is co-located with the web server) uses that connection with the browser to help initiate the process of securely getting a handle for the user. The rest of this section discusses this process, also shown in Figure 4.

Getting the handle

When a user first surfs to a Shibboleth-protected site, it is the SHIRE that takes over. The SHIRE's goal in this case is to determine the name and location of the user's Handle Service (HS), and ask it to send back a handle for the user. The Handle Service is responsible for making sure the user is authenticated locally at the origin site, and for creating a handle that can be used to retrieve attributes about the user. The HS will be discussed in further detail later in this section.

The SHIRE employs an important helper that does most of the initial work. It is called the "Where Are You From? (WAYF)" service.

The WAYF is a component that knows the name and location of the Handle Service for each origin site that is participating in Shibboleth. Its primary job is to map an origin site name (like harvard.edu) to the HS information for that site. The WAYF's other responsibility is to ask the user's HS to send a handle for the user to the SHIRE. Conceptually, the WAYF is a network-based, possibly replicated service that lives "somewhere" in the Internet. However, in practice, WAYF information can be held locally by each SHIRE (for example, in a database, or even a file; the specifics are up to the destination site).

The SHIRE uses the WAYF by redirecting the user's browser to it. The WAYF will interact with the user, asking "Where are you from?" The user then enters the name of his or her institution (e.g. Brown, MIT). The WAYF provides the mapping between the name of the user's institution and the URL (and identity) of that site's HS.

The WAYF then acts on behalf of the SHIRE and asks the HS to create a handle for the user and send it to the SHIRE. (This request is formally known as the "Attribute Query Handle Request" but we'll generally call it simply the "handle request"). The WAYF redirects the user's browser to the HS, appending the SHIRE's "handle acceptance" URL and the originally requested target URL as query string parameters.

Once the HS receives the Attribute Query Handle Request, it will interact with the user to get them "logged in" to the origin site's authentication system (if the user hasn't already done so). The HS may then optionally interact with the Attribute Authority, asking it to create an attribute query handle. Alternatively, depending on the implementation, the HS may create the handle on its own.

In either case, the HS passes the handle along with some additional information (altogether called a "handle package") back to the user's browser inside an HTML form that posts the data back to the destination SHIRE. The additional data in the handle package includes the location of the AA at which the handle will be usable, and information that helps the SHIRE detect attempts at impersonation of the legitimate user. A discussion of the handle package and how it helps the SHIRE defeat impersonation attacks can be found in section 5.2.

Figure 4 shows the entire process of getting a handle. The steps are as follows:

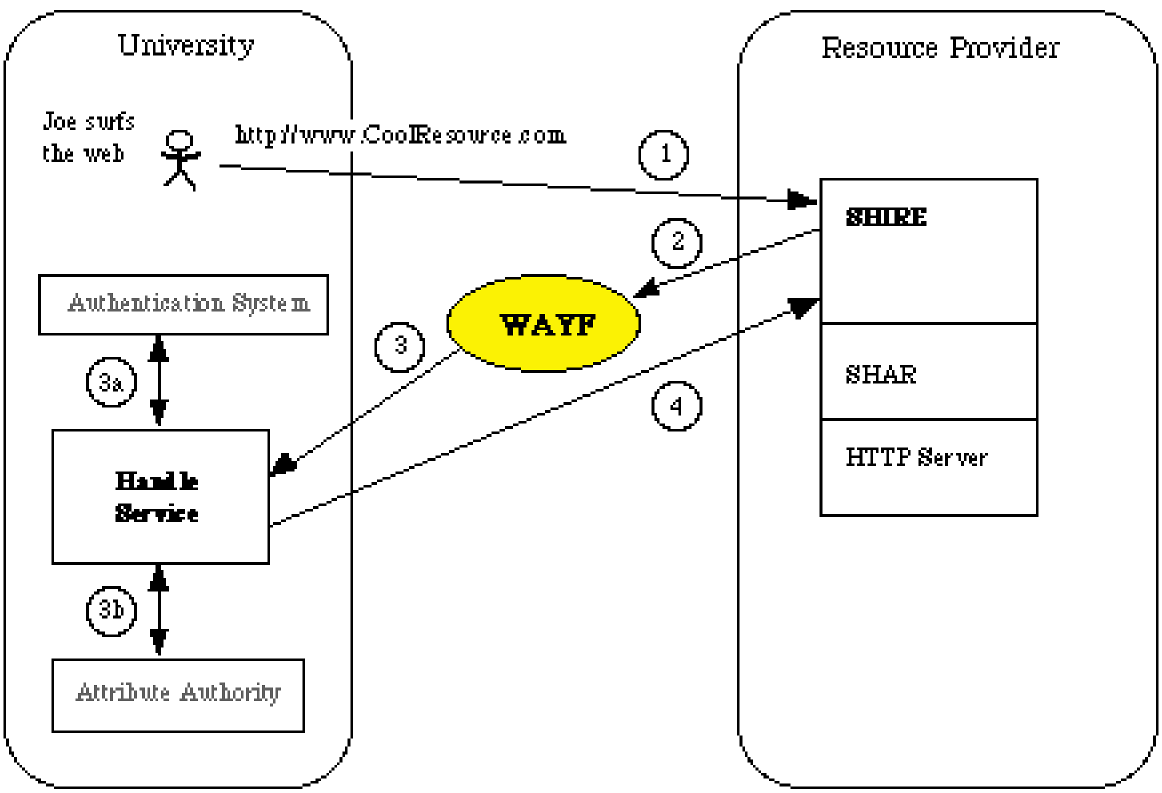

Figure 4: Handle Acquisition

1. The user "Joe" tries to get to the destination http://www.coolResource.com

2. The SHIRE at the destination redirects Joe to the WAYF. The SHIRE passes along its "handle acceptance URL" and the user's desired target URL as parameters. The WAYF interacts with Joe to find out his origin site. The WAYF then looks up the location and identity of the Handle Service for that origin site.

3. The WAYF redirects Joe to the HS, passing along the SHIRE's "handle acceptance URL" and the user's desired target URL as parameters. The HS makes sure Joe is authenticated and then …

4. … the HS sends back an opaque handle associated with Joe to the SHIRE's "handle acceptance URL" by means of a form posting. The user's desired target URL is passed as an additional parameter, and impersonation counter-measure information is presented as part of a "package" around the handle.

After the SHIRE performs the impersonation checks, and makes sure that the handle comes from a legitimate HS, the SHIRE passes the handle, AA contact information, and the origin site name to the SHAR.

The SHAR can then send an AQM (attribute query message) directly to the origin site's AA, and will receive an ARM (attribute response message), as was shown in Figure 3. Logically, these are steps 5 and 6 (not shown in Figure 4).

Once the SHAR has a set of attributes for the requesting user, it will pass them onto the manager of the resource the user requested, as was discussed in the previous subsection, "Slightly Simplified Architecture".

Complete details about the format of the messages exchanged and the process by which such messages are evaluated can be found in the Specification sections of this document.

Handle Service and Multiple AAs

Shibboleth separates the functions of handle creation and handle use. The HS creates the handle but only an AA uses it. Why this separation?

Shibboleth allows for the possibility of multiple AAs at a given institution. This is because some universities have distinctly administered user populations (for example, among separate "schools" within the university) even while sharing a common authentication mechanism across the university.

Permitting multiple AAs allows Shibboleth to be more easily adopted by institutions that have domains of separate administration.

Note that each AA must be fully able to answer an attribute query about a user within its domain. There is no means within Shibboleth for a destination to be told to contact multiple AAs to retrieve attributes about a single user.

The Handle Service at an institution with multiple AAs must be able to determine not only the user's identity, but which AA the user belongs to (so to speak). How this is done is up to the specific implementation.

A bit more about handles and the HS

Shibboleth doesn't specify how the HS creates a handle or how the HS knows the identity of the user. That said, we expect that most HS implementations will first make sure that the user is logged into the origin site's authentication system (whatever it may be). The HS might also communicate the user's authenticated identity to the appropriate AA and even ask the AA to create the handle. In any case, both the HS and the AA must jointly understand that a given handle belongs to a particular user.

One caveat is that the handle itself should be opaque. By opaque, we mean that no entity other than the AA (and the HS) can learn anything about the user from examining the handle alone. Thus, a username or the campus "id" of the user, while convenient, would not be a suitable handle. See the Shibboleth Deployment Guide for additional guidance on handle creation.

Isn't there too much complexity for the user?

The flows are relatively complex because we are trying to insure that the destination gets the right attributes about the particular browser user. This is much more difficult when there is no direct login by the user to the destination site. The important thing is that the user will be blissfully unaware of the complexity. In a fully-realized implementation, after the user speaks to the WAYF once (during a given browser session or for some defined period of time), the user shouldn't see any more Shibboleth-related pages. The flows between the destination, WAYF, and HS will still happen as the user surfs from destination site to destination site, but they will happen "under the covers".

3.2.2 Local Navigation or Resource Listing Sites

The previous section describes a user contacting a destination directly. An alternative form of contact is for the user to go through a "thin local portal" at the origin site. That is, the user would surf to a page that contains a series of destination links. In reality, each destination link the user sees would be invisibly paired with the "handle acceptance URL" with which it is associated.

When the user clicks on a link, a handle request to the HS is made automatically via a redirection. This request has exactly the same format as a handle request coming from the WAYF: the handle acceptance URL and the user's desired destination URL are sent as query string parameters.

The HS will authenticate the user if necessary, and then return a form containing the same sort of "handle package" that it would have, had the user accessed the destination first, as described earlier.

When the SHIRE receives the URL request with a handle package, it performs all of its impersonation checking and other validation work, and then passes on the handle and associated information to the SHAR. From the perspective of the SHIRE, there is nothing to distinguish an 'unsolicited' handle package from the result of a handle request it makes through the WAYF, because the SHIRE does not maintain state between handle request and response.

3.2.3 Client Certificates

A third scenario supported by the Shibboleth architecture is the use of a client certificate presented by the user to the destination site. (Note that this is distinct from the use of client certificates for local authentication; Shibboleth has nothing to say about this, since how a user locally authenticates to a HS is not specified by Shibboleth.)

This scenario is quite a bit different from the previous two. In particular, since the user is directly authenticating to the destination, there is no need for the SHIRE to establish a handle-based context for the user (since this only applies to users who don't directly authenticate themselves to a destination). Also, because of the potential variation in certificate formats we don't yet have a standard mechanism by which the SHAR can obtain the necessary information from the certificate to build and send an AQM.

At this point in time, origin sites that have certificate-bearing users will have to make agreements with destinations as to how information in the certificate should be used to form an AQM.

3.3 Attributes and Attribute Policies

Shibboleth's purpose is to securely transfer a user's attributes between the user's AA and the destinations the user wishes to interact with. But simply transferring attributes isn't enough: both origin and destination sites must agree on the attributes they will exchange and accept. Further, the origin site must allow users to choose which attributes get released and to whom, and the destination site must have a means of validating that the attributes it receives are legitimate. The next sections discuss these topics.

3.3.1 Attributes

Generally, attributes in Shibboleth are name/value pairs, though richer structures are possible.

The Shibboleth architecture doesn't specify a required set of attributes; however, any practical use of Shibboleth will require a reasonably small set of "core" attributes be understood by all or most cooperating sites. We expect that in addition to a core set of attributes, pairs (or small groups) of sites will agree to additional attributes that have specific relevance. For example, two universities may decide that "Role=Life Sciences Student" is an attribute/value that one university may send and that the other will receive and use in access control decisions.

Shibboleth's attribute syntax is described in more detail in Section 6.1.2.1. Throughout this specification (including the preceding paragraph), many attributes are denoted in examples in an informal, non-normative fashion designed to focus attention on the use of the attributes and not on the attributes themselves. In actual practice, SAML defines precise syntax for the metadata that serves to identify attributes used in Shibboleth.

3.3.2 Attribute Release Policies

Attribute release policies (ARP) are the rules that an AA follows when deciding whether or not to release an attribute and its value(s) to a requesting SHAR. A user may possess (and an institution may define) many different attributes; a SHAR asks generically for all attributes it is allowed to receive. An ARP defines the subset of attributes and their values that the user and/or the institution wish to reveal to a particular SHAR for a particular resource.

A basic ARP at an AA consists of at least the following:

- A destination SHAR name
- Optionally a URL tree
- A list of attributes (and possibly specific values) that should be released to this SHAR and URL tree

An ARP could also apply additional rules and guidelines based on just about any criteria an AA wishes to support, such as time of day, location, or many others. Each user may have as many attribute release policies as s/he needs. A fuller description of the different parts of the ARP follows:

The destination SHAR name allows the AA to find the right ARP when a SHAR makes a request for attributes. The AA, in addition to having the name, also has (or can obtain) the public key certificate of the SHAR. Since attribute request messages are always authenticated, the AA can ensure that only legitimate requesters get attributes. (There is one exception to the "always authenticated" rule: the AA allows for the "anonymous" requester. An anonymous requester doesn't and can't authenticate.)

The optional URL is for the case where a given SHAR may sit in front of very different sub-sites, or "application domains". An application domain is a set of resources that share the same resource manager. For example, recall that Professor Mary Smith was interested in both a multi-institution research site and a multiple sclerosis site. And recall that she wished to release different attributes to each. The research and multiple sclerosis sites constitute two different application domains. What if these application domains were both hosted at Ohio State behind the same SHAR? In that case, Mary's two ARPs would include the same SHAR, but different URLs.

The list of attributes that should be released to the designated SHAR is maintained by the AA (perhaps in conjunction with the organization's directory). Note though that the user can only release attributes and associated values that s/he actually has. Thus Sue, the IT staff person, can release the attribute "Affiliation=staff", but cannot release "Affiliation=faculty" if she is not a faculty member.

Finally, note that from the user’s perspective, a SHAR is an alien concept. Web browsers deal with hostnames and URLs, and most users are becoming at least somewhat comfortable with these notions. When a user asks that a given attribute be released (or not) to a particular target URL, the ARP that is being implicitly expressed is associated with a particular SHAR. The AA must be configured ahead of time to understand the association of URL targets and SHARs. A recommended default association is for the SHAR name to be deemed to match the hostname in the URL.

3.3.3 Attribute Acceptance Policies

An attribute acceptance policy (AAP) is the flip side of the ARP; it protects the SHAR rather than the user or the AA. When a SHAR receives a set of attributes from an AA, it performs basic checks such as authenticating the sender (making sure it is the expected AA), expiration, signatures, etc. SHARs should also validate that each attribute is something that the AA can legitimately assert. For example, if an AA at MIT claimed that the user had the attribute "field of study=PhD candidate at Harvard", then a SHAR might choose to ignore that attribute, and perhaps any other attributes that refer to institutions other than MIT. On the other hand, the SHAR might accept this attribute as a legitimate claim for a cross-registered student.

The destination site SHAR should know the domain, or scope, of attributes it is willing to accept from an AA. A natural default is to accept only attributes that refer to the AA's own institution. Shibboleth, however, doesn't specify AAPs since what is acceptable depends on private agreements between institutions. Shibboleth also doesn't currently specify a format for AAPs since we don't currently have enough experience with the type of policy expressions that are going to be of interest to destinations.

Strictly speaking, attribute acceptance policy might be a matter for components other than the SHAR. Attribute validation could be done by a component that the SHAR invokes or by a back end component like a resource manager that receives a set of attributes from the SHAR. Any time a component relies on an attribute, it has enforced an AAP, even if that policy is just "accept anything."

4 Relationship with SAML

Shibboleth has much in common with SAML, an emerging OASIS standard. SAML stands for "Security Assertion Markup Language". Shibboleth uses SAML formats and binding protocols whenever possible and appropriate. Of particular note: Shibboleth uses the SAML query and response protocol and formats for the AQM and ARM messages, and Shibboleth uses SAML's attribute statement and assertion format. As SAML evolves, Shibboleth will evolve along with it. Shibboleth, though, is both narrower and broader than SAML.

Shibboleth is narrower in that its use cases are far more limited than those of SAML: Shibboleth focuses on the browser user, while SAML also includes complex scenarios involving buyers, sellers, and brokers. SAML also specifies message exchanges and formats for authorization decisions, in addition to the attribute queries and assertions that are used by Shibboleth.

Shibboleth is also broader in that it travels outside the "edges" of the SAML model to include the notions of attribute release policies at the AA (a privacy issue) and attribute acceptance policies at the destination (a semantic trust issue).

5 Shibboleth Components in Detail

5.1 Review of the Shibboleth Flow

Before we detail each Shibboleth component, we briefly review the entire flow, from the user's initial contact of a destination through the SHAR receiving attributes about the user. The other use cases (portals, client certificates) are simpler subsets of this process. Figure 5 shows this flow. Here are the steps:

1) The user makes an initial request for a resource protected by a SHIRE.

2) The SHIRE obtains the URL of the user's HS, or redirects the user to a WAYF service for this purpose.

3) The SHIRE or WAYF asks the HS to create a handle for this user, redirecting the request through the user's browser.

4) The HS returns an opaque handle for the user that can be used by the SHAR to get attributes from the appropriate AA at the origin site. The arrow shows the SHIRE, after performing impersonation checks, passing on the handle (and AA information, and organization name) to the SHAR.

5) The SHAR asks the AA for attributes via an AQM message.

6) It receives attributes back from the AA via an ARM message.

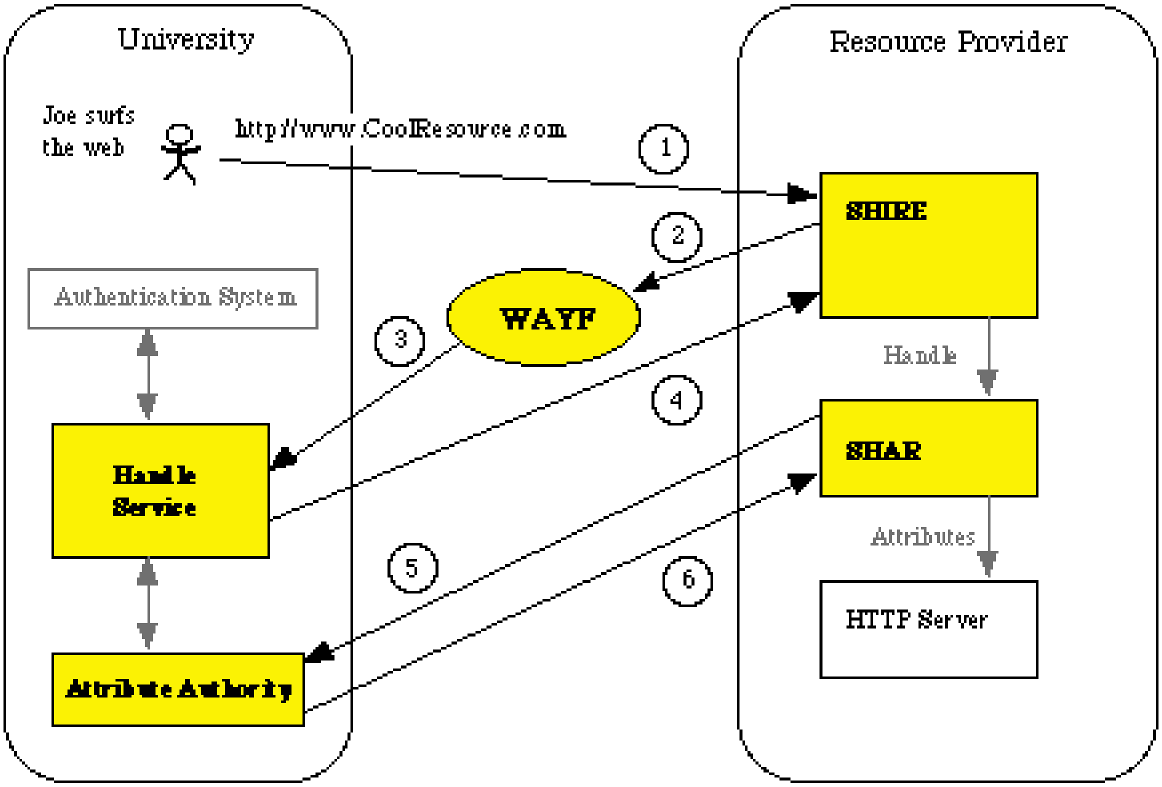

Figure 5: The Shibboleth Flow for "First Contact" case

The following sections discuss the components in the order they appear in the flow: SHIRE, WAYF, Handle Service, SHAR, and AA.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this section are to be interpreted as described in **[RFC 2119]**.

5.2 The SHIRE

The SHIRE is the component responsible for intercepting an HTTP request for a protected resource or service and associating it with a handle suitable for attribute requests by the SHAR. The handle is therefore known as an Attribute Query Handle (AQH). The SHIRE also provides the SHAR with the domain name of the origin site of the user making the request, along with the location and binding information needed to contact the appropriate AA for attributes.

If, instead of utilizing a SHIRE, the web server requests a client certificate, the browser may present a compatible one while establishing an SSL connection. The certificate, if acceptable to the server, could itself be used as a handle by the SHAR. How to identify the origin site or the AA are not currently specified by this document. Use of client certificates by destination sites is therefore possible, but not fully specified.

In the event that a certificate is not presented, the SHIRE must obtain a handle for the browser user from the user's HS. The SHIRE also performs a number of checks to reduce the likelihood of impersonation, and should take actions to maintain state with the user (to obviate the need to obtain another AQH if the user again visits the destination during the same browser session). State management is discussed in greater detail in section 5.2.6.

5.2.1 Obtaining an Attribute Query Handle

If an AQH can be associated with an incoming request via some kind of state or session with the browser, then processing continues with section 5.2.5. If the SHIRE is not able to recognize the browser user from a previous request, a new AQH needs to be obtained.

The first step in obtaining an AQH is to discover the user's origin site. The next step is the actual request to the appropriate HS for an AQH. Finally, an AQH is "presented" to the SHIRE in response to the request.

5.2.1.1 Determining the User's Origin Site and Handle Service

To discover the location of the user's HS, the SHIRE can employ the service of a distributed WAYF service, if one exists, or it can ask the user directly. In the former case, the SHIRE must be configured with the URL of the WAYF, or must otherwise be able to look it up. This is an implementation decision. The request to the WAYF is a URL directed to it with the following appended query string parameters:

- target
  - The target, or destination, URL originally requested by the user
- shire
  - The URL at which the destination SHIRE receives the AQH presentation

The SHIRE redirects the request to the WAYF through the user's browser. The parameters are URL encoded per **[RFC 2396]**. Refer to section 5.3 for more details regarding the WAYF's interaction with the user and other properties. The WAYF will, after determining the correct origin site HS, direct the browser to the HS, effectively acting as a proxy for the SHIRE in requesting a handle. The redirection format is the same as the redirection from the SHIRE to the WAYF.

If a SHIRE implements WAYF functionality itself, it obviously must have (or be able to otherwise obtain) the locations of the HS's for institutions acting as origin sites for its user population. The SHIRE can interact with the user as it chooses to determine the user's origin institution. Section 5.3 will describe the kinds of interactions desired.

The "shire" parameter is an absolute URL at which the SHIRE has been configured to receive AQH information per the profile described in section 5.2.1.3. Maximum URL length in clients and servers is limited in practical, if not specific, terms. Therefore, a SHIRE's handle acceptance URL SHOULD be as short as possible to leave maximum space for the "target" parameter. Various means of URL rewriting and virtualization can be used to minimize its length. The browser submission of the handle presentation SHOULD be protected with SSL/TLS, but there may be valid reasons for using HTTP alone.

5.2.1.2 Attribute Query Handle Request

The attribute query handle request from the SHIRE (or WAYF) to the HS is identical to the request from the SHIRE to the WAYF described above in section 5.2.1.1.

5.2.1.3 Attribute Query Handle Presentation

An attribute query handle is "presented" to the SHIRE at a specific URL as a package containing the handle along with AA location and impersonation countermeasure information. The package MUST be submitted in the body of an HTTP POST in accordance with the SAML Browser/POST profile described in **[SAMLBind]**§ 4.1.2 and section 5.4.4. The package consists of two REQUIRED form elements, TARGET and SAMLResponse.

TARGET is the URL to which the user's browser will be redirected after the handle has been accepted, and may derive from the SHIRE's request to the HS or be set by the origin site, as in the case of a portal of some sort.

SAMLResponse contains status information, the handle, AA location information and information for impersonation countermeasures. The SAMLResponse form element contains a SAML protocol response wrapped around an authentication assertion. Formally, it is a base64-encoded XML instance document conforming to the SAML schema for a Response containing an Assertion containing an AuthenticationStatement element. In accordance with the SAML profile, the response MAY contain additional assertions and statements; such usage goes beyond the bounds of this specification.

Section 6.4 describes required schema and element usage for the SAMLResponse XML instance; the following sections reference many of these elements and attributes.

5.2.2 Handle Validation

When the SHIRE receives an HTTP or HTTPS request at its designated acceptance URL, it first examines the incoming submission to insure it is in accordance with the profile described in section 5.2.1.3 (i.e. a POST containing the standard form encoding and the TARGET and SAMLResponse form elements).

Before accepting the handle, the SHIRE must use other information inside the response and the assertion to validate the handle and make impersonation of a user as difficult as possible given the constraints of commercial web browsers. Without such protections, the browser user may be associated with the wrong originating identity, incorrect attributes may be retrieved, and ultimately an incorrect access control decision may be made.

If the Response element does not contain an Assertion element containing an AuthenticationStatement, then the mandatory SAML Status element SHOULD be used to generate an appropriate error message to the browser. The SHIRE is not required to explicitly display messages received in StatusMessage elements, but this may be advisable.

The following steps MUST be followed, in any order. Any failure MUST result in an error being returned to the user and could be logged in some fashion. See section 5.2.4 for suggestions on error handling behavior.

- The MajorVersion and MinorVersion Response attributes MUST indicate a compatible SAML version (see **[SAML]**§ 4 for SAML version compliance requirements).
- Compare the IssueInstant of the response against the current time. The SHIRE SHOULD NOT accept a response older or post-dated more than approximately 5 minutes, but the exact allowance for transmit time and clock skew is an implementation decision.
- Compare the ResponseID against a replay cache of known responses. It MUST be unique and not previously seen. The allowable clock skew bounds the size of the cache.
- Find a Recipient attribute value in the Response equal to the SHIRE's acceptance URL.
- Validate the signature over the response using the signer's key. Obtaining and validating this key are outside the scope of this document.
- Locate an AuthenticationStatement in an Assertion containing the “bearer” Subject ConfirmationMethod, and evaluate the validity of that assertion and statement, including Version, Conditions, and the signature, if one is provided.
- In any and all other respects, the response and assertion MUST be considered valid in accordance with **[SAML]** and **[SAMLBind]**§ 4.1.2.

Additionally, if the assertion statement contains a SubjectLocality element with an IPAddress attribute, the SHIRE MAY match its value against the address of the client providing the response.

Having followed these steps, the SHIRE can protect itself against various attacks. The timestamp and address checking reduce the likelihood of an attacker successfully using a stolen assertion. Embedding the destination SHIRE in the signed response prevents a malicious SHIRE from passing an assertion to a second SHIRE thus masquerading as the user. Finally, signing the response prevents arbitrary construction of an acceptable credential, provided the signing key can be validated independently in some fashion. The following section, 5.2.3, discusses some aspects of this problem.

Note that it is permissible for the assertion to be signed independent of the response, but this is of questionable use because **[SAMLBind]**§ 4.1.2 specifies that the assertion containing the statement with the “bearer” confirmation method should be short lived. If semantics that go beyond those described here require a long lived, transportable statement of authentication (or other) information, another assertion can be sent alongside the mandatory content of the response.

5.2.3 Handles, Trust, and Organizational Identity

Before accepting a handle, the SHIRE must validate the signature provided by the HS in order to protect itself from forgery and authenticate the handle's source. The key needed to verify the signature may or may not be included with the response. If it is not, the name of the issuing HS (from the assertion’s Issuer attribute) could be used to locate a corresponding key.

Once the signature has been validated, the SHIRE can be assured that the owner of the corresponding key created the response. Questions remain, however, as to whether the owner of that key is authorized to issue handles, and for what organization(s) it is allowed to issue them. These questions of legitimacy are critical, and from an architectural perspective, the SHIRE as the relying party is free to conduct whatever checking it requires in answering them. However, it specifically must not make assumptions about the name of the organization from the name of the HS. This association must be established by some other means; the strength of those means is purely a matter of policy.

However the SHIRE determines the identity and legitimacy of the issuing HS, the ultimate results are a validated handle and the issuing organization's domain name. The name is not of direct significance to the SHIRE, but may well be important for the SHAR to use in evaluating the attributes it receives (see section 3.3.3).

Summarizing, the SHIRE must obtain a key, use it to verify the response signature, and associate the valid handle internally with the domain name of the organization that "owns" the HS. These processes are left unspecified, but there are some straightforward certificate-based mechanisms that provide at least partial solutions to these problems, such as using predefined lists of valid certificates or verification of the issuing CA using a set of trusted roots. The details depend on implementation choices, policies, and out-of-band agreements necessary anyway for inter-organizational sharing of security attributes to have meaning or value.

5.2.4 Error Handling

Due to the nature of the interactions, most error handling on the part of the SHIRE occurs while processing a new, incoming handle. Other predictable errors involve WAYF functionality that the SHIRE may be implementing, and are discussed within section 5.3. Since the SHIRE is interacting directly with the browser, there is little error formalism specified. Most other problems would arise after redirection of the browser away from the SHIRE, such as the case in which a HS is unreachable for some reason. Note also that the HS MAY provide to the SHIRE a response that contains error information in the form of SAML status content, and the SHIRE MAY reflect all or part of this descriptive information to the client.

Common errors and some possible suggested actions are discussed below. None of this should be considered normative.

- Unrecognized or unacceptable information in any of the input (e.g. missing, ill-formed or invalid XML, bad version, badly formatted domain names, etc.)
    - The SHIRE might return a page indicating that a HS provided malformed information and refer the user to the origin site's technical support.
- A mismatch between the SHIRE and the Recipient attribute, an expired or replayed assertion or response, or an invalid signature
    - The SHIRE might return a page indicating that the request was rejected for security reasons, and ask the user to try the interaction again, or refer the user to the origin site's technical support.
- A mismatch between the embedded address and the requesting client's address

- o The SHIRE might return a page indicating the request was rejected for security reasons, but might also suggest that the cause could be a firewall or proxy used in the request.

Any or all of these errors could indicate (but do not prove) an attempt to impersonate the user and could be logged as such for audit purposes. Presentation to the user should take a more relaxed stance, with a technical description of the problem accompanying a more palatable message. Whenever possible, solutions or explanations should be suggested. For example, if an address mismatch occurs, it's far more likely that the user is behind an address translating device than that s/he is trying to attack the system; a suggestion to contact a local system administrator who may understand the problem would be appropriate.

Another set of errors outside the scope of this document may occur as the SHIRE attempts (perhaps unsuccessfully) to locate and validate a key and organization name for the HS. A SHIRE might find certificates have expired or have been revoked, or it may receive a response from a previously unknown HS. The previous guidance regarding error handling is likely to apply equally well to these cases. In some cases, it may be difficult to ascertain if the failing is in the origin site or a lack of necessary configuration at the destination.

5.2.5 Handle Extraction and SHIRE/SHAR Interaction

Shibboleth doesn't specify the interaction between the SHIRE and the SHAR components. In many, perhaps most, cases, the SHIRE and SHAR will be elements of a common implementation module within an HTTP server, allowing in-memory exchanges to take place in an obvious way.

What is specified, however, is the data to be communicated. The SHIRE must provide to the SHAR (or be certain the SHAR can obtain by other means) the following pieces of information for each request:

- the HTTP request URL, method, headers, etc. (i.e. the desired resource and the set of information which would naturally be expected by the endpoint of an HTTP request)
- an Attribute Query Handle (or an X.509 certificate serving as such)
- all AuthorityBinding information sent by the HS
- the domain name of the organization that issued the handle or certificate

5.2.6 State Maintenance

Shibboleth doesn't explicitly define a way of managing state or sessions for users, but the SHIRE MUST provide some means of subsequently identifying the browser user as "owning" the handle it acquired. Acquiring a new handle for each request (even if repeatedly getting the same handle back) is technically correct, but likely not acceptable because of performance. Session state will include the user's handle and all of the information that was associated with it (i.e. origin site and AA location information). A typical session design might rely on locally scoped, non-persistent HTTP cookies, although other approaches have been seen "in the wild."

The handle itself is not secret, nor is it likely to be impractically large to store in a cookie, but there are various reasons why using the handle itself as a session identifier is a poor decision. Foremost among these is that the namespace of handles is not guaranteed to be unique, although it may be unique for a given HS. Additionally, since handles are not secret, it may be relatively easy to manufacture one that a SHIRE would accept as a valid session identifier, leaving the optional checking of IP address as the only protection against impersonation.

A much better solution would involve a more uniquely-generated token that relies on cryptography applied to the session data involved. For example, the handle, a timestamp, origin site and a random value maintained by the SHIRE could be used as input to a message digest function or encrypted with a secret key, producing a compact session identifier that is hard to forge but easy to verify. Such an approach would require the SHIRE to maintain an internal state table, rather than recovering all of the client's contextual data from the cookie.

Note, however, that once the token is associated with its state and issued to the client, the only practical protection possible remains IP address checking (and even that may be impossible for reasons discussed earlier). The token can be made hard to forge, and the use of SSL may make it hard to steal, but the token may ultimately still be vulnerable at the client, whose security is dependent on many software and human factors.

5.3 The WAYF

The WAYF component is so named because its purpose is to ask Shibboleth users "Where are you from?" in order to direct them to their origin site's HS. Its functionality can be implemented in a stand alone fashion, or it can be implemented by the SHIRE directly. Of the three interaction scenarios described in Section 3.2, only Direct Access to Destination Site (section 3.2.1) requires the functionality of a WAYF service. With a local navigation site, the user's handle package is part of the initial request to the destination site, and when using client certificates, both the certificate and an associated digital signature are presented by the client.

5.3.1 Requesting a WAYF Lookup

If a SHIRE wishes to use a stand-alone WAYF service, it must be preconfigured with the URL of that service. The WAYF implementation SHOULD use SSL for all interactions (see Section 5.3.3 for more on security). The format of a WAYF lookup request is described in Section 5.2.1.1. The two parameters provided will ultimately be passed to the HS as a handle request, and must therefore be preserved across whatever interactions take place between the WAYF and the browser. Hidden form fields could be used for this purpose.

5.3.2 Determining and Transferring to Origin Site

Only the browser user is accurately able to determine the organization which can locally authenticate him or her and issue a Shibboleth handle. The actual URL of each HS is known to the WAYF by association with a set of one or more human-readable, possibly colloquial, names corresponding to the origin site. For example, "The Ohio State University" might also be known to the WAYF as "OSU", "Ohio State", or even "Buckeyes".

The WAYF's job, upon receiving a request in the proper format, is to present a form to the user prompting him/her to enter an origin site name. The WAYF will search its database and present the most likely candidates and their synonyms. It should allow for misspellings and variant names to the greatest extent practical. The search results should be presented as a set of links or selections. Each link or selection should send the browser to the associated handle service along with the parameters originally passed to the WAYF from the SHIRE. Again, section 5.2.1.1 describes the exact format which must be used.

The WAYF MAY issue a cookie of some sort to cache the identification of the user's origin site. This can be used to bypass the user prompting that would ordinarily take place, and transfer the browser directly to the origin site. For such a design to work, the WAYF may need to alter the presentation of the search results to direct the links back to itself so that a cookie may be issued identifying the selection while redirecting the browser to the HS. Other designs using JavaScript may also be possible; this is left entirely up to the WAYF implementation. Note that if such caching is employed, it SHOULD NOT be perpetual and some mechanism MUST be provided to override it.

5.3.3 WAYF Security Implications

While the WAYF does not play a formal role in the security of the Shibboleth architecture, it is a prime target for implementing social attacks that would steal passwords through misrepresentation. In directing the user to their HS, the WAYF is implicitly saying "Here's a site that may ask you to locally identify yourself, perhaps with a password." Were the WAYF compromised, it could ask for such information directly, or redirect the browser to a malicious service (possibly itself) that might do so.

A typical unobservant user may well give away their password without noticing anything until the original destination is unable to process their handle (assuming the attacker even bothers with creating one; s/he might well display an official-looking error page to further mislead the user into complete confusion). This kind of semantic attack is very difficult to defend against in most cases, because users are not predisposed to verify in detail the sites that they visit. SSL, in particular, has engendered a kind of implicit trust among some users that if they see a locked browser icon in their status bar, the site they are accessing and their interactions with it are "secure".

Being certain that the page asking one to login is the "right" one is a very complex process, including steps such as verifying the destination of the form submission, the name of the site, the contents of its certificate, and ultimately a comparison of much of this to known information that most users may not possess (and some of which may change for entirely innocuous reasons anyway). The WAYF plays a role in this process by securing itself against attack and taking its responsibility seriously, but it is also the job of each origin site to take on the responsibility of educating users and circulating the information necessary to help them protect themselves.

5.4 The Handle Service

The Handle Service is the origin site component responsible for (indirectly) providing the SHAR with a handle to be used for making attribute requests to an origin site AA. In concert with the user's browser and the SHIRE, the HS establishes a secure context for communication about the user that will later occur between the SHAR and AA, It does this without revealing the user's name.

Functionally, the HS is a web-based service that waits for Attribute Query Handle requests and responds with a properly constructed AQH Presentation, as described in Section 5.2.1.3. Such requests may be directly from a SHIRE, from an intermediate WAYF service, or via a web page hosted by the origin site designed to transfer users to specific destination sites. As long as the parameters needed are present, the initiator is irrelevant. Once a request takes place, the specific steps involved in responding are enumerated here.

5.4.1 Origin Site Authentication and Single Sign-On

The HS receives the handle request as a set of parameters from the user's browser; it must at this point determine who the user is. Shibboleth doesn't specify how this is accomplished; the choice of authentication method is left to the origin site. Logically, the HS can be thought of as a local application that requires authentication for it to operate properly. This can be accomplished by whatever means are locally acceptable, be they name and password, client certificate, a customized single sign-on interaction with cookies, or something else.

The HS may or may not be a component of an origin site's implementation of single sign-on across servers and applications. It may perform authentication itself or it may delegate this function and perform its task following that authentication. It may remember the user's identity in order to implement single sign-on across Shibboleth destination sites, or it may choose to require the user to authenticate each time a new destination requests a handle. These decisions are left to implementers.

5.4.2 Determining the Correct Attribute Authority

Due to internal organizational issues or political boundaries, some origin sites may find it impractical to construct a single view of the relevant attributes of all its potential users. Even without internal organization issues, it may still be desirable to partition the user population across multiple AA responders for performance or reliability.

If an origin site wishes to have multiple AA's, the HS must determine which one is an appropriate point of contact for the SHAR for each user. Shibboleth doesn't specify how this is accomplished, as it is a very local implementation decision, based on the reasons for having multiple AAs, and the way in which users may be divided across them. An origin site may establish a set of equivalent AAs and pass information about each of them to a SHIRE. When a destination site is provided with multiple AAs, it is assumed by the SHAR that any of the AAs will respond identically when asked for attributes.

Note that if an organization only needs a single AA (which may be quite common), the HS itself bears a one to one correspondence with it, and the two components could be implemented together with no loss of flexibility.

5.4.3 Handle Creation

The HS must construct an opaque handle for the browser user which is acceptable to the AA that will provide attributes about that user. In general, this means that the AA must be able to determine the user's identity (or principal name) from the handle. There are many ways to do this (aside from the obvious and privacy-compromising degenerate case of using the principal name as the handle). For example, the HS could interact with the AA via some unspecified means, passing it the user's identity, and allow the AA to create the handle. Alternatively, the HS could encrypt the user's principal name and a salt value with the public key of the AA, in which case no real-time interaction with the AA is needed.

The handle string itself may contain additional information for use by the AA, such as a validity period. This is implementation dependent, and opaque to the destination site. Note that such additional information ought not to be in plaintext as the handle would then be subject to modification. To defeat this, a HS or AA might sign and/or encrypt the handle, but this is all implementation dependent.

Though Shibboleth doesn't specify the format or content of the handle, the handle SHOULD be opaque. No observer should be able to figure out the identity of the user simply by examining the handle. Note however that the handle is not "secret" in Shibboleth. It is passed from HS to SHIRE, from SHIRE to SHAR, and may be passed on to additional entities. A handle that revealed the user's identity would therefore defeat the privacy controls in Shibboleth.

Architecturally, Shibboleth stops short of mandating opacity. If an organization chooses to forgo privacy, it can still interoperate if it wishes. Clearly, such an organization should emphasize the implications of this to its user population.

5.4.4 AQH Presentation

Once the HS has issued (or obtained) a handle for the browser user, the handle is packaged in a signed SAML response as described in section 6.4. The HS MUST digitally sign the response using its key. The verification key is assumed to be obtainable through unspecified means (e.g. in a certificate passed along with the assertion); also unspecified is how the association between that key and the HS is to be validated by the SHIRE (see section 5.2.3 for details on this topic).

The HS MUST return to the browser a standard HTML document containing a form element. The form MUST contain at least two parameters in "hidden" input elements, "TARGET" and "SAMLResponse".

The TARGET parameter is the URL to which the browser will be redirected after the SHIRE processes the information. The HS is provided with an input parameter named "target" whose value is a URL; the HS MUST set the TARGET element value to an equivalent URL value.

The SAMLResponse element value MUST contain the signed SAML response in Base64 format.

The form's "enctype" attribute value MUST be "application/x-www-form-urlencoded". Its "action" atttribute value MUST be a URL at which the destination SHIRE can process the form submission. The HS MUST set this attribute to the value of the input parameter named "shire". The form's "method" attribute value MUST be "POST".

The HS MUST include in the form at least one input element of type "submit". The HS SHOULD include in the response appropriate text explaining the purpose of the form and the expected outcome of its submission. Further, it MAY include in the response sufficient client-side scripting to cause the form to be submitted automatically without intervention by the user, but the form MUST be usable if the user's browser does not support or has disabled this feature.

5.4.4.1 Partial Example

A partial example of an XHTML response that fulfills the profile requirements follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
     lang="en">
  <head>
    <title>You have been Shibbolized!</title>
  </head>
  <body onload="shib.submit()" style="text-align:center">
    <p>Ready to transmit your Shibboleth handle...</p>
    <form name="shib" action="https://frobozzica.com/shire"
        method="POST">
      <input type="hidden" name="TARGET"
          value="https://frobizzica.com/restricted/"/>
      <input type="hidden" name="SAMLResponse"
          value=" bHMgZGlyIC9oIC93ICUmDQ...9wIlwNCg=="/>
      <input type="submit" value="Transmit"/>
    </form>
  </body>
</html>
```

5.5 The SHAR

SHAR is an acronym for "Shibboleth Attribute Requester". The SHAR acquires attributes about a browser user from the user's AA. The SHAR, once it has these attributes, will send them on to the manager of the resource the user is trying to access. The resource manager (RM) will then make an access control decision based on the user's attributes, and either grant or deny the user's request. If the user is simply trying to access a static web page or a typical web application, this RM may be the web server itself. In the case where the user is attempting a more complex action (say updating experimental results or transferring grant money), the RM may sit "behind" the web server on a separate machine.

Many distinct RMs (and associated resources) may "sit" behind the same SHAR.   For example, a set of resources about the disease AIDS, and resources about actor John Lithgow may be located behind the SHAR at say, Harvard University (Lithgow's alma mater). Since the user may have very different attribute release policies for the AIDS RM vs. the John Lithgow RM, the SHAR is responsible for knowing when the user "switches" from accessing one RM to using another, and for then acquiring the attributes the user wishes to release to the current RM (rather than using potentially inappropriate cached attributes.)

The rest of this section discusses first attribute acquisition and error handling for attribute acquisition, and then the question of multiple RMs and attribute caching, and finally how SHARs interact with RMs.

5.5.1 Attribute Query and Response Exchange

5.5.1.1 General Considerations

The SHAR uses the information it receives from the SHIRE to issue an attribute request to an AA, and to evaluate the "appropriateness" of the attributes it gets back. Section 5.2.5 describes this information in detail, but summarizing, the location(s) of the AA and the handle (or X.509 certificate) are used to construct the attribute request, and the origin site name is used in evaluating the returned attributes. (For example, a SHAR might decide that an MIT AA is not allowed to assert attributes related to Harvard University.)

The SHAR MUST consider each AA it is given to be equivalent. It MAY use one or more of them in sequence, in any order, until it receives a valid response. A response that leads to a denial of access for the user SHALL NOT be considered invalid. (In other words, a SHAR does not contact a second AA simply because the first returns attributes that do not result in access being granted.)

5.5.1.2 Attribute Query Message

The message sent by the SHAR to the AA is called an Attribute Query Message (AQM) and MUST contain a single XML element that conforms to the SAML schema for a Request containing an AttributeQuery. The format of this message is specified in **[SAML]** and section 6.1.1. Summarizing, it consists of the following:

- A message version and request identifier
- The attribute query handle or X.509 certificate of the user
- Optionally, the target URL of the resource the user requested (The SHAR SHOULD send this as a parameter when asking for attributes about an online user. Since SHARs may theoretically also issue requests about users who are not online, the URL is optional.)

5.5.1.3 Attribute Response Message

In response to the AQM, the AA is expected to return an Attribute Response Message (ARM). The ARM must contain a single XML element conforming to the SAML schema for a Response, containing an Assertion, itself containing an AttributeStatement. The message format is specified in **[SAML]** and section 6.1.2. The ARM consists of:

- A message version, request identifier, and response identifier

- The subject of the assertion (in Shibboleth terms, the handle or certificate)
- Validity constraints (e.g. time and usage policies)
- Zero or more attributes about the user

In accordance with the standard SAML protocol, the response MAY contain additional assertions and statements; such usage goes beyond the bounds of this specification.

5.5.1.4 Message Exchange Requirements

An AQM can be sent to an AA via any exchange protocol shared between the SHAR and the AA. Any such protocol MUST support mutual authentication, message integrity, and response confidentiality. These features may be provided at the underlying transport level or with protections attached to the query and response messages. The protocol SHOULD be synchronous and MUST provide request/response semantics. A request/response protocol and SOAP/HTTP binding specified by SAML that all SHAR and AA implementations MUST support is described in **[SAMLBind]**§ 3.1 and section 6.3.

5.5.2 Error Handling On the AQM/ARM

There are lower-level, operational semantics that always apply when processing an ARM. Many will depend on the specific binding protocol in use, and are specified as part of such a binding. Provided no such errors are detected, the Response, Assertion, and AttributeStatement information in the ARM is evaluated as follows:

- The InResponseTo attribute on the Response element MUST match the RequestID attribute in the Request element sent by the SHAR.
- The MajorVersion and MinorVersion Response attributes MUST indicate a compatible SAML version (see **[SAML]**§ 4 for SAML version compliance requirements).
- If a Conditions element containing NotBefore or NotOnOrAfter attributes is present in Assertion, then the assertion MUST be valid at the current time. Post-dated or expired assertions are considered an error.
- The Subject of the Assertion MUST match the Subject of the AttributeQuery sent by the SHAR.
- In any and all other respects, the response and assertion MUST be considered valid in accordance with **[SAML]**.

Should evaluation of the response fail, an error condition results at the SHAR. The SHAR MUST communicate the problem back to the user in some fashion. This may involve passing error detail or some other interaction; the specifics depend on internal implementation details not specified in this document.

In addition to "request-side" errors, the AA may communicate error conditions to the SHAR in the ARM. The possible errors and how they are communicated and detected depends on the binding protocol in use. In general, the SHAR SHOULD use the information provided by the AA and/or the binding layer to describe the problem to the user. It may respond however it sees fit based on the kind of error it detects.

Any error on either end in requesting attributes MUST result in a failure to fulfill the user's request. Note that it is specifically not an error for an AA to return zero attributes in an ARM, although this may result in a subsequent access failure.

5.5.2.1 Real-Time Attribute Release

One particular kind of "error" condition exists to support the potential real-time interaction of users in controlling the release of their attributes to a SHAR. A SHAR MAY check for this error condition indicating that the user's ARP at the AA requests a real-time decision.

To communicate this condition to a SHAR, an AA MUST return a SAML Response containing a Status consisting of a top-level StatusCode of "samlp:Responder" (see **[SAML]**§ 3.4.3.1) and a second-level StatusCode of "shib:RealTimeRelease", where "shib" denotes the Shibboleth XML namespace of "urn:mace:shibboleth:1.0". In addition, the StatusDetail element MUST enclose a "shib:RealTimeReleaseURL" element containing the URL of a web interface via which the user can interact with their AA to control attribute release. The SHAR MUST directly or indirectly redirect the user's browser to this URL, with the current request URL appended as a parameter called "target".

Once the user has indicated his/her decision to the origin site, the browser MUST be redirected back to the URL contained in the "target" parameter, and processing resumes at the destination site. See section 5.6.X for further information about the nature of this interaction with the AA.

5.5.3 Multiple RMs and Attribute Caching

Recall that the SHAR may provide attributes to very distinct RMs, for example an AIDS subsite, and a John Lithgow subsite. A given user may (or may not) have different attribute release policies for each subsite. The SHAR doesn't know the user's specific ARPs but does know when the user browses between "application domains". An application domain is a set of resource URLs that are controlled by a single RM.

The SHAR is responsible for communicating the proper attributes to each RM. One way to do this is to ask for attributes every time a user requests a URL. This is technically correct, but very inefficient. SHARs will generally cache attributes for the user. The SHAR (if it caches) MUST cache attributes for a user segmented by application domain.

If the user request for a URL is in an application domain for which the SHAR does not have cached attributes that remain valid, the SHAR must send a new AQM (with the user's requested URL) to the AA. If it did not, and rather used cached attributes, there are two unpleasant possible outcomes: The user could be denied access to the new resource if its RM required different attributes than those that were cached; Alternatively, the cached attributes may reveal information to the resource's RM that the user didn't which to release.

Note that application domains are generally URL subtrees. Though Shibboleth doesn't specify how the SHAR knows about the distinct application domains that it provides access to, it may simply keep a configured list of the URL subtrees for each application domain.

5.5.3.1 Attribute Caching and Application Domain Example

Consider an AQM containing the following information:

- the opaque handle " 004a4f71-35f2-1b79-87c4-00b0d0a7039c", which actually refers to Professor Mary Smith
- the target URL "http://www.jhu.edu/research/diseases/MultipleSclerosis"
- the SHAR name "www.jhu.edu"

Mary Smith has two ARPs that reference the SHAR "www.jhu.edu":

SHAR:	www.jhu.edu
URL:	http://www.jhu.edu/research/diseases
Release:	Affiliation=MemberOfCommunity

SHAR:	www.jhu.edu
URL:	http://www.jhu.edu/research/diseases/MultipleSclerosis
Release:	Username, Affiliation=MemberOfCommunity

The AA will determine that the second ARP is the best "match" for the request (see section 5.6.1.2 for specific information on this process), and return Username and Affiliation with the value MemberOfCommunity (assuming the user does, in fact, have those attributes), and the SHAR will cache those attributes for the application containing that URL. Should a later request from the user arrive for "http://www.jhu.edu/research/diseases/ALS", the SHAR must determine whether it is part of the same application domain as the original URL.

If it is, then the SHAR can reuse the pair of attributes. If it isn't, then the SHAR MUST re-query the AA. Note that a "mistake" can occur if the user believes that two URLs at the same site constitute two distinct application domains and the SHAR believes there is only one application domain. There are two possible types of mistakes:

- User Confused: If the user thinks that there are separate application domains when there are not, the user is potentially revealing more information than s/he'd like.

- SHAR Misconfigured: If there are two separate application domains, and the SHAR believes there is only one, the SHAR will re-use information when it should not. This is an outright error. The SHAR must have correct knowledge of the application domains it serves.

Further analyzing this example, the SHAR needs to be aware of the application domains behind it, and must be able to efficiently map the request URL to its corresponding application domain in order to identify the proper cache of attributes to examine. In this case, the “www.jhu.edu” SHAR might divide its space into (at least) three application domains:

- http://www.jhu.edu/research/diseases/*
- http://www.jhu.edu/research/diseases/MultipleSclerosis/*
- http://www.jhu.edu/research/diseases/ALS/*

A consistent view of this separation of URL trees at the SHAR and AA allows caching to take place without violating the intent of the user’s ARPs.

5.5.3.2 Validity of Cached Attributes

When the SHAR queries the AA for an assertion, the assertion may contain a NotOnOrAfter attribute in a Conditions element. If so, the SHAR must keep this information when it caches the attributes, and it must check on whether or not the attributes are still within the validity period before using them. If the attributes have expired, the SHAR MUST re-query the AA.

5.5.4 SHAR's Interaction with Resource Managers

Because attributes are treated generically in SAML and Shibboleth, a detailed specification for communication of attributes between the SHAR and RM is not possible. Shibboleth also allows the SHAR and RM to be implemented separately, so only a very general mechanism using existing HTTP semantics is universally applicable. Because the size of the information and the cost of transmitting it downstream may be significant, there is no required mechanism that the SHAR must use to communicate attributes to the RM. This allows the SHAR and RM to optimize their interaction in special cases in which they share a process space or have specific knowledge of each other.

For uniformity, the SHAR MAY annotate the user's HTTP request with the attributes appropriate for that user and resource, in a single HTTP header named "Shib-Attributes", as described in section 6.5. The SHAR MAY add additional unspecified headers if it wishes (for example, a distinct header per attribute).

It is up to the RM to grant or deny the user access to the requested resource. Standard HTTP “403 Forbidden” responses MAY be used, but are not required.

5.6 The Attribute Authority

The Attribute Authority is an origin site component responsible for the following tasks:

- responding to attribute queries from SHARs
- providing the means for users and administrators to specify ARPs
- acquiring and maintaining information about SHAR/target associations for the purposes of managing ARPs
- enforcing the privacy precautions inherent in the ARPs

The following sections will discuss these responsibilities. There may be other functionality provided, depending on the specifics of an implementation.

5.6.1 Responding to an AQM

Most of the AA's time is generally spent waiting for, and responding to, AQMs sent by various SHARs. Section 5.5.1.2 summarizes the contents of these messages, and section 6.1.1 describes the message format in detail. The protocols and additional syntax governing the request/response exchange can essentially include anything a particular SHAR and AA mutually understand; in practice, Shibboleth defines one or more protocols for this exchange. See section 6.3 for specific information.

There are 3 steps involved in responding to an AQM:

1. Authenticate the requesting SHAR
2. Find the releasable attributes associated with the handle presented in the AQM
3. Return a response message (an ARM) to the requesting SHAR

Each of these steps is described in detail below.

5.6.1.1 Authenticating the SHAR

As described in section 5.5.1.4, any exchange protocol MUST provide for mutual authentication; this insures that any successful request will be associated with the requesting SHAR's identity or will be anonymous. The AttributeQuery's Subject element contains the attribute request handle (or X.509 certificate) for the browser user. The AQM also optionally contains the resource URL requested by the browser user at the time of the attribute query.

5.6.1.2 Finding releasable attributes associated with the handle

Most of the work in responding to a request is the process of obtaining the attributes associated with the handle, locating the appropriate policies that control attribute release, and applying those policies to arrive at a final set of information to return to the SHAR. The three steps in this process are described below, followed by an example. (Note that the process described is intended to illustrate the high-level steps to be followed and does not imply specific implementations of attribute and ARP storage, access, or structure.)

1) Find the user associated with the handle

The handle or certificate in the query message is mapped to the user's actual identity. The identity is used to find the user's ARPs and to determine the set of attributes and attribute values that might potentially be returned to the requesting SHAR.

2) Determine the effective ARP for the request

The AA must identify the subset of information to return by determining the ARP to apply to the request. In the simplest case, this ARP may be an explicit rule that identifies the SHAR by name and enumerates what to release. An ARP could instead be associated with a group of SHARs using a wildcarded expression, or a more arbitrary grouping based on real world designations like academic consortia (e.g. CIC or Ivy League) or corporate relationships.

Taken further, the ARP could even be a derived, or computed, policy that is the result of evaluating a combination of rules. These rules could be simple policies such as the ARPs described in the previous paragraph, or they could be more complex rules that take additional factors into account and combine in various ways to form an effective policy that applies to a given request. Section 5.6.1.5 describes a few examples of more advanced policies.

Despite the freedom left to implementers, when a request arrives at the AA, the AA must be able to determine the policy to apply based on the user's identity, the SHAR's identity, the resource being accessed, and any other available factors related to other institutional requirements.

As an example believed to be a common use case, here is a set of matching rules for finding the applicable ARP in light of possible wildcarding of both SHARs (e.g. *.edu) and target URLs ("http://www.jhu.edu/*") in ARPs. Many complex policy designs may be reducible into these simpler terms, which can subsequently be evaluated using this matching process to locate the appropriate policy to apply.

- The authenticated identity of the requesting SHAR (or "anonymous") is used to locate any ARPs that apply specifically to that SHAR, with no wildcarding.
- If one or more exact matches exist, then within that set, find an ARP which best (i.e. most specifically) matches the target URL, if provided by the SHAR. (If the target URL is not provided, then only an ARP with a wildcarded target URL of '*' is considered to match.)
- If an ARP is found to match, then it is applied to the request in step 3, "Validating the Attributes".
- If instead, no matches for the target URL exist, or if no ARPs apply specifically to the SHAR, then select the ARP that best matches the requesting SHAR using a wildcard, and apply it to the request in step 3.
- If none match, then no ARP can be selected for the request, and the AA returns a successful response with no attributes enclosed (see section 5.6.1.3).

3) Validate the attributes

Whatever the range of policy expressions and combinations supported by the AA, if the previous step identifies an ARP to apply, that ARP will specify the attributes (possibly restricted to specific values) to return to the SHAR. The AA must now verify that these attributes and corresponding values truly still belong to the user. This is especially relevant if an attribute in an ARP refers to a highly changeable role such as membership in a course. If no ARP applies to the request, or if the attributes specified by the ARP are no longer valid, then no attributes are returned.

Note that the AA may obtain the user's actual attribute information prior to determining the effective ARP, and may also implement some degree of caching. This is an implementation choice. However, the AA must take some care if highly changeable attributes are cached.

5.6.1.3 Return a response message

The allowable attributes, if any, are packaged along with some additional information into the ARM. The contents of this message are summarized in section 5.5.1.3 and specified in section 6.1.2. The exchange protocol in use may specify additional information and semantics for the response. If no attributes are to be returned, a complete ARM with no attributes MUST be returned, with a StatusCode of "samlp:Success".

5.6.1.4 Processing Example

To help illustrate the rules for processing an AQM, consider an example request containing the following parameters:

- Handle: refers to Professor Mary Smith
- SHAR: www.jhu.edu
- Target URL: http://www.jhu.edu/research/diseases/ALS

Assume the following three ARPs exist at the AA:

SHAR: *.edu
URL: *
Release: Affiliation=faculty

SHAR: www.jhu.edu
URL: http://www.jhu.edu/research/diseases/MultipleSclerosis/*
Release: Username, Role=MS Researcher

SHAR: www.jhu.edu
URL: http://www.jhu.edu/research/diseases/*
Release: Role=MS Researcher

Now, what gets released in response to this request? All three ARPs match the requesting SHAR, but the last two match exactly and so the AA examines them first. Of these two, the last ARP matches the target URL and therefore "Role=MS Researcher" is released.

If, instead, the target URL in the request was "http://www.jhu.edu/research/", neither of the exact SHAR matches would apply. Searching would continue, the first ARP would be found to match, and therefore "Affiliation=faculty" would be released instead.

5.6.1.5 Additional ARP Concepts

The Shibboleth specification provides examples throughout of a simple ARP design focused on privacy-oriented control over attribute release to specific (or groups of) destinations. An AA may support and apply other kinds of policies to control attribute release. For instance, an AA may support more than one "type" of ARP.

A likely example would be differentiating between ARPs set up by and for individual users, and those that might be set up by an administrator to apply to all users. This latter type of ARP might be called an "institutional ARP", reflecting (as an example) out-of-band agreements and contracts between institutions of higher education and commercial information providers. There may be other types of useful ARPs that emerge as institutions and users start using Shibboleth.

As a matter of local policy, these institutional ARPs could override individual ARPs or could add the release of attributes that might not otherwise be exposed to the user to choose for release. For example, an origin site that had an agreement with a particular database vendor might force the release of the attribute "contract=#1234".

Since there could be more than one identical ARP, distinguished only by type, the question arises as to which one takes precedence if the specified SHAR makes a request. Another question, relating to the matching rules described in section 5.6.1.2, is whether an ARP of one type that matches by means of a wildcard can override a specifically matching ARP of a different type.

Shibboleth doesn't specify or constrain how an AA can answer these kinds of questions. AA implementers are free to support many different kinds of ARPs with varying semantics as long as the AA can efficiently process requests and determine the effective policy to apply.

5.6.2 Providing the Interface to Specify ARPs

The AA's primary interactive function is to support the specification of ARPs by users and administrators. The specifics are left up to implementations, but AA implementers SHOULD provide a web-based interface for ARP maintenance.

One important note made previously is that users generally shouldn't have to interact with and view ARPs in terms of specific SHARs, but rather in terms of target hosts and URLs, or perhaps even more advantageously in terms of application domain names like "Encyclopedia Frobozzica," or logical groups of SHARs like "Ohio Public Colleges" or "Preferred Vendors."

From an administrative perspective, a given URL tree is associated with one (and only one) specific SHAR. In the absence of overriding information, the SHAR name is assumed to match the hostname in the URL. If a user specifies a new ARP for a target URL that is known to be associated with a different SHAR, the AA SHOULD internally make the appropriate association in the ARP without needing to make the user aware of this distinction.

As section 5.5.4 describes, even within the set of URLs associated with a specific SHAR, subdivisions along application and policy lines may exist that dictate how attributes are cached and used at the SHAR. It is to the user's benefit if these divisions are known by the AA in order to prevent misspecification of policy. If a user tries to create an ARP that differs from an existing ARP that references the same SHAR and application domain, the AA SHOULD alert the user to the policy conflict.

5.6.3 Default and Wildcarded ARPs

The use of default and wildcarded ARPs is of great importance in simplifying the task of administration for both end users and AA administrators. Defaults and wildcarding allow policy to be aggregated across resources, saving time and making it easier to verify that appropriate policies are in effect without manual intervention.

The AA SHOULD allow the user to select intelligent default ARPs and SHOULD have a "default" default setting.

5.6.3.1 AA Default ARP

The AA ought to return something (if possible) rather than an error message in response to an AQM from a SHAR that isn't designated in any of the user's ARPs, or from an anonymous SHAR. "Affiliation=MemberOfCommunity," or something similar, is an example of a good default attribute. This policy would look like the following, expressed as an ARP.

SHAR: *
URL: *
Release: Affiliation=MemberOfCommunity

The implication for the AQM processing described in section 5.6.1 is that any and all SHARs should receive this attribute/value pair, in the absence of a more specific ARP.

5.6.3.2 User Default ARPs

A user's default policies might include:

1. A partially wildcarded SHAR:

   SHAR: *.edu
   URL: *
   Release: Affiliation

The meaning of this ARP is that each of the user's "Affliliation" attribute values (e.g. faculty, staff, member) should be released to any SHAR that represents an educational institution, assuming that only such institutions are able to authenticate as a SHAR with a name ending in ".edu".

An AA allowing ARPs with broadly wildcarded SHAR names must be very sure that certificates with matching names (or whatever the underlying authentication credential happens to be) are all issued with an acceptable policy. Allowing ARPs with very broad wildcarding of SHAR names (e.g. a*.com) is likely to lead to "interesting" and possibly unwanted behavior, since the SHAR name is the basis for the attribute release decision.

2. A wildcard in the Release field:

   SHAR: www.jhu.edu
   URL: http://www.jhu.edu/research/diseases/MultipleSclerosis
   Release: *

This ARP means that all of the user's applicable attributes should be sent to www.jhu.edu. Note that this doesn't mean that every possible piece of information about the user should be sent. The number of attributes that the AA considers to be applicable should be far smaller than the number that might be kept in a typical enterprise directory, though Shibboleth doesn't limit or specify which attributes can be sent by an AA in partnership with a destination site.

Thus, while some might think that "hair color" would a silly attribute to send, an AA may send "hair color=mousy brown" and an RM at a destination site could base an access control decision or application behavior on this bit of information. However, each attribute supported by the AA should be "well-considered" to avoid a burden on both administrators and users.

5.6.3.3 Implied Wildcarding of URLs

A URL specified in an ARP implicitly ends with the wildcard '*' (i.e. 'match any'), because it refers to an application domain. Recall that an application domain is a set of resources managed by the same RM (and in the end, the same human administrators). The ARP applies to any target resource whose URL matches the URL expression in the ARP. Recall that although multiple ARPs may match a given target URL, the AA will choose the ARP that is the best match.

5.6.3.4 Misuse of Defaults

Some wildcarded ARPs that may on the surface seem acceptable are on closer inspection not legitimate. For example, a wildcarded SHAR name (e.g. "*") combined with a specific URL doesn't make sense since any single URL is associated with a specific SHAR. A SHAR specification that includes a wildcard character MUST have an empty (or "*") URL value.

5.6.4 Error Handling

Since only a limited amount of information is provided as part of a query, most errors during the attribute exchange are typically either protocol issues like authentication or network failures, or origin site problems, such as an inaccessible attribute repository. Each binding protocol MUST clearly define the errors it will communicate to the SHAR and whether or not the operation should be retried. The specific codes and mechanisms by which any success or failure conditions are communicated back to the SHAR (beyond high-level SAML processing) MUST also be specified as part of the binding protocol.

The AA MUST detect an expired or invalid handle or certificate in the AttributeQuery Subject and communicate either condition to the SHAR using a top level StatusCode of "samlp:Requester" and a second level StatusCode of "shib:InvalidHandle", where "shib" denotes the Shibboleth XML namespace of "urn:mace:shibboleth:1.0".

If the AA supports real-time release of attributes by users (see section 5.5.2.1), then it MAY also communicate this error condition to the SHAR along with the URL at which it will mediate the user interaction. The SHAR is not required to check for this condition, and MAY instead fail the user's request.

5.6.5 SHAR/Target Association

As discussed earlier, if the name of the SHAR responsible for a particular target URL is not determinable directly from the URL's hostname, this information MUST be known to the AA before an ARP for that target can be properly enforced. Shibboleth does not specify a standard mechanism for establishing this association; it is assumed to be a manual exception-driven process that is the responsibility of AA administrators. It is obviously incumbent upon them to acquire the necessary evidence by which the association should be accepted as "valid," perhaps a signed email message or a document in writing.

Section 5.5.4 discusses the possible division of a single URL tree into distinct application domains behind a single SHAR, and these distinctions SHOULD be known by AAs in order to present the most consistent and accurate view possible to the user without burdening him/her with details of the destination site's implementation. This aspect of Shibboleth is expected to evolve as further experience is gained.

5.6.6 AA Security Considerations

The attributes that an AA sends to a SHAR will be used to grant (or deny) access to resources, which might include research data, student loan information, grades, and other important information and services. Thus, these attributes are nearly as sensitive as user passwords. Origin site administrators must ensure that administrative access to the AA and to the AA's source of attributes (be it a persistent store like a database or an LDAP directory, or something else entirely), is highly secured.

Poor security practices will increase the risk that the "wrong" attribute information could be associated with a user. Attributes that don't really belong to the user could be sent, thus empowering a user who shouldn't be empowered, or attributes that the user doesn't want sent (e.g. username) could be released, thus violating the user's privacy expectations.

This is not an exhaustive list, but some "good practices" include the following:

- Auditing modifications to ARPs
- Auditing modifications to "meta policies" (e.g. policies about ARP precedence)
- Auditing modifications to the list of AA administrators and other roles
- Auditing modifications to attribute data, if appropriate and feasible

6 Message Formats and Protocol Specifications

The following XML namespace prefix assignments for XML Signature and SAML are used in the enumeration of elements in this section:

- xmlns:ds = "http://www.w3.org/2000/09/xmldsig#"
- xmlns:saml = "urn:oasis:names:tc:SAML:1.0:assertion"
- xmlns:samlp = "urn:oasis:names:tc:SAML:1.0:protocol"

Unless otherwise specified, all attributes are unqualified.

6.1 Attribute Query and Attribute Response Messages

The SHAR and AA components communicate by exchanging SAML messages using any shared protocol that supports the required functional characteristics. The complete syntax of the AQM and ARM depend on the protocol used, but all protocols MUST share the core AQM/ARM syntax and semantics described in the following two subsections.

6.1.1 Attribute Query Message (AQM) Common Syntax

The AQM is sent by a SHAR to an AA. The complete syntax of an AQM depends on the exchange protocol, but all AQM formats MUST include a single XML element adhering to the SAML schema for a Request containing an AttributeQuery, as defined in **[SAML]**. Guidance on usage of the schema definition by Shibboleth components is as follows:

EL	samlp:Request	MUST appear once and only once

ATT	RequestID	
ATT	MajorVersion	MUST equal “1”
ATT	MinorVersion	MUST equal “0”
ATT	IssueInstant	MUST equal the current GMT date and time
EL	samlp:RespondWith*	MAY appear zero or more times MUST connote a SAML AttributeStatement, if used
EL	ds:Signature?	MAY contain an XML signature MAY include an X.509 certificate
EL	samlp:AttributeQuery	
ATT	Resource	SHOULD contain the target URL, if applicable
EL	saml:Subject	See section 6.2 for subject information

In all respects, the Request element and its contents MUST conform to all schematic and semantic requirements described in **[SAML]**, and nothing in this specification should be taken to mean otherwise.

6.1.2 Attribute Response Message (ARM) Common Syntax

The ARM is sent by an AA to a SHAR in response to an AQM. The complete syntax of an ARM depends on the exchange protocol, but all ARM formats MUST include a single XML element adhering to the SAML schema for a Response, containing an Assertion, itself containing an AttributeStatement, as defined in **[SAML]**. Guidance on usage of the schema definition by Shibboleth components is as follows:

EL	samlp:Response	MUST appear once and only once
ATT	ResponseID	
ATT	InResponseTo	MUST equal RequestID in AQM
ATT	MajorVersion	MUST equal “1”
ATT	MinorVersion	MUST equal “0”
ATT	IssueInstant	MUST equal the current GMT date and time
EL	ds:Signature?	MAY contain an XML signature MAY include an X.509 certificate
EL	saml:Assertion	
ATT	AssertionID	
ATT	MajorVersion	MUST equal “1”

ATT	MinorVersion	MUST equal “0”
ATT	Issuer	
ATT	IssueInstant	SHOULD equal GMT date and time of statement generation
EL	saml:AttributeStatement	MUST appear once and only once
EL	saml:Subject	See section 6.2 for subject information
EL	saml:Attribute*	See section 6.1.2.1 for attribute information
EL	saml:Conditions	
ATT	NotBefore	SHOULD be omitted, MUST be in the past
ATT	NotOnOrAfter	MAY be used to signify attribute expiration
EL	ds:Signature?	MAY contain an XML signature MAY include an X.509 certificate

In all respects, the Response element and its contents MUST conform to all schematic and semantic requirements described in **[SAML]**, and nothing in this specification should be taken to mean otherwise.

6.1.2.1 Attribute Syntax

The attributes returned by the AA are a sequence of zero or more Attribute elements as described above. **[SAML]**§ 2.4.5.1 defines an attribute syntax consisting of a pair of XML attributes, AttributeName and AttributeNamespace, and an enclosed AttributeValue element. The AttributeValue contains arbitrary XML content. The attribute’s name and namespace MUST be sufficient to uniquely identify to the SHAR what to expect in the AttributeValue and how to process it.

Section 3.3.3 describes the notion of attribute acceptance policies. Shibboleth does not specify how a SHAR or RM decides whether or not to accept an attribute from an AA, but the combination of the attribute’s namespace, name, value content, and the sending origin site is presumably sufficient to make a determination as to the worthiness of an attribute.

6.2 SAML Subject Usage

SAML assertions generally include one or more Statements about a Subject, and the assertions used in Shibboleth use the Subject element consistently to represent the “blinded” identity of the browser user. The user is assigned a handle by his/her origin site for some period of time, and the handle is embedded in assertions referring to this user in the form of a name. The name is understood by Shibboleth components to have certain properties, among them a lack of persistence that makes it impractical to use in an access control policy statement.

The use of the Subject element in the Shibboleth assertions and query described in sections 6.1 and 6.4 is in accordance with **[SAML]**§ 2.4.2.1 and is as follows:

EL	saml:Subject	
EL	saml:NameIdentifier	MUST contain the handle issued by the origin site
ATT	NameQualifier	MUST contain the domain name of the user's origin site
ATT	Format	SHOULD be omitted, usage is undefined if provided

6.3 Attribute Query and Response Exchange Protocols

Although Shibboleth does not require the use of a specific protocol between SHAR and AA, in the interest of interoperability, one protocol is designated as mandatory for all compliant implementations to support. Other optional protocols may be defined, as long as they provide the necessary application and security semantics.

6.3.1 SAML SOAP/HTTP Protocol

[SAMLBind]§ 3.1 specifies a SAML request/response protocol bound to a SOAP 1.1 envelope over HTTP. Shibboleth SHAR and AA implementations MUST support this protocol over HTTP using SSLv3 or TLS. Use of this protocol is completely in accordance with the SAML specification. If client or server certificates are used in conjunction with SSL, then use of XML signatures to digitally sign the messages is optional. Use of XML signatures, if used, MUST be in accordance with **[SAML]**§ 5 and **[XMLSig]**.

The URI "urn:oasis:names:tc:SAML:1.0:bindings:SOAP-binding" identifies this protocol when referenced from within the AuthorityBinding element described in section 6.4.

6.4 Attribute Query Handle Presentation

Sections 5.2.1.3 and 5.4.4 describe the use of the SAML Browser/POST profile for delivery of a newly assigned Shibboleth user handle to a SHIRE to establish a new context for the user at a destination site. The "handle presentation" requires that the handle be embedded within an XML instance document conforming to the SAML schema for a Response containing an Assertion containing an AuthenticationStatement element. In addition, the issuing HS MUST digitally sign the XML instance. Guidance on usage of the schema definition by Shibboleth components is as follows:

EL	samlp:Response	
ATT	ResponseID	
ATT	MajorVersion	MUST equal "1"
ATT	MinorVersion	MUST equal "0"
ATT	IssueInstant	MUST equal current GMT date and time
ATT	Recipient	MUST equal SHIRE acceptance URL

EL	ds:Signature	MUST contain an XML signature MAY include an X.509 certificate
EL	saml:Assertion	
ATT	AssertionID	
ATT	MajorVersion	MUST equal “1”
ATT	MinorVersion	MUST equal “0”
ATT	Issuer	MUST contain name of issuing HS
ATT	IssueInstant	MUST be in the past
EL	saml:Conditions	
ATT	NotBefore	MUST be in the past
ATT	NotOnOrAfter	MUST be in the future
EL	saml:AudienceRestrictionCondition?	MAY appear zero or once
EL	saml:Audience*	MAY enumerate set of applicable policies
EL	saml:AuthenticationStatement	MUST appear at least once with the described content
ATT	AuthenticationMethod	MUST identify how user authenticated
ATT	AuthenticationInstant	MUST equal GMT date and time of user's authentication
EL	saml:Subject	See section 6.2 for general subject usage
EL	saml:ConfirmationMethod	MUST equal SAML assertion bearer URI
EL	saml:SubjectLocality?	MAY be omitted
ATT	IPAddress	MUST equal client’s dotted decimal IP address
EL	saml:AuthorityBinding*	MAY appear zero or more times MAY include any legal SAML usage, but at least one SHOULD fulfill the requirements below
ATT	AuthorityKind	MUST connote a SAML AttributeQuery
ATT	Binding	MUST identify a SHAR/AA protocol
ATT	Location	MUST point to AA via the chosen protocol

EL	ds:Signature?	MAY contain an XML signature MAY include an X.509 certificate

In all respects, the Response element and its contents MUST conform to all schematic and semantic requirements described in **[SAML]**, and nothing in this specification should be taken to mean otherwise.

6.5 SHAR/RM Interface

The SHAR is responsible for insuring that a set of attributes provided by an AA are associated with each individual HTTP request. Before the request is handed off by the web server to the RM, the SHAR MAY create an HTTP request header named "Shib-Attributes" and append it to the list of headers associated with the request. The header, if present, MUST contain a base64-encoded XML instance containing a SAML Assertion element. The assertion MUST contain at least one AttributeStatement, and SHOULD be identical to the assertion received from the AA, if possible. If the SHAR does not create this request header, then it MUST insure that any existing request header with the same name is destroyed before passing on the request.

7 References

[RFC 2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997

[RFC 2396] T. Berners-Lee et. al., *Uniform Resource Identifiers (URI): Generic Syntax* http://www.ietf.org/rfc/rfc2396.txt IETF RFC 2396

[SAML] *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)*, http://www.oasis-open.org/committees/security/docs/cs-sstc-core-00.pdf, OASIS, April 2002.

[SAMLBind] *Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML)*, http://www.oasis-open.org/committees/security/docs/cs-sstc-bindings-00.pdf, OASIS, April 2002.

[XMLSig] D. Eastlake et al., *XML-Signature Syntax and Processing*, http://www.w3.org/TR/xmldsig-core/, World Wide Web Consortium.

8 Acknowledgements

The authors of this document are Marlena Erdos (Tivoli Systems Inc, an IBM company) and Scott Cantor (The Ohio State University).

The major contributors to the Shibboleth architecture are (in reverse alphabetical order):

- David Wasley (david.wasley@ucop.edu), University of California
- RL 'Bob' Morgan (rlmorgan@washington.edu), University of Washington

- Keith Hazelton (hazelton@doit.wisc.edu), University of Wisconsin, Madison
- Marlena Erdos (marlena@us.ibm.com), Tivoli Systems Inc, an IBM company.
- Steven Carmody (Steven_Carmody@brown.edu), Brown University and Internet2
- Scott Cantor (cantor.2@osu.edu), The Ohio State University

Other technical contributors include Michael Gettes (Georgetown University), Ken Klingenstein (Internet2), and Scott Fullertown (University of Wisconsin, Madison).

Ken Klingenstein has also provided on-going administrative leadership.

Other contributors that helped produce the architecture include Renee Frost (Internet2), Ben Chinowsky (Internet2), and Nate Klingenstein (Internet2). Thank you!

The authors wish to additionally thank Nathan Kane for help with the initial illustrations of the architecture. The authors also wish to thank David Wasley for supplying the final versions of the illustrations.