Shibboleth Architecture

Technical Overview

Working Draft 02, 8 June 2005

Document identifier:

draft-mace-shibboleth-tech-overview-02

Location:

http://shibboleth.internet2.edu/shibboleth-documents.html

Editors:

Tom Scavo (trscavo@ncsa.uiuc.edu), NCSA

Scott Cantor (cantor.2@osu.edu), The Ohio State University

Contributors:

Nathan Dors (dors@cac.washington.edu), University of Washington

Abstract:

This non-normative document gives a technical overview of Shibboleth.

This is a **working draft** and the text may change before completion. Please submit comments to the shibboleth-dev mailing list (see http://shibboleth.internet2.edu/ for subscription details).

Table of Contents

1 Introduction

The *Shibboleth Architecture* [ShibProt] extends the SAML 1.1 single sign-on and attribute exchange mechanisms by specifying service-provider-first SSO profiles and enhanced features for user privacy. This document provides a technical overview of Shibboleth and as such is an extension of [SAMLTech].

1.1 Background

The *Shibboleth Architecture* is built upon the following standards:

- Hypertext Transfer Protocol (HTTP)
- Extensible Markup Language (XML)
- XML Schema
- XML Signature
- SOAP[1]
- Security Assertion Markup Language (SAML)

Definitive references for each of these standards are found in section 7 of this Overview.

1.2 Notation

Conventional XML namespace prefixes are used throughout the listings in this Overview, whether or not a namespace declaration is present in the listing:

- The prefix `saml:` stands for the SAML 1.1 assertion namespace:

  ```
  urn:oasis:names:tc:SAML:1.0:assertion
  ```

- The prefix `samlp:` stands for the SAML 1.1 request-response protocol namespace:

  ```
  urn:oasis:names:tc:SAML:1.0:protocol
  ```

- The prefix `md:` stands for the SAML 2.0 metadata namespace:

  ```
  urn:oasis:names:tc:SAML:2.0:metadata
  ```

- The prefix `ds:` stands for the W3C XML Signature namespace:

  ```
  http://www.w3.org/2000/09/xmldsig#
  ```

- The prefix `xsd:` stands for the W3C XML Schema namespace:

  ```
  http://www.w3.org/2001/XMLSchema
  ```

- The prefix `xsi:` stands for the W3C XML Schema namespace for schema-related markup that appears in XML instances:

  ```
  http://www.w3.org/2001/XMLSchema-instance
  ```

The following typographical conventions for *displayed text* are used:

```
HTTP requests appear like this.
HTTP responses appear like this.
HTML code listings appear like this.
JavaScript code listings appear like this.
XML code listings appear like this.
SAML code listings appear like this.
```

[1] Originally, SOAP was an acronym for "Simple Object Access Protocol", but this turned out to be a misnomer since SOAP has nothing to do with "object access" in the object-oriented programming sense of the phrase. So today we no longer think of "SOAP" as an acronym although the capitalization persists.

```
URNs appear like this.
URLs appear like this.
```

Likewise the following typographical conventions for *in-line text* are used: `<XMLelement>`, `<HTMLelement>`, `XMLattribute`, `HTMLattribute`, `LDAPattribute`, `HTTPparameter`, `OtherCode`. Finally, variable placeholders in code fragments (both displayed and in-line) are italicized (e.g., *`artifact`*).

1.3 Outline

Immediately following this introduction, in section 2 we describe the components comprising the *Shibboleth System*. Section 3 is a general introduction to *Security Assertion Markup Language* (SAML) with special emphasis on those SAML constructs used by Shibboleth. In section 4, the Shibboleth *browser profiles* are described in detail, while the Shibboleth *attribute profiles* are outlined in section 5. Also in section 5, we emphasize the importance of *directory schema* using *eduPerson* as the canonical example of a directory in higher education. An introduction to SAML *metadata* is given in section 6, and finally in section 7 we provide a list of definitive *references* for further study.

2 Shibboleth Components

The functional components of a conforming Shibboleth implementation include an identity provider, a service provider, an optional “Where are you from?” (WAYF) service, and various interacting subcomponents.

2.1 Identity Provider

An *identity provider* (formerly called an *origin*) maintains user credentials and attributes. Upon request the identity provider (IdP) will assert authentication statements or attribute statements to relying parties, specifically service providers. IdP subcomponents are depicted in Figure 1 and discussed in the following subsections.

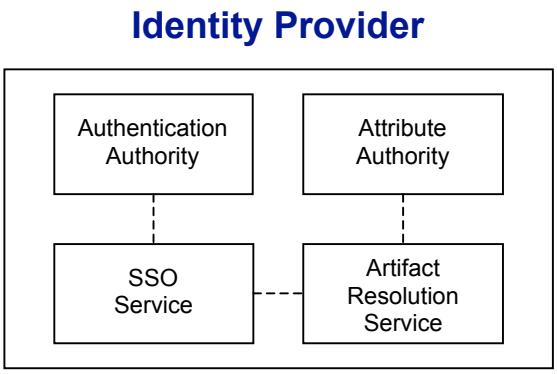

Figure 1: Shibboleth Identity Provider

2.1.1 Authentication Authority

The *authentication authority* issues authentication statements to other components. The authentication authority is integrated with the IdP's authentication service (the details of which are out of scope).

2.1.2 Single Sign-On Service

A *single sign-on (SSO) service* is the first point of contact at the IdP. The SSO service initiates the authentication process at the IdP and ultimately redirects the client to the inter-site transfer service (unless the function of the SSO service and inter-site transfer service are combined, which is encouraged). Note: The SSO service is not defined by SAML 1.1, which specifies only identity-provider-first SSO profiles.

2.1.3 Inter-Site Transfer Service

The *inter-site transfer service* issues HTTP responses conforming to the Browser/POST and Browser/Artifact profiles. The inter-site transfer service interacts with the authentication authority behind the scenes to produce the required authentication assertion. Note: The concept of an inter-site transfer service has been removed in SAML 2.0, so we assume that the functions of the SSO service and the inter-site transfer service are combined, and therefore ignore the latter in all that follows.

2.1.4 Artifact Resolution Service

If the Browser/Artifact profile is used, the IdP sends an artifact to the SP instead of the actual assertion. (An *artifact* is a reference to an authentication assertion.) The SP then sends the

artifact to the *artifact resolution service* at the IdP via a back-channel exchange. In return, the IdP sends the required authentication assertion to the SP.

2.1.5 Attribute Authority

The *attribute authority* processes attribute requests; that is, it issues attribute assertions. The attribute authority authenticates and authorizes any requests it receives.

2.2 Service Provider

A *service provider* (formerly called a *target*) manages secured resources. User access to resources is based on assertions received by the service provider (SP) from an identity provider. SP subcomponents are shown in Figure 2. Note that the service provider has access control in place that prevents access by clients without a security context.

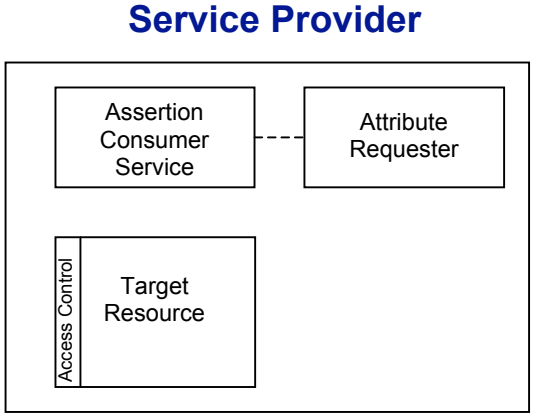

Figure 2: Shibboleth Service Provider

2.2.1 Assertion Consumer Service

The *assertion consumer service* (formerly called a *SHIRE*) is the service provider endpoint of the SSO exchange. It processes the authentication assertion returned by the SSO service (or artifact resolution service, depending on the profile used), initiates an optional attribute request (see section 5), establishes a security context at the SP, and redirects the client to the desired target resource.

2.2.2 Attribute Requester

An *attribute requester* (formerly called a *SHAR*) at the SP and the attribute authority at the IdP may conduct a back-channel attribute exchange once a security context has been established at the SP. That is, the SP and IdP interact directly, bypassing the browser.

2.3 WAYF Service

An optional *WAYF service* is operated independent of the SP and IdP. The WAYF can be used by the SP to determine the user's preferred IdP, with or without user interaction. The WAYF is essentially a proxy for the authentication request passed from the SP to the SSO service at the IdP.

3 SAML Assertions

This section provides some background material on SAML assertions that sets the stage for the profiles to follow. In Shibboleth, *SAML assertions* are transferred from identity providers to service providers. Assertions contain statements that service providers can use to make access control decisions. Three types of statements are specified by SAML:

1. Authentication statements
2. Attribute statements
3. Authorization decision statements

Authentication statements assert to the service provider that the principal did indeed authenticate with the identity provider at a particular time using a particular method of authentication. The examples in section 3.1, for instance, are assertions containing authentication statements typically used in the Browser/POST and Browser/Artifact profiles (section 4), respectively.

Attribute statements provide additional information about a principal so that service providers can make access control decisions. For example, the attribute assertion given in section 3.2 is the result of the attribute exchange described in section 5.1.

Authentication statements and attribute statements may be combined in a single assertion. Alternatively, multiple assertions may be issued by the IdP, each containing its own statement. A variation of the Browser/Artifact profile (section 5.2), for instance, relies on a pair of assertions, both obtained as the result of a single message exchange. The example in section 3.3 illustrates the corresponding pair of assertions.

In some situations, it may be preferable for the application to delegate an access control decision to another component or service. In this case, the service provider indicates to the service the resource being accessed where after the service asserts an *authorization decision statement* that dictates whether or not the principal is allowed access to the resource. In Shibboleth, such statements are out of scope, but the interested reader is referred to [GSIsecurity] for a relevant example.

Finally, SAML assertions may be digitally signed. An XML element such as that given in section 3.4 is inserted into those assertions where additional message integrity is required.

3.1 Authentication Assertions

The following *authentication assertion* (sometimes called an *SSO assertion*) contains a `<saml:AuthenticationStatement>` element (but no attributes):

```
<saml:Assertion
  xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
  MajorVersion="1" MinorVersion="1"
  AssertionID="a75adf55-01d7-40cc-929f-dbd8372ebdfc"
  IssueInstant="2004-12-05T09:22:02Z"
  Issuer="https://idp.example.org/shibboleth">
  <saml:Conditions
    NotBefore="2004-12-05T09:17:02Z"
    NotOnOrAfter="2004-12-05T09:27:02Z">
    <saml:AudienceRestrictionCondition>
      <saml:Audience>http://sp.example.org/shibboleth</saml:Audience>
    </saml:AudienceRestrictionCondition>
  </saml:Conditions>
  <saml:AuthenticationStatement
    AuthenticationInstant="2004-12-05T09:22:00Z"
```

```
    AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
    <saml:Subject>
      <saml:NameIdentifier
        Format="urn:mace:shibboleth:1.0:nameIdentifier"
        NameQualifier="https://idp.example.org/shibboleth">
        3f7b3dcf-1674-4ecd-92c8-1544f346baf8
      </saml:NameIdentifier>
      <saml:SubjectConfirmation>
        <saml:ConfirmationMethod>
          urn:oasis:names:tc:SAML:1.0:cm:bearer
        </saml:ConfirmationMethod>
      </saml:SubjectConfirmation>
    </saml:Subject>
  </saml:AuthenticationStatement>
</saml:Assertion>
```

The value of the `<saml:NameIdentifier>` element in the preceding example is called a Shibboleth *handle*: an opaque, transient identifier associated with the authenticated user. Further messages regarding this user (such as the one in the next section) explicitly refer to this handle instead of the actual identity of the user.

Note that the above assertion (called a *bearer assertion*) assigns the following value to the `<saml:ConfirmationMethod>` element:

```
urn:oasis:names:tc:SAML:1.0:cm:bearer
```

Whereas the preceding assertion is used in the Browser/POST profile (section 4.2), the following assertion is used in the Browser/Artifact profile (section 4.3):

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

In this case, the value of the <saml:ConfirmationMethod> element is

```
urn:oasis:names:tc:SAML:1.0:cm:artifact
```

which indicates the Browser/Artifact profile was used to retrieve the assertion.

Some use cases require more than an opaque identifier, so for the sake of illustration the value of the <saml:NameIdentifier> element in the preceding example is an email address. Presumably this uniquely identifies the authenticated user to the SP.

Note that an authentication assertion may include additional information about the subject, such as attributes.

3.2 Attribute Assertions

The following *attribute assertion* contains a <saml:AttributeStatement> element only:

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

Note that the value of the <saml:NameIdentifier> element is the same as that in the first example in section 3.1, which implies that the attribute statement is obtained subsequent to and as a consequence of the previous authentication statement.

3.3 Multiple Assertions

The following pair of assertions contain <saml:AuthenticationStatement> and <saml:AttributeStatement> elements, respectively:

```
<saml:Assertion
  xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
  MajorVersion="1" MinorVersion="1"
```

```
  AssertionID="00305ed1-a1bd-10f9-a2d0-004005b13a2b"
  IssueInstant="2004-12-05T09:22:05Z"
  Issuer="https://idp.example.org/shibboleth">
  <saml:Conditions
    NotBefore="2004-12-05T09:17:05Z"
    NotOnOrAfter="2004-12-05T09:27:05Z">
    <saml:AudienceRestrictionCondition>
      <saml:Audience>http://sp.example.org/shibboleth</saml:Audience>
    </saml:AudienceRestrictionCondition>
  </saml:Conditions>
  <saml:AuthenticationStatement
    AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password"
    AuthenticationInstant="2004-12-05T09:22:00Z">
    <saml:Subject>
      <saml:NameIdentifier
        Format="urn:mace:shibboleth:1.0:nameIdentifier"
        NameQualifier="https://idp.example.org/shibboleth">
        082dd87d-f380-4fd6-8726-694ef2bb71e9
      </saml:NameIdentifier>
      <saml:SubjectConfirmation>
        <saml:ConfirmationMethod>
          urn:oasis:names:tc:SAML:1.0:cm:artifact
        </saml:ConfirmationMethod>
      </saml:SubjectConfirmation>
    </saml:Subject>
  </saml:AuthenticationStatement>
</saml:Assertion>
<saml:Assertion
  xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  MajorVersion="1" MinorVersion="1"
  AssertionID="008c6181-a288-10f9-b6d6-004005b13a2b"
  IssueInstant="2004-12-05T09:22:05Z"
  Issuer="https://idp.example.org/shibboleth">
  <saml:Conditions
    NotBefore="2004-12-05T09:17:05Z"
    NotOnOrAfter="2004-12-05T09:52:05Z">
    <saml:AudienceRestrictionCondition>
      <saml:Audience>http://sp.example.org/shibboleth</saml:Audience>
    </saml:AudienceRestrictionCondition>
  </saml:Conditions>
  <saml:AttributeStatement>
    <saml:Subject>
      <saml:NameIdentifier
        Format="urn:mace:shibboleth:1.0:nameIdentifier"
        NameQualifier="https://idp.example.org/shibboleth">
        082dd87d-f380-4fd6-8726-694ef2bb71e9
      </saml:NameIdentifier>
    </saml:Subject>
    <saml:Attribute
      AttributeName="urn:mace:dir:attribute-def:eduPersonScopedAffiliation"
      AttributeNamespace="urn:mace:shibboleth:1.0:attributeNamespace:uri">
      <saml:AttributeValue Scope="example.org">
        member
      </saml:AttributeValue>
      <saml:AttributeValue Scope="example.org">
```

```
        faculty
      </saml:AttributeValue>
    </saml:Attribute>
    <saml:Attribute
      AttributeName="urn:mace:dir:attribute-def:eduPersonEntitlement"
      AttributeNamespace="urn:mace:shibboleth:1.0:attributeNamespace:uri">
      <saml:AttributeValue xsi:type="xsd:anyURI">
        urn:mace:oclc.org:100277910
      </saml:AttributeValue>
      <saml:AttributeValue xsi:type="xsd:anyURI">
        https://sp.example.org/entitlements/123456789
      </saml:AttributeValue>
      <saml:AttributeValue xsi:type="xsd:anyURI">
        urn:mace:incommon:entitlement:common:1
      </saml:AttributeValue>
    </saml:Attribute>
  </saml:AttributeStatement>
</saml:Assertion>
```

3.4 Signed Assertions

Many SAML assertions, especially assertions that pass through a browser, are digitally signed (see section 5 of [SAMLCore]). Here we give an example of a `<ds:Signature>` element (with visual formatting, which invalidates the signature) used to sign the authentication assertion obtained via the Browser/POST profile in section 4.2:

```
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <ds:SignedInfo>
    <ds:CanonicalizationMethod
      Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
    <ds:SignatureMethod
      Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
    <ds:Reference URI="#c7055387-af61-4fce-8b98-e2927324b306">
      <ds:Transforms>
        <ds:Transform
          Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
signature"/>
        <ds:Transform
          Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
          <InclusiveNamespaces
            PrefixList="#default saml samlp ds xsd xsi"
            xmlns="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        </ds:Transform>
      </ds:Transforms>
      <ds:DigestMethod
        Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <ds:DigestValue>TCDVSuG6grhyHbzhQFWFzGrxIPE=</ds:DigestValue>
    </ds:Reference>
  </ds:SignedInfo>
  <ds:SignatureValue>
    x/GyPbzmFEe85pGD3c1aXG4Vspb9V9jGCjwcRCKrtwPS6vdVNCcY5rHaFPYWkf+5
    EIYcPzx+pX1h43SmwviCqXRjRtMANWbHLhWAptaK1ywS7gFgsD01qjyen3CP+m3D
    w6vKhaqledl0BYyrIzb4KkHO4ahNyBVXbJwqv5pUaE4=
  </ds:SignatureValue>
  <ds:KeyInfo>
    <ds:X509Data>
      <ds:X509Certificate>
```

```
        MIICyjCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgakxCzAJBgNVBAYTAlVT
        MRIwEAYDVQQIEwlXaXNjb25zaW4xEDAOBgNVBAcTB01hZGlzb24xIDAeBgNVBAoT
        F1VuaXZlcnNpdHkgb2YgV2lzY29uc2luMSswKQYDVQQLEyJEaXZpc2lvbiBvZiBJ
        bmZvcm1hdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgQ0Eg
        LS0gMjAwMjA3MDFBMB4XDTAyMDcyNjA3Mjc1MVoXDTA2MDkwNDA3Mjc1MVowgYsx
        CzAJBgNVBAYTAlVTMREwDwYDVQQIEwhNaWNoaWdhbjESMBAGA1UEBxMJQW5uIEFy
        Ym9yMQ4wDAYDVQQKEwVVQ0FJRDEcMBoGA1UEAxMTc2hpYjEuaW50ZXJuZXQyLmVk
        dTEnMCUGCSqGSIb3DQEJARYYcm9vdEBzaGliMS5pbnRlcm5ldDIuZWR1MIGfMA0G
        CSqGSIb3DQEBAQUAA4GNADCBiQKBgQDZSAb2sxvhAXnXVIVTx8vuRay+x50z7GJj
        IHRYQgIv6IqaGG04eTcyVMhoekE0b45QgvBIaOAPSZBl13R6+KYiE7x4XAWIrCP+
        c2MZVeXeTgV3Yz+USLg2Y1on+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7O27rhRjE
        pmqOIfGTWQIDAQABox0wGzAMBgNVHRMBAf8EAjAAMAsGA1UdDwQEAwIFoDANBgkq
        hkiG9w0BAQQFAAOBgQBfDqEW+OI3jqBQHIBzhujN/PizdN7s/z4D5d3pptWDJf2n
        qgi7lFV6MDkhmTvTqBtjmNk3No7v/dnP6Hr7wHxvCCRwubnmIfZ6QZAv2FU78pLX
        8I3bsbmRAUg4UP9hH6ABVq4KQKMknxu1xQxLhpR1ylGPdiowMNTrEG8cCx3w/w==
      </ds:X509Certificate>
    </ds:X509Data>
  </ds:KeyInfo>
</ds:Signature>
```

SAML metadata elements are also signed. The examples in section 6, for instance, implicitly include a `<ds:Signature>` element similar to the one above.

4 Shibboleth SSO Profiles

Shibboleth specifies two browser-based single sign-on (SSO) profiles:

1. Browser/POST Profile
2. Browser/Artifact Profile

These profiles extend the SAML 1.1 profiles of the same name [SAMLBind]. While the SAML 1.1 profiles begin with a request to the IdP, the Shibboleth profiles are service-provider-first and therefore more complex.

4.1 Authentication Request Profile

To accommodate SP-first profiles, Shibboleth introduces the notion of an *authentication request*. A Shibboleth authentication request is a URL-encoded message sent to an SSO service endpoint at the IdP. The following parameters are included in the query string:

- `providerId`

  The `providerId` parameter is the unique identifier (usually a URI) of the SP. The IdP may use the `providerId` to perform special handling or processing of the authentication request.

- `shire`

  The `shire` parameter (so named for historical reasons) is the location of the assertion consumer service endpoint at the SP. When using the Browser/POST profile, this URL becomes the value of the `<form>` element's `action` attribute. For the Browser/Artifact profile, the value of the `shire` parameter is a prefix of the redirect URL used by the SSO service.

- `target`

  The `target` parameter is a state maintenance parameter, possibly the location of the target resource. Regardless of the profile used, the `target` parameter must be preserved by the IdP and included in the response to the SP.

- `time`

  The (optional) `time` parameter is the current time in seconds past the epoch. It is used to assist the IdP in detecting stale requests from the client. It is important that the `time` parameter *not* be used as any kind of security measure.

Here is an example of a Shibboleth authentication request (without URL encoding for clarity):

```
https://idp.example.org/shibboleth/SSO?
  target=https://sp.example.org/myresource&
    shire=https://sp.example.org/shibboleth/SSO&
      providerId=https://sp.example.org/shibboleth&
        time=1102260120
```

Detailed handling of this request is outlined in the subsections below.

4.2 Browser/POST Profile

The *Shibboleth Browser/POST profile* is a combination of two profiles, the Shibboleth Authentication Request profile [ShibProt] and the SAML 1.1 Browser/POST profile [SAMLBind]. The message flow associated with the Shibboleth Browser/POST profile is depicted in Figure 3. As with both Shibboleth SSO profiles, the message flow begins with a request for a secured resource at the SP.

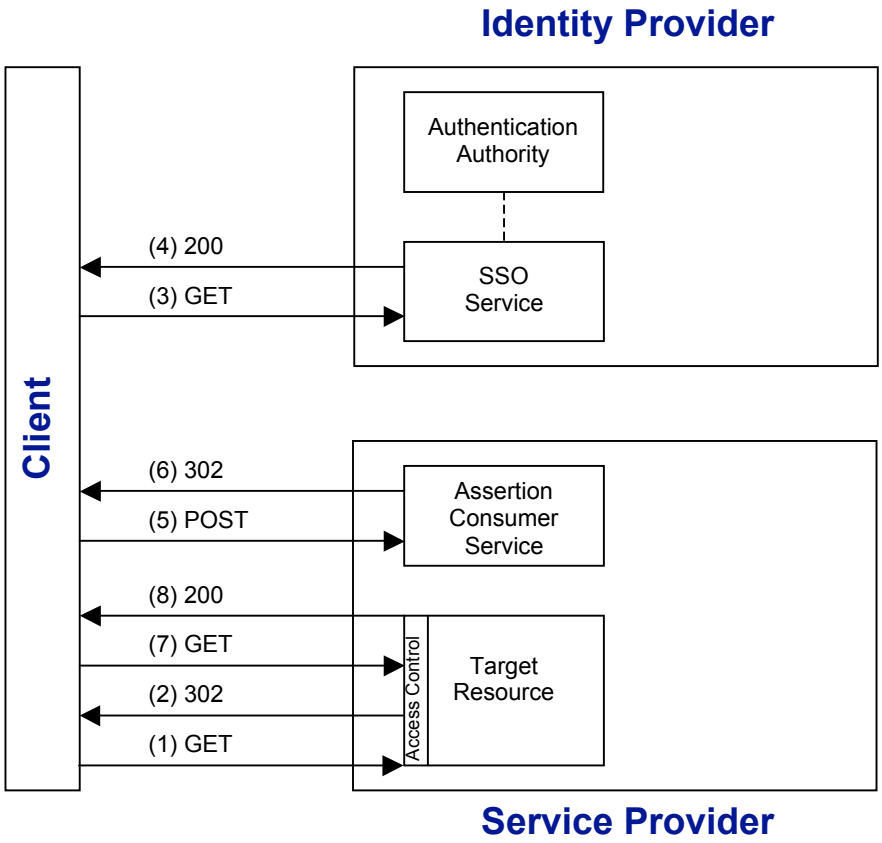

Figure 3: Shibboleth Browser/POST Profile

(1) The client requests a target resource at the SP:

```
https://sp.example.org/myresource
```

The SP performs a security check on behalf of the target resource. If a valid security context at the SP already exists, skip steps 2–7.

(2) The SP redirects the client to the single sign-on (SSO) service at the IdP. Three parameters are appended to the redirect URL (the optional `time` parameter is omitted in this example).

(3) The client requests the SSO service at the IdP:

```
https://idp.example.org/shibboleth/SSO?
  target=https://sp.example.org/myresource&
    shire=https://sp.example.org/shibboleth/SSO/POST&
      providerId=https://sp.example.org/shibboleth
```

The SSO service processes the authentication request and performs a security check. If the user does not have a valid security context, the IdP identifies the principal (details omitted). Once the principal has been identified, the SSO service obtains an authentication statement from the authentication authority.

(4) The SSO service responds with a document containing an HTML form:

```
<form method="post"
      action="https://sp.example.org/shibboleth/SSO/POST" ...>
  <input name="TARGET" type="hidden"
         value="https://sp.example.org/myresource" />
  <input name="SAMLResponse" value="response" type="hidden" />
```

```
  ...
  <input type="submit" value="Submit" />
</form>
```

The value of the `TARGET` parameter has been preserved from step 3. The value of the `SAMLResponse` parameter is the base64 encoding of a digitally signed `<samlp:Response>` element such as the one below:

```
<samlp:Response
  xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
  MajorVersion="1" MinorVersion="1"
  IssueInstant="2004-12-05T09:22:02Z"
  Recipient="https://sp.example.org/shibboleth/SSO/POST"
  ResponseID="c7055387-af61-4fce-8b98-e2927324b306">
  <!-- insert ds:Signature element here (section 3.4) -->
  <samlp:Status><samlp:StatusCode Value="samlp:Success"/></samlp:Status>
  <!-- insert SAML assertion here (section 3.1) -->
</samlp:Response>
```

(5) The client issues a POST request to the assertion consumer service at the SP. To automate the submission of the form, the following line of JavaScript (or its equivalent) may appear in the HTML document containing the form in step 4:

```
window.onload = function() { document.forms[0].submit(); }
```

This assumes that the page contains a single HTML `<form>` element.

(6) The assertion consumer service processes the authentication response, creates a security context at the SP and redirects the client to the target resource.

(7) The client requests the target resource at the SP (again):

```
https://sp.example.org/myresource
```

(8) Since a security context exists, the SP returns the resource to the client.

4.2.1 Browser/POST Profile with WAYF Service

In general, the SP does not know the user's preferred IdP at step 2 of the Browser/POST profile. The process whereby the SP determines the appropriate IdP is called *identity provider discovery* (generally, a very difficult problem). Shibboleth specifies an (optional) WAYF service to aid the SP in IdP discovery.

A WAYF ("Where are you from?") service is an intermediary between the SP and the IdP. The SP sends its authentication request to the WAYF, which somehow determines the user's preferred IdP (through unspecified means) and then subsequently redirects the client to the desired IdP. In the process, the WAYF preserves the values of all parameters except the `time` parameter, which is updated.

In practice, the first time the client accesses an SP, the WAYF might present the user with a form containing a list of all available IdPs. The user selects an IdP from the list and submits the form, after which the WAYF redirects the client to the selected IdP. At the same time, the client stores a reference to the IdP in a client-side cookie such that subsequent visits to the same SP are redirected straight through to the corresponding IdP by the WAYF.

As a hypothetical example, suppose that a WAYF service resides at domain example.org and that the SP redirects the client to this WAYF as follows:

```
https://wayf.example.org/?
  target=https://sp.example.org/myresource&
    shire=https://sp.example.org/shibboleth/SSO/POST&
      providerId=https://sp.example.org/shibboleth
```

In response, the WAYF returns a form to the client with the following elements:

```
<form method="get" action="https://wayf.example.org/" ...>
  <input name="target" type="hidden"
         value="https://sp.example.org/myresource" />
  <input name="shire" type="hidden"
         value="https://sp.example.org/shibboleth/SSO/POST" />
  <input name="providerId" type="hidden"
         value="https://sp.example.org/shibboleth" />
  ...
  <input type="submit" value="Submit" />
</form>
```

Presumably the form contains additional controls that allow the user to select an IdP, after which the form is submitted to the WAYF. The WAYF processes the form, sets the cookie and redirects the client to the selected IdP.

The message flow of the Shibboleth Browser/POST profile with WAYF service is outlined in Figure 4. (The flow of the corresponding Browser/Artifact profile is similar.) In step 4 of that flow, the WAYF returns an HTML form to the user who chooses an IdP from a list. After the form is submitted at step 5, the client is redirected to the SSO service of the chosen IdP where after the flow is identical to the ordinary Browser/POST profile.

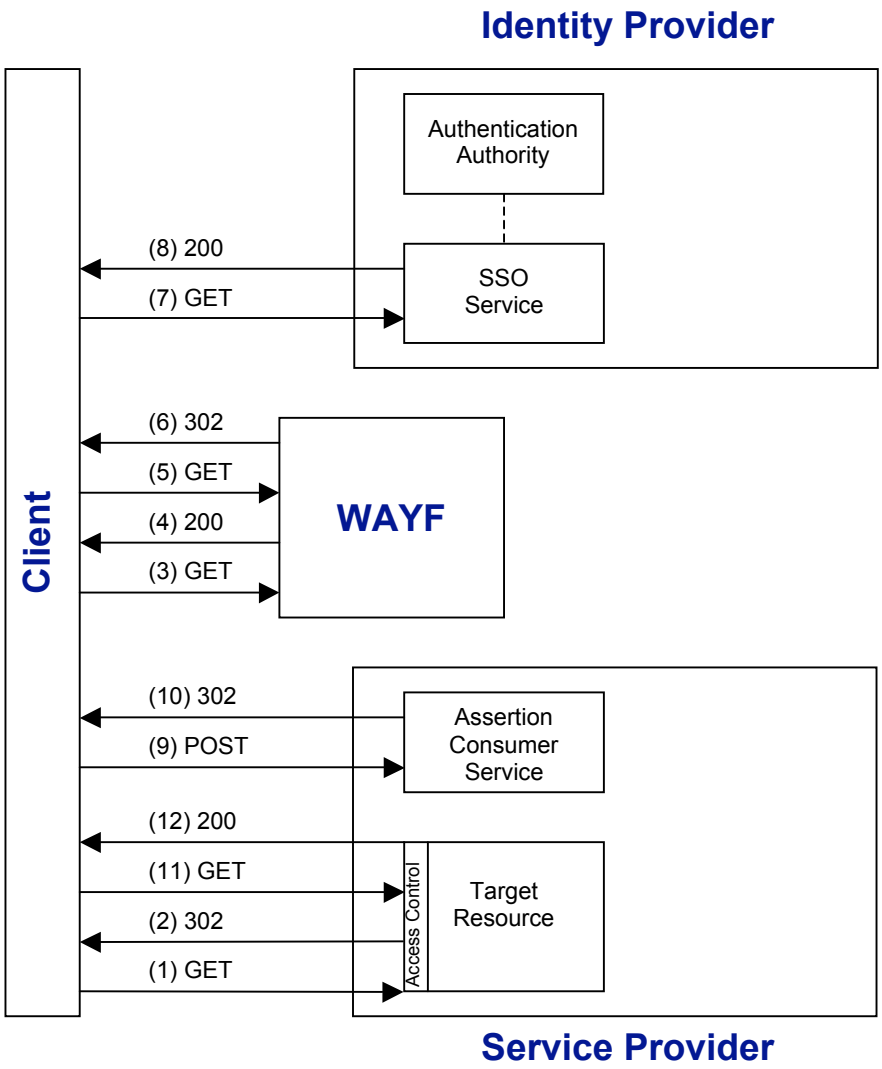

Figure 4: Shibboleth Browser/POST Profile with WAYF Service

(1) The client requests a target resource at the SP:

```
https://sp.example.org/myresource
```

The SP performs a security check on behalf of the target resource. If a valid security context at the SP already exists, skip steps 2–11.

(2) The SP redirects the client to the WAYF. Three parameters are appended to the redirect URL (the optional `time` parameter is omitted in this example).

(3) The client requests the WAYF service:

```
https://wayf.example.org/?
  target=https://sp.example.org/myresource&
    shire=https://sp.example.org/shibboleth/SSO/POST&
      providerId=https://sp.example.org/shibboleth
```

The WAYF service processes the authentication request and performs a cookie check. If the user has the required cookie, skip steps 4 and 5. Otherwise an HTML form is returned to the client.

(4) The WAYF returns an HTML form to the client. The parameters of the authentication request (shown in the previous step) are encoded as hidden fields in the form as illustrated above.

(5) The user selects an IdP from the list and submits the form, which causes the browser to issue a GET request to the WAYF. (Depending on the particular WAYF implementation, multiple HTTP transactions may occur at this step.)

(6) The WAYF updates the cookie with the user's preferred IdP and redirects the client to the SSO service. Three parameters are appended to the redirect URL.

(7) The client requests the SSO service at the IdP:

```
https://idp.example.org/shibboleth/SSO?
  target=https://sp.example.org/myresource&
    shire=https://sp.example.org/shibboleth/SSO/POST&
      providerId=https://sp.example.org/shibboleth
```

The SSO service processes the authentication request and performs a security check. If the user does not have a valid security context, the IdP identifies the principal (details omitted). Once the principal has been identified, the SSO service obtains an authentication statement from the authentication authority.

(8) The SSO service responds with an HTML form as in step 4 of the ordinary Browser/POST profile.

(9) The client issues a POST request to the assertion consumer service at the service provider as in step 5 of the ordinary Browser/POST profile.

(10) The assertion consumer service processes the authentication response, creates a security context at the SP and redirects the client to the target resource.

(11) The client requests the target resource at the SP (again):

```
https://sp.example.org/myresource
```

(12) Since a security context exists, the SP returns the resource to the client.

4.3 Browser/Artifact Profile

The *Shibboleth Browser/Artifact profile* is a combination of the Shibboleth Authentication Request [ShibProt] profile and the SAML 1.1 Browser/Artifact profile [SAMLBind]. The message flow associated with the Shibboleth Browser/Artifact profile is shown in Figure 5.

Note the back-channel exchange of the SAML artifact in steps 6 and 7. This is the distinguishing characteristic between the Browser/Artifact and Browser/POST profiles.

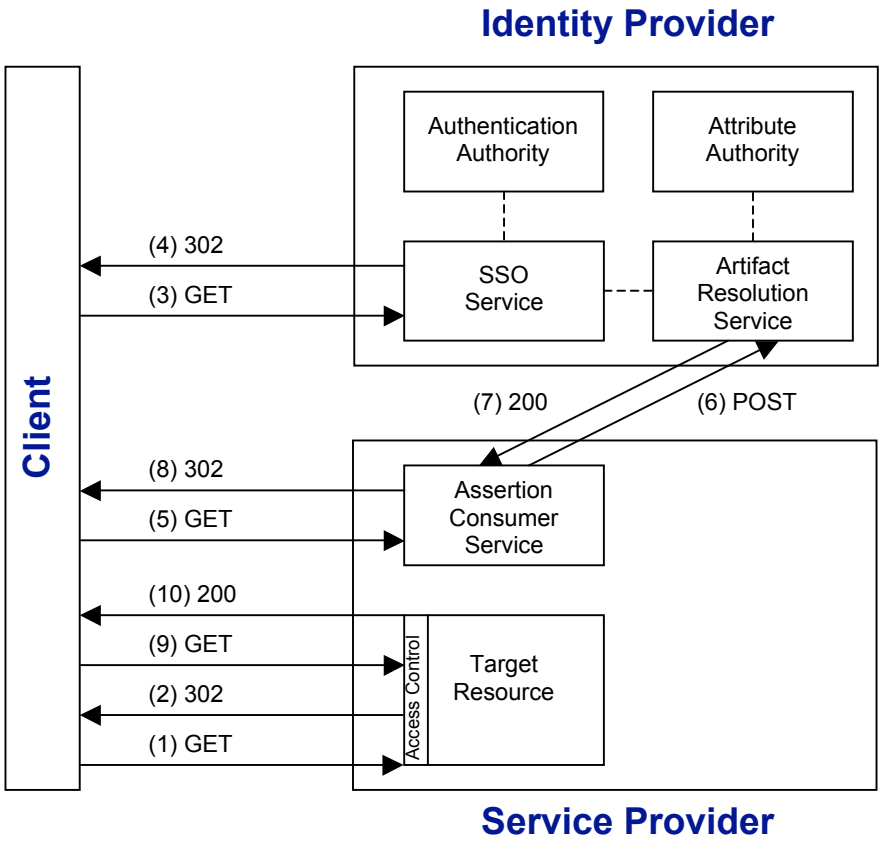

Figure 5: Shibboleth Browser/Artifact Profile

(1) The client requests a target resource at the SP:

```
https://sp.example.org/myresource
```

The SP performs a security check on behalf of the target resource. If a valid security context at the SP already exists, skip steps 2–9.

(2) The SP redirects the client to the single sign-on (SSO) service at the IdP. Three parameters are appended to the redirect URL (the optional `time` parameter is omitted in this example).

(3) The client requests the SSO service at the IdP:

```
https://idp.example.org/shibboleth/SSO?
  target=https://sp.example.org/myresource&
    shire=https://sp.example.org/shibboleth/SSO/Artifact&
      providerId=https://sp.example.org/shibboleth
```

The SSO service processes the authentication request and performs a security check. If the user does not have a valid security context, the IdP identifies the principal (details omitted). Once the principal has been identified, the SSO service obtains an authentication statement from the authentication authority.

(4) The SSO service redirects the client to the assertion consumer service at the SP. A `TARGET` parameter and a `SAMLart` parameter are appended to the redirect URL prefix given by the `shire` parameter in step 3.

(5) The client requests the assertion consumer service at the SP:

```
https://sp.example.org/shibboleth/SSO/Artifact?
  TARGET=https://sp.example.org/myresource&
    SAMLart=AAEwGDwd3Z7Fr1GPbM82Fk2CZbpNB1dxD%2Bt2Prp%2BTDtqxVA78iMf3F23
```

where the value of the `TARGET` parameter is preserved from step 3 and the value of the `SAMLart` parameter is a SAML artifact of type 0x0001 defined in [SAMLBind].

(6) The assertion consumer service validates the request and dereferences the artifact by sending a SAML `Request` to the artifact resolution service at the IdP:

```
POST /shibboleth/Artifact HTTP/1.1
Host: idp.example.org
Content-Type: text/xml
Content-Length: nnn
SOAPAction: http://www.oasis-open.org/committees/security

<?xml version="1.1" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <samlp:Request
      xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
      MajorVersion="1" MinorVersion="1"
      RequestID="f81d4fae-7dec-11d0-a765-00a0c91e6bf6"
      IssueInstant="2004-12-05T09:22:04Z">
      <samlp:AssertionArtifact>
        AAEwGDwd3Z7Fr1GPbM82Fk2CZbpNB1dxD+t2Prp+TDtqxVA78iMf3F23
      </samlp:AssertionArtifact>
    </samlp:Request>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

where the value of the `<samlp:AssertionArtifact>` element is the SAML artifact at step 5.

(7) The artifact resolution service at the IdP returns a `<samlp:Response>` element (containing an authentication statement) to the assertion consumer service at the SP:

```
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: nnnn

<?xml version="1.1" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <samlp:Response
      xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
      MajorVersion="1" MinorVersion="1"
      Recipient="https://sp.example.org/shibboleth/SSO/Artifact"
      ResponseID="00099cf1-a355-10f9-9e95-004005b13a2b"
      InResponseTo="f81d4fae-7dec-11d0-a765-00a0c91e6bf6"
      IssueInstant="2004-12-05T09:22:05Z">
      <samlp:Status>
        <samlp:StatusCode Value="samlp:Success"/>
      </samlp:Status>
```

```
      <!-- insert SAML assertion here (see section 3.1) -->
    </samlp:Response>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

(8) The assertion consumer service processes the authentication response, creates a security context at the SP and redirects the client to the target resource.

(9) The client requests the target resource at the SP (again):

```
https://sp.example.org/myresource
```

(10) Since a security context exists, the SP returns the resource to the client.

4.3.1 Browser/Artifact Profile with WAYF Service

Like the Browser/POST profile, an SP that supports the Browser/Artifact profile may also utilize the services of a WAYF. The resulting Browser/Artifact profile with WAYF service is analogous to the Browser/POST case (section 4.2.1).

5 Attributes

Shibboleth specifies the use of a standard SAML *attribute request protocol* to facilitate attribute sharing among IdPs and SPs. In this section we augment the SSO profiles of sections 4.2 and 4.3 with an attribute exchange that permits the SP to make an informed access control decision. Such an attribute exchange is optional since the SP may choose to provide a security context based solely on an authentication assertion. In practice, however, the SP may need to know more about the authenticated user before access to a resource may be granted.

5.1 Browser/POST Profile with Attribute Exchange

Although attributes may be embedded in the authentication assertion transferred from IdP to SP, Shibboleth specifies an optional back-channel exchange using a SAML protocol binding such as the SOAP 1.1 binding. The attribute exchange is a simple two-step process initiated towards the end of the ordinary Browser/POST profile (see Figure 6).

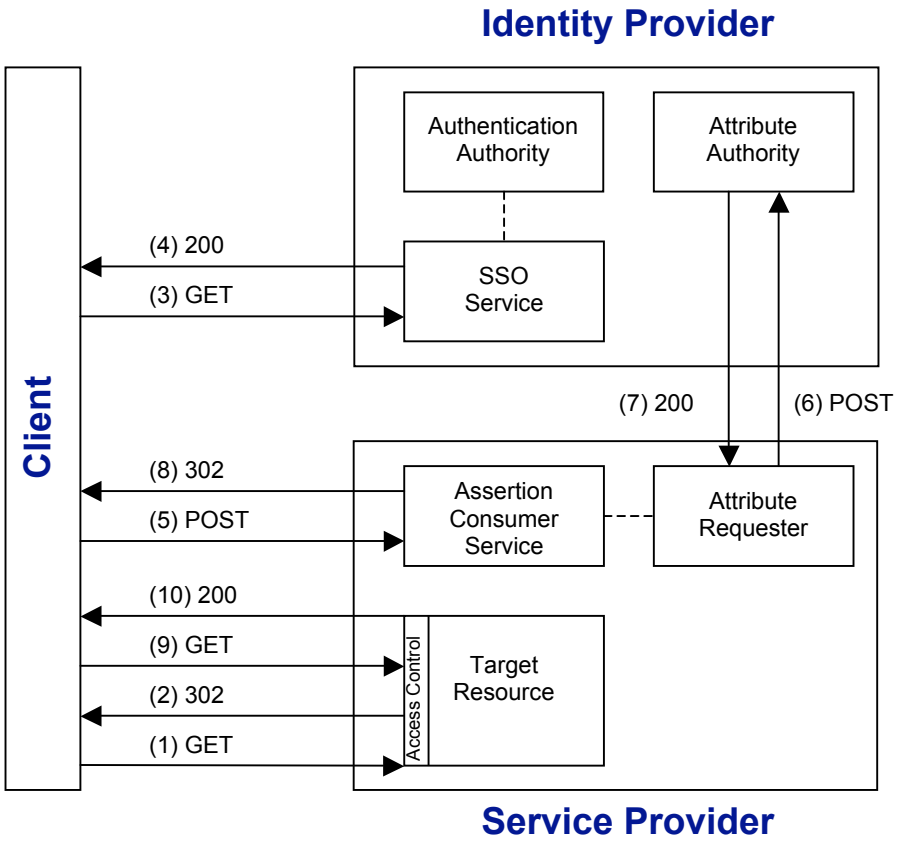

Figure 6: Shibboleth Browser/POST Profile with Attribute Exchange

The first five steps of the ordinary Browser/POST profile proceed as before. Before access to the resource is granted, however, an attribute exchange is required.

(1–5) Same as the ordinary Browser/POST profile.

(6) The assertion consumer service parses the POST request, validates the signature on the `<samlp:Response>` element, creates a security context at the SP and passes control to the

attribute requester to initiate the attribute exchange. The attribute requester POSTs a SAML SOAP message to the attribute authority (AA) at the IdP:

```
POST /shibboleth/AA/SOAP HTTP/1.1
Host: idp.example.org
Content-Type: text/xml
Content-Length: nnn
SOAPAction: http://www.oasis-open.org/committees/security

<?xml version="1.1" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <samlp:Request
      xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
      xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
      MajorVersion="1" MinorVersion="1"
      IssueInstant="2004-12-05T09:22:04Z"
      RequestID="aaf23196-1773-2113-474a-fe114412ab72">
      <samlp:AttributeQuery
        Resource="https://sp.example.org/shibboleth">
        <saml:Subject>
          <saml:NameIdentifier
            Format="urn:mace:shibboleth:1.0:nameIdentifier"
            NameQualifier="https://idp.example.org/shibboleth">
            3f7b3dcf-1674-4ecd-92c8-1544f346baf8
          </saml:NameIdentifier>
        </saml:Subject>
        <saml:AttributeDesignator
          AttributeName="urn:mace:dir:attribute-def:eduPersonPrincipalName"
          AttributeNamespace="urn:mace:shibboleth:1.0:attributeNamespace:uri"/>
      </samlp:AttributeQuery>
    </samlp:Request>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

(7) The AA at the IdP processes the request and returns the required attributes to the attribute requester:

```
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: nnnn

<?xml version="1.1" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <samlp:Response
      xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
      InResponseTo="aaf23196-1773-2113-474a-fe114412ab72"
      IssueInstant="2004-12-05T09:22:05Z"
      MajorVersion="1" MinorVersion="1"
      ResponseID="b07b804c-7c29-ea16-7300-4f3d6f7928ac">
      <samlp:Status>
        <samlp:StatusCode Value="samlp:Success"/>
      </samlp:Status>
      <!-- insert SAML assertion from section 3.2 here -->
```

```
    </samlp:Response>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

(8) The assertion consumer service filters the attributes, updates the security context and redirects the client to the target resource.

(9) The client requests the target resource at the SP (again):

```
https://sp.example.org/myresource
```

(10) Since a security context exists, the SP returns the resource to the client.

5.2 Browser/Artifact Profile with Attribute Exchange

Recall that the Browser/Artifact profile of section 4.3 incorporates its own back-channel exchange to dereference the artifact. To obtain attributes, an additional back-channel exchange might be used (as in section 5.1). Alternatively, attributes could be retrieved at the same time the artifact is dereferenced, which we illustrate in this section. The resulting flow is identical to the previous flow associated with the Browser/Artifact profile (see Figure 5).

As observed in section 3, an attribute statement may be combined with an authentication statement in one of two ways: both statements may be wrapped in a single assertion or separate assertions may contain individual statements. The latter approach is illustrated here.

Two assertions require two artifacts, which is the primary difference between this example and the example of section 4.3. Steps 4–7 of the latter example are modified for two artifacts as follows:

(1–3) Same as the ordinary Browser/Artifact profile.

(4) The SSO service redirects the client to the assertion consumer service at the SP. A `TARGET` parameter and *two* `SAMLart` parameters are appended to the redirect URL.

(5) The client requests the assertion consumer service at the SP:

```
https://sp.example.org/shibboleth/SSO/Artifact?
  TARGET=https://sp.example.org/myresource&
    SAMLart=AAEwGDwd3Z7Fr1GPbM82Fk2CZbpNB7YuJ8gk%2BvmCjh9Y4Wsq6H5%2BKU4C&
      SAMLart=AAEwGDwd3Z7Fr1GPbM82Fk2CZbpNB8tb%2By6YOO40XGfl4QfLKq9xZ8UW
```

where one of the artifacts references an authentication assertion while the other artifact corresponds to an attribute assertion (order does not matter).

(6) The assertion consumer service validates the request and dereferences both artifacts by sending a single SAML `Request` to the artifact resolution service at the IdP:

```
POST /shibboleth/Artifact HTTP/1.1
Host: idp.example.org
Content-Type: text/xml
Content-Length: nnn
SOAPAction: http://www.oasis-open.org/committees/security

<?xml version="1.1" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <samlp:Request
      xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
      MajorVersion="1" MinorVersion="1"
      RequestID="f81d4fae-7dec-11d0-a765-00a0c91e6bf6"
      IssueInstant="2004-12-05T09:22:04Z">
```

```
      <samlp:AssertionArtifact>
        AAEwGDwd3Z7Fr1GPbM82Fk2CZbpNB7YuJ8gk+vmCjh9Y4Wsq6H5+KU4C
      </samlp:AssertionArtifact>
      <samlp:AssertionArtifact>
        AAEwGDwd3Z7Fr1GPbM82Fk2CZbpNB8tb+y6YOO40XGfl4QfLKq9xZ8UW
      </samlp:AssertionArtifact>
    </samlp:Request>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

where the values of the `<samlp:AssertionArtifact>` elements are the SAML artifacts at step 5.

(7) The artifact resolution service at the IdP returns a `<samlp:Response>` element (containing two assertions) to the assertion consumer service at the SP:

```
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: nnnn

<?xml version="1.1" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <samlp:Response
      xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
      MajorVersion="1" MinorVersion="1"
      Recipient="https://sp.example.org/shibboleth/SSO/Artifact"
      ResponseID="00099cf1-a355-10f9-9e95-004005b13a2b"
      InResponseTo="f81d4fae-7dec-11d0-a765-00a0c91e6bf6"
      IssueInstant="2004-12-05T09:22:05Z">
      <samlp:Status>
        <samlp:StatusCode Value="samlp:Success"/>
      </samlp:Status>
      <!-- insert SAML assertions from section 3.3 here -->
    </samlp:Response>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

(8–10) Same as the ordinary Browser/Artifact profile.

5.3 Directory Schema

Shibboleth, by virtue of SAML, has a well-defined attribute exchange mechanism, but neither specification defines any attributes per se. Instead it is left to individual implementations to define their own attributes. In a cross-domain federated environment, in particular, a standard approach to user attributes is crucial. In the absence of such standards, federation on a large scale will be difficult if not impossible.

To address this issue within the higher education community, Internet2 and EDUCAUSE have jointly developed a set of attributes and associated bindings called *eduPerson* [eduPerson]. The LDAP binding of *eduPerson* is derived from the standard LDAP object class called `inetOrgPerson` [RFC 2798] which itself is based on other standard LDAP classes (see e.g., [RFC 2256]).

Of all the attributes defined by this hierarchy of object classes, approximately 40 attributes have been defined by InCommon as *common identity attributes* [InCommonAttribs]:

Common Attributes	
Highly recommended	6
Suggested	10
Optional	25
	41

For example, here are InCommon's six "highly recommended" attributes:

Attribute Name	Attribute Value
`givenName`	Mary
`sn` (surname)	Smith
`cn` (common name)	Mary Smith
`eduPersonScopedAffiliation`	student@idp.example.org
`eduPersonPrincipalName`	mary.smith@idp.example.org
`eduPersonTargetedID`	—

The attribute `eduPersonTargetedID` does not have a precise value syntax. See the *EduPerson Specification* [eduPersonSpec] for more information about `eduPersonTargetedID`.

6 Metadata

Both IdPs and SPs publish information about themselves in special XML files called *metadata* files. Since metadata precisely identify the provider and the services it offers, metadata files are digitally signed for integrity protection.

A SAML metadata specification was first published as part of SAML 2.0 [SAML2Meta]. Since that time, the OASIS Security Services Technical Committee has considered adopting a minimal subset of SAML 2.0 metadata for use with SAML 1.x as specified in [SAMLMeta]. Shibboleth further constrains this profile in [ShibProt].

Only four metadata descriptors are defined for use within Shibboleth:

1. `<md:IDPSSODescriptor>`
2. `<md:SPSSODescriptor>`
3. `<md:AuthnAuthorityDescriptor>`
4. `<md:AttributeAuthorityDescriptor>`

These metadata elements correspond to the SSO service (IdP), the assertion consumer service (SP), the authentication authority (IdP), and the attribute authority (IdP), respectively. We will omit the `<md:AuthnAuthorityDescriptor>` element in what follows since the authentication authority operates behind the scenes in a typical Shibboleth deployment, in which case it need not publish metadata about itself.

Multiple `<md:EntityDescriptor>` elements may be grouped together in a single `<md:EntitiesDescriptor>` element:

```
<md:EntitiesDescriptor
  xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:oasis:names:tc:SAML:2.0:metadata ..."
  Name="urn:mace:shibboleth:examples"
  validUntil="2010-01-01T00:00:00Z">
  <!-- insert ds:Signature element (section 3.4) -->
  <!-- insert md:EntityDescriptor elements here -->
</md:EntitiesDescriptor>
```

Note that an `<md:EntitiesDescriptor>` element may be signed (see section 3.4). Alternatively, the individual `<md:EntityDescriptor>` elements may be signed (see the examples in the following subsections).

6.1 Identity Provider Metadata

An identity provider publishes data about itself in an `<md:EntityDescriptor>` element:

```
<md:EntityDescriptor
  xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  entityID="https://idp.example.org/shibboleth">
  <!-- insert ds:Signature element (section 3.4) -->
  <!-- insert md:IDPSSODescriptor element (section 6.1.1) -->
  <!-- insert md:AttributeAuthorityDescriptor element (section 6.1.2) -->
  <md:Organization>
    <md:OrganizationName xml:lang="en">
      Shibboleth Identity Provider
```

```
    </md:OrganizationName>
    <md:OrganizationDisplayName xml:lang="en">
      Shibboleth Identity Provider @ Some Location
    </md:OrganizationDisplayName>
    <md:OrganizationURL xml:lang="en">
      http://www.idp.example.org/
    </md:OrganizationURL>
  </md:Organization>
  <md:ContactPerson contactType="technical">
    <md:SurName>Shibboleth IdP Support</md:SurName>
    <md:EmailAddress>shib-support@idp.example.org</md:EmailAddress>
  </md:ContactPerson>
</md:EntityDescriptor>
```

The `entityID` attribute is the unique providerId of the IdP. Note that the details of the digital signature (in the `<ds:Signature>` element) have been omitted from this example. See section 3.4 for relevant discussion.

The IdP manages an SSO service and an attribute authority, each having its own descriptor. These are detailed in the following subsections.

6.1.1 SSO Service Metadata

The SSO service at the IdP is encapsulated in an `<md:IDPSSODescriptor>` element:

```
<md:IDPSSODescriptor
  protocolSupportEnumeration="urn:oasis:names:tc:SAML:1.1:protocol
urn:mace:shibboleth:1.0">
  <md:KeyDescriptor use="signing">
    <ds:KeyInfo>
      <ds:KeyName>IdP SSO Key</ds:KeyName>
    </ds:KeyInfo>
  </md:KeyDescriptor>
  <md:ArtifactResolutionService isDefault="true" index="0"
    Binding="urn:oasis:names:tc:SAML:1.0:bindings:SOAP-binding"
    Location="https://idp.example.org/shibboleth/Artifact"/>
  <md:NameIDFormat>
    urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress
  </md:NameIDFormat>
  <md:NameIDFormat>
    urn:mace:shibboleth:1.0:nameIdentifier
  </md:NameIDFormat>
  <md:SingleSignOnService
    Binding="urn:mace:shibboleth:1.0:profiles:AuthnRequest"
    Location="https://idp.example.org/shibboleth/SSO"/>
</md:IDPSSODescriptor>
```

The previous metadata element describes the SSO service at the IdP of section 4. Note the following details about this element:

- Since Shibboleth supports an SSO service (whereas SAML1 does not), protocol support includes the URI

  `urn:mace:shibboleth:1.0`

  which is a unique identifier specified by [ShibProt].
- Key information has been omitted for brevity.
- The `Binding` attribute of the `<md:ArtifactResolutionService>` element indicates that the SAML SOAP binding should be used (see [SAMLBind]).

- The `Location` attribute of the `<md:ArtifactResolutionService>` element is used in step 6 of the Browser/Artifact profile of section 4.3.
- The `index` attribute of the `<md:ArtifactResolutionService>` element is ignored.
- The `<md:NameIDFormat>` elements indicate what SAML *NameIdentifier formats* [SAMLCore] the SSO service supports. One of these is a standard Shibboleth *handle* indicated by URI

  `urn:mace:shibboleth:1.0:nameIdentifier`
- The `Binding` attribute of the `<md:SingleSignOnService>` element is a unique identifier specified by [ShibProt].
- The `Location` attribute of the `<md:SingleSignOnService>` element is used by SPs and WAYFs to redirect the client to the SSO service at the IdP.

6.1.2 Attribute Authority Metadata

The attribute authority at the IdP is given by an `<md:AttributeAuthorityDescriptor>` element:

```
<md:AttributeAuthorityDescriptor
  protocolSupportEnumeration="urn:oasis:names:tc:SAML:1.1:protocol">
  <md:KeyDescriptor use="signing">
    <ds:KeyInfo>
      <ds:KeyName>IdP AA Key</ds:KeyName>
    </ds:KeyInfo>
  </md:KeyDescriptor>
  <md:AttributeService
    Binding="urn:oasis:names:tc:SAML:1.0:bindings:SOAP-binding"
    Location="https://idp.example.org/shibboleth/AA/SOAP"/>
  <saml:Attribute
    Name="urn:mace:dir:attribute-def:eduPersonPrincipalName"
    NameFormat="urn:mace:shibboleth:1.0:attributeNamespace:uri"/>
  <saml:Attribute
    Name="urn:mace:dir:attribute-def:eduPersonScopedAffiliation"
    NameFormat="urn:mace:shibboleth:1.0:attributeNamespace:uri">
    <saml:AttributeValue>member</saml:AttributeValue>
    <saml:AttributeValue>student</saml:AttributeValue>
    <saml:AttributeValue>faculty</saml:AttributeValue>
    <saml:AttributeValue>employee</saml:AttributeValue>
    <saml:AttributeValue>staff</saml:AttributeValue>
  </saml:Attribute>
  <saml:Attribute
    Name="urn:mace:dir:attribute-def:eduPersonEntitlement"
    NameFormat="urn:mace:shibboleth:1.0:attributeNamespace:uri"/>
  <md:NameIDFormat>
    urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress
  </md:NameIDFormat>
  <md:NameIDFormat>
    urn:mace:shibboleth:1.0:nameIdentifier
  </md:NameIDFormat>
</md:AttributeAuthorityDescriptor>
```

This element is not needed for the SSO profiles of section 4, but it is required for the attribute exchanges outlined in sections 5.1 and 5.2.

6.2 Service Provider Metadata

A service provider also publishes data about itself in an `<md:EntityDescriptor>` element:

```
<md:EntityDescriptor
  xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  entityID="https://sp.example.org/shibboleth">
  <!-- insert ds:Signature element (section 3.4) -->
  <!-- insert md:SPSSODescriptor element (section 6.2.1) -->
  <md:Organization>
    <md:OrganizationName xml:lang="en">
      Shibboleth Service Provider
    </md:OrganizationName>
    <md:OrganizationDisplayName xml:lang="en">
      Shibboleth Service Provider @ Some Location
    </md:OrganizationDisplayName>
    <md:OrganizationURL xml:lang="en">
      http://www.sp.example.org/
    </md:OrganizationURL>
  </md:Organization>
  <md:ContactPerson contactType="technical">
    <md:SurName>Shibboleth SP Support</md:SurName>
    <md:EmailAddress>shib-support@sp.example.org</md:EmailAddress>
  </md:ContactPerson>
</md:EntityDescriptor>
```

The primary component managed by the SP is the assertion consumer service, which is discussed below.

6.2.1 Assertion Consumer Service Metadata

The assertion consumer service is represented by an `<md:SPSSODescriptor>` element:

```
<md:SPSSODescriptor
  protocolSupportEnumeration="urn:oasis:names:tc:SAML:1.1:protocol">
  <md:KeyDescriptor use="signing">
    <ds:KeyInfo>
      <ds:KeyName>SP SSO Key</ds:KeyName>
    </ds:KeyInfo>
  </md:KeyDescriptor>
  <md:NameIDFormat>
    urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress
  </md:NameIDFormat>
  <md:NameIDFormat>
    urn:mace:shibboleth:1.0:nameIdentifier
  </md:NameIDFormat>
  <md:AssertionConsumerService isDefault="true" index="0"
    Binding="urn:oasis:names:tc:SAML:1.0:profiles:browser-post"
    Location="https://sp.example.org/shibboleth/SSO/POST"/>
  <md:AssertionConsumerService index="1"
    Binding="urn:oasis:names:tc:SAML:1.0:profiles:artifact-01"
    Location="https://sp.example.org/shibboleth/SSO/Artifact"/>
  <md:AttributeConsumingService isDefault="true" index="0">
    <md:ServiceName xml:lang="en">
```

```
      Service Provider Portal
    </md:ServiceName>
    <md:RequestedAttribute
      Name="urn:mace:dir:attribute-def:eduPersonScopedAffiliation"
      NameFormat="urn:mace:shibboleth:1.0:attributeNamespace:uri">
    </md:RequestedAttribute>
    <md:RequestedAttribute
      Name="urn:mace:dir:attribute-def:eduPersonEntitlement"
      NameFormat="urn:mace:shibboleth:1.0:attributeNamespace:uri">
      <saml:AttributeValue xsi:type="xsd:anyURI">
        https://sp.example.org/entitlements/123456789
      </saml:AttributeValue>
    </md:RequestedAttribute>
  </md:AttributeConsumingService>
</md:SPSSODescriptor>
```

Note the following details about the `<md:SPSSODescriptor>` metadata element:

- Key information has been omitted for brevity.
- The `Binding` attributes of the `<md:AssertionConsumerService>` elements are standard URIs specified by [SAMLMeta].
- The `Location` attribute of the first `<md:AssertionConsumerService>` element (`index="0"`) is used in step 3 of the Browser/POST profile in section 4.2.
- The `Location` attribute of the second `<md:AssertionConsumerService>` element (`index="1"`) is used in step 3 of the Browser/Artifact profile in section 4.3.
- The `<md:AttributeConsumingService>` element is used by the IdP to formulate an attribute assertion that is *pushed* to the SP in step 7 of the Browser/Artifact profile outlined in section 5.2.
- The `index` attributes of the `<md:AssertionConsumerService>` and `<md:AttributeConsumingService>` elements are ignored.

As noted above, the values of the `Location` attribute are the same as the values of the `shire` parameter in the corresponding profile. Upon receiving the authentication request, the IdP checks the value of the `shire` parameter against the locations in SP metadata, which minimizes the possibility of a rogue SP orchestrating a man-in-the-middle attack.

7 References

7.1 Normative References

[eduPersonSpec] *EduPerson Specification* (200312), http://www.nmi-edit.org/eduPerson/internet2-mace-dir-eduperson-200312.html

[RFC 2256] M. Wahl, *A Summary of the X.500(96) User Schema for use with LDAPv3*, December 1997. http://www.ietf.org/rfc/rfc2256.txt

[RFC 2798] M. Smith, *Definition of the inetOrgPerson LDAP Object Class*, April 2000. http://www.ietf.org/rfc/rfc2798.txt

[SAML2Meta] S. Cantor et al., *Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, 15 March 2005. Document ID saml-metadata-2.0-os http://www.oasis-open.org/committees/security/

[SAMLBind] E. Maler et al., *Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-bindings-profiles-1.1 http://www.oasis-open.org/committees/security/

[SAMLCore] E. Maler et al., Assertions *and Protocols for the OASIS Security Assertion Markup Language (SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-core-1.1 http://www.oasis-open.org/committees/security/

[SAMLMeta] G. Whitehead and S. Cantor, *SAML 1.x Metadata Profile*. OASIS SSTC, February 2005. Document ID draft-saml1x-metadata-04 http://www.oasis-open.org/committees/security/

[ShibProt] S. Cantor et al., *Shibboleth Architecture: Protocols and Profiles*. Internet2-MACE, February 2005. Document ID draft-mace-shibboleth-arch-protocols-09 http://shibboleth.internet2.edu/

[SOAP] D. Box et al., *Simple Object Access Protocol (SOAP) 1.1*. W3C Note 08 May 2000. http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

7.2 Non-Normative References

[eduPerson] *eduPerson Object Class*. http://www.educause.edu/eduperson/

[HTTP] *HTTP - Hypertext Transfer Protocol*. World Wide Web Consortium (W3C). http://www.w3.org/Protocols/

[InCommonAttribs] *InCommon Federation: Common Identity Attributes*. http://www.incommonfederation.org/docs/policies/federatedattributes.html

[GSIsecurity] K. Bhatia et al., *Engineering an End-to-End GSI-based Security Infrastructure*. http://users.sdsc.edu/~chandras/Papers/ccgrid-submission.pdf

[SAMLTech] J. Hughes et al. Technical *Overview of the OASIS Security Assertion Markup Language (SAML) V1.1*. OASIS, May 2004. Document ID sstc-saml-tech-overview-1.1-cd http://www.oasis-open.org/committees/security/

[XML] *XML Core Working Group Public Page*. World Wide Web Consortium (W3C). http://www.w3.org/XML/Core/

[XMLSchema] *XML Schema*. World Wide Web Consortium (W3C). http://www.w3.org/XML/Schema

[XMLSig] *XML Signature WG*. World Wide Web Consortium (W3C). http://www.w3.org/Signature/