Google Search Appliance

Authentication/Authorization for Enterprise SPI Guide

Google Search Appliance software version 7.2 and later

Google

Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043
www.google.com

GSA-AUTH_200.01
March 2015

Contents

Authentication/Authorization for Enterprise SPI Guide

The SAML Authentication and Authorization Service Provider Interfaces (SPIs) enable a Google Search Appliance to communicate with an existing access control infrastructure via standard Security Assertion Markup Language (SAML) messages. The Authentication and Authorization SPIs are also required to support Windows Integrated Authentication with the Google SAML Bridge for Windows.

This document describes how to set up the Identity Provider and Policy Decision Point web services that are required by the Authentication and Authorization SPIs.

For more information on search appliance configuration for use with these SPIs, refer to "Configuring Crawl for the SAML Authentication and Authorization Service Provider Interface" in *Managing Search for Controlled-Access Content*.

Authentication (AuthN) is used to identify users, and authorization (AuthZ) is used to allow users access to documents according to their credentials.

The Authorization SPI (see "Authorization" on page 17) requires web services from a Policy Decision Point and an authentication method. The Authorization SPI can be used with any one of the following authentication methods:

- The SAML Authentication SPI (see "Authentication" on page 6), which requires web services from an Identity Provider
- LDAP directory service integration, including ActiveDirectory
- x.509 Certificates for user authentication
- Forms-based SSO with the cookie-cracker (see "Using Cookie Cracking" in *Managing Search for Controlled-Access Content*).

Note: Authentication through LDAP integration or x.509 certificates is configured on the search appliance. For more information on these authentication methods, refer to *Managing Search for Controlled-Access Content*.

Before You Begin

To write an Identity Provider and Policy Decision Point web service, you should be familiar with these technologies.

- **XML**: Extensible Markup Language. [http://www.w3.org/XML/]
- **SAML 2.0:** An XML-based standard whose primary use case is inter-domain single sign-on. [http://www.oasis-open.org/specs/#samlv2.0]
- **SOAP 1.1**: The Simple Object Access Protocol is an XML-based protocol for exchanging information over the Internet. [http://www.w3.org/TR/2003/REC-soap12-part1-20030624/]
- **GSA Universal Login with SPI**: The GSA's security-manager providing universal login forms/SPI. ["The SAML Authentication Service Provider Interface (SPI)" in *Managing Search for Controlled-Access Content*]
- **XML Digital Signatures**: Used for integrity protection of SAML Assertions. [http://www.w3.org/TR/xmldsig-core/]

Tip: One way to implement an Identity Provider and Policy Decision Point is to access a SOAP server using Apache Axis (see http://ws.apache.org/axis/) or by extending the authn.py (see https://code.google.com/p/gsa-admin-toolkit/source/browse/trunk/authn.py) GSA admin toolkit sample.

Security Manager

The Google Search Appliance's security manager handles user authentication processing on behalf of the search appliance. Its job is to provide the search appliance with a verified ID of the user performing the secure search and essentially broker credential management across various authentication mechanisms. The security manager is integrated into the search appliance software and runs inside the search appliance itself.

When writing an SPI, you only need to integrate with the protocol messages between the security-manager and your IdP/PDP server. The internal protocol between the search appliance and the security-manager, although visible to the user, is not something an administrator should account for. That is, an SPI provider would have to integrate with only a portion of the communication shown in this document and can essentially ignore the search appliance to security-manager communication (in Figure 1 below, an SPI provider integrates with steps **[2]**,**[3]**,**[4]**, and **[5]**).

Authentication

Purpose of the Google Search Authentication SPI

When implemented, the Authentication SPI allows search users to authenticate to the search appliance. It is designed to allow customers to integrate the search appliance into an existing access control infrastructure. Instead of authenticating search users itself, the search appliance redirects the user to an Identity Provider (IdP), a customer-implemented server, where the actual authentication takes place. The IdP then redirects or posts the user back to the search appliance, while passing information that includes the identity of the search user. The protocol that governs this communication between the search appliance, the browser, and the customer's IdP is SAML 2.0, an XML-based standard. Specifically, the two SAML bindings (how SAML protocol messages are communicated) an IdP could employ to return a response are *HTTP Artifact* and *HTTP POST*. Regardless of the binding used for the response, the *HTTP Redirect* binding is used for sending the request.

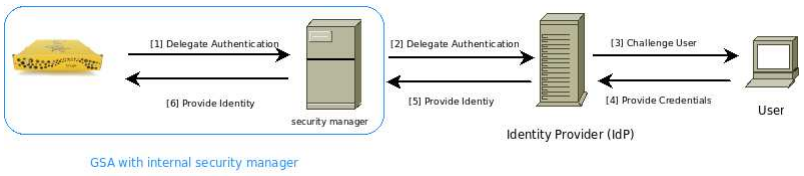

Figure 1: The search appliance communicates through an Identity Provider to authenticate users' access to intranet web pages.

How the Authentication SPI Works

The Authentication SPI exposed by the search appliance is the SAML 2.0 standard; specifically, the "Web Browser SSO Profile." The Web Browser SSO profile makes use of the "Authentication Request Protocol," a request-response protocol. The search appliance sends a SAML `<AuthnRequest>` message to the customer's Identity Provider, and the Identity Provider responds with a SAML `<Response>` message that contains an `<Assertion>`, which in turn contains an `<AuthnStatement>`. These messages are transferred between the search appliance and the customer's Identity Provider, via the browser, using the "HTTP Redirect + Artifact" or "HTTP Redirect + POST" bindings.

The messaging flow for the Artifact and POST bindings are different and are discussed separately:

Authentication using Artifact and Post Binding

- A user enters a search query into a browser.
- The search appliance checks for an existing user session. If it finds a session running, the search appliance supplies the search results and the sequence is complete. If no session exists, an `<AuthnRequest>` element is sent to the Identity Provider (via the Security Manager) using HTTP Redirect binding.
- The HTTP Redirect binding redirects the browser to the Identity Provider with a SAML message sent as a URL query inside the `SAMLRequest` string parameter.
- The Identity Provider authenticates the search user. The IdP can perform this authentication in any way: HTML form, checking an already established session cookie, NTLM, Kerberos, certificate, 2-factor, and so on.

- Depending on the SAML Binding option:

  Artifact Binding

  - The Identity Provider redirects the user back to the Security Manager providing a `SAMLArtifact` token.
  - The Security Manager uses this `SAMLArtifact` and sends an `ArtifactResolve` request to the identity provider's Artifact Resolution URL.
  - The Identity Provider responds with a SAML `<Response>` element to the Security Manager. The `<AuthnStatement>` in the response contains the identity of the search user.

  POST Binding

  - The Identity Provider creates an `AuthnStatement` Assertion for this user, digitally signs the message and responds with an HTML form to auto-submit to the Security Manager. That is, the HTML form returned by the IdP to the user's browser contains an HTML form that is auto-submitted (POST) to the Security Manager (which, in turn, eventually sends the identity to the search appliance).

- The Security Manager sends this identity enclosed in a SAML `<Assertion>` to the search appliance, which uses it for authorization.

Session Cookie

After a search user logs in using the Authentication SPI, the search appliance maintains a session for the search user so that the user doesn't have to reauthenticate to the Identity Provider on every search.

This session is maintained with a cookie named `GSA_SESSION_ID`, which contains the session ID. This cookie is securely sent over HTTPS and is set for the search appliance's hostname only.

Note: The Security Manager tags log entries with the session ID, in order to make it easier to follow a particular session's activity.

HTTP Redirect Binding and SAML AuthnRequest

When a search user performs a query (having no session cookie set), the search appliance communicates with the IdP using HTTP 302 Redirects via the security manager.

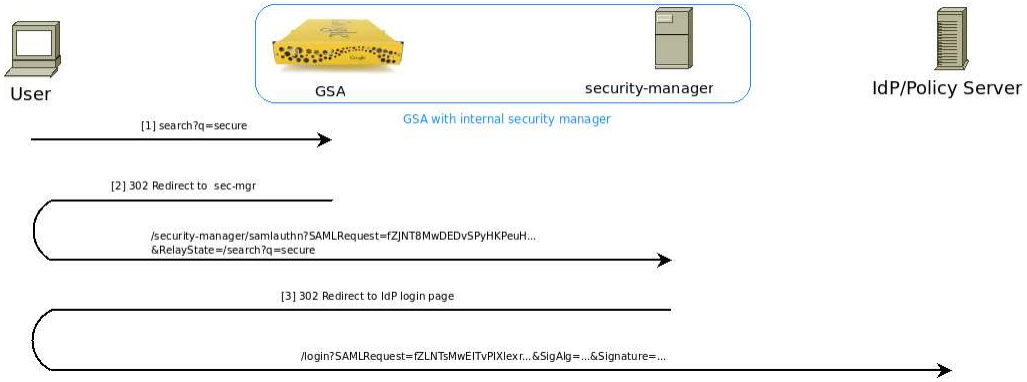

Figure 2: Initial 302 redirect from the search appliance to the IdP via the REDIRECT Binding.

[1] User performs a secure search.

Assume no prior search appliance session or SSO cookie has been granted.

```
http://gsa.yourdomain.com/search?q=secure&btnG=Google
Search&access=s&client=default_frontend&output=xml_no_dtd&proxystylesheet=defaul
t_frontend&sort=date:
D:L:d1&entqr=3&oe=UTF-8&ie=UTF-8&ud=1&site=default_collection

GET /search?q=secure&btnG=Google
+Search&access=a&client=default_frontend&output=xml_no_dtd&proxystylesheet=defau
lt_frontend&sort=date
%3AD%3AL%3Ad1&entqr=3&oe=UTF-8&ie=UTF-8&ud=1&site=default_collection HTTP/1.1
Host: gsa.yourdomain.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.0.19) Gecko/
2010031422
Firefox/3.0.19 (.NET CLR 4.0.20506)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://gsa.yourdomain.com/search?
site=default_collection&client=default_frontend&output=xml_no_dtd&proxystyleshee
t=default_frontend&pr
oxycustom=%3CHOME/%3E
Cookie: COOKIETEST=1
```

[2] Google Search Appliance redirects the user to the security manager.

```
HTTP/1.x 302 Found
Connection: Close
Set-Cookie: GSA_SESSION_ID=f233e6451746aceec55a60e1a8f9708e
Location: https://gsa.yourdomain.com/security-manager/samlauthn?
SAMLRequest=fZJNT8MwDEDvSPyHKPeuHxIMRWvRYJrYBKhAQYJb1rltRuKUOB3w7+k6EHCAq2P7PduZ
nL4ZzbbgSFlMeTyKOAMs7
VphnfL7Yh6c8NPs8GBC0uhWTDvf4C28dECe9ZVIYnhIeedQWEmKBEoDJHwp7qZXlyIZRaJ11tvSas4Ws
5SvVlJvTLOpn7G2gPhsjT
IV6o2tNK4aJSsjy3bTcPbwpZXstBZEHSyQvETfh6I4CqJxEB8XUSKiIxHFT5zln6QzhfsJ/
tNa7ZNIXBRFHkydV5Us/dBkq9bgrvu
KlNfW1hpGpTU7hVwSqW0frqQm4NmwFzGouR8L+R/
c94CeZpFnjfetCMNvSAjowbVOEYQ1ybBIgutxcnZzk+SPy7tlMQl/ELPPs
+xMF7PcalW+s6nW9vXcgfS9pncdcDa3zkj/t1Q8ioeIWgfVkCo6pBZKVSlYcxZme+rv+/e/
4gM=&RelayState=/search?
q=secure&btnG=Google
+Search&access=a&client=default_frontend&output=xml_no_dtd&proxystylesheet=defau
lt_frontend&sort=date
%3AD%3AL%3Ad1&entqr=3&oe=UTF-8&ie=UTF-8&ud=1&site=default_collection
Content-Type: text/html
Content-Length: 0
```

```
GET /security-manager/samlauthn?
SAMLRequest=fZJNT8MwDEDvSPyHKPeuHxIMRWvRYJrYBKhAQYJb1rltRuKUOB3w7+k6EHCAq2P7PduZ
nL4ZzbbgSFlMeTyKOAMs7
VphnfL7Yh6c8NPs8GBC0uhWTDvf4C28dECe9ZVIYnhIeedQWEmKBEoDJHwp7qZXlyIZRaJ11tvSas4Ws
5SvVlJvTLOpn7G2gPhsjT
IV6o2tNK4aJSsjy3bTcPbwpZXstBZEHSyQvETfh6I4CqJxEB8XUSKiIxHFT5zln6QzhfsJ/
tNa7ZNIXBRFHkydV5Us/dBkq9bgrvu
KlNfW1hpGpTU7hVwSqW0frqQm4NmwFzGouR8L+R/
c94CeZpFnjfetCMNvSAjowbVOEYQ1ybBIgutxcnZzk+SPy7tlMQl/ELPPs
+xMF7PcalW+s6nW9vXcgfS9pncdcDa3zkj/t1Q8ioeIWgfVkCo6pBZKVSlYcxZme+rv+/e/
4gM=&RelayState=/search?
q=secure&btnG=Google
+Search&access=a&client=default_frontend&output=xml_no_dtd&proxystylesheet=defau
lt_frontend&sort=date
%3AD%3AL%3Ad1&entqr=3&oe=UTF-8&ie=UTF-8&ud=1&site=default_collection HTTP/1.1
Host: gsa.yourdomain.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.0.19) Gecko/
2010031422
Firefox/3.0.19 (.NET CLR 4.0.20506)
Accept: text/html,application/xhtml xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://gsa.yourdomain.com/search?
site=default_collection&client=default_frontend&output=xml_no_dtd&proxystyleshee
t=default_frontend&pr
oxycustom=%3CHOME/%3E
Cookie: COOKIETEST=1; GSA_SESSION_ID=f233e6451746aceec55a60e1a8f9708e
```

As you can see, the redirect request itself does not contain the hostname of Google Search Appliance. The security manager knows the host/port of the search appliance because it is configured to know it. This step should be immaterial to an external SPI provider (its part of the search appliance --> security manager interaction).

[3] Security manager redirects to the IdP.

The security manager stores some session information (e.g., the `&RelayState=`) and redirects the user to the Identity Provider for authentication challenge (in this case, `http://idp.yourdomain.com:28080/login`). The security manager also adds on a new `&SAMLRequest=` parameter because the original one was in context only between the search appliance and security manager; this one is in context between the security manager and IdP). This request also contains an optional `Signature` for the SAMLRequest.

```
HTTP/1.x 302 Moved Temporarily
Server: Apache-Coyote/1.1
Set-Cookie: JSESSIONID=DBF3B9029E3F442CDB78FABDFD4CFDE6; Path=/security-manager
Date: Fri, 16 Jul 2010 02:05:02 GMT
Cache-Control: no-cache, no-store
Pragma: no-cache
Location: http://idp.yourdomain.com:28080/login?
SAMLRequest=fZLNTsMwEITvPIXlexrHCSRYTVABgUBQFVqQ4IKMswRLiV28TgVvjxMK4kfiaGtndvbb
nR68di3ZgENtTUmTCaMEj
LK1Nk1Jb1YnUUEPqp0pyq5di1nvn801vPSAnswQwfkgO7IG
+w7cEtxGK7i5vijps/drFHEMxr/
phE8AbW07qc1E2U7kRZbGCKp32r9FnTSyARcPPeSnqdqaUnIcmmkj/Rhw8A22rTb96y9TXrCCx
a1ttKHk7LikD2la70uWPKZ1nSaZ2ntU6X6mMs6eFCt2cxnKcBE66g2U9Em2CMMP9nBm0EvjS8pZwiKWR
8neinHBdgXjkyTL7ylZOL
vRNbi57IL41NqmBbLcjkQuP0ai5PaTLB/
IBtYGxciypL0zwkrUKEzwQOGVWM4uL0SoFGtnvVW2pdVIXoyp3Df9//IvirTaAmvGgAO
nYSXg1k4jxA3KeMWjec4Pr6744u58eb76s5dp/C1C9fH6eQrVOw==&SigAlg=http://www.w3.org/
2000/09/xmldsig#rsa-
sha1&Signature=tj/elo8bd/9/
ydMfPUfr1EucF6bUHZtdBwVza5v7+v6i9GsfZjURiXBpBFko8N5iIzLk5RSFPeMgCRkR1HJbNV
pnCog+/7mv39kvMuG21nEyiGvdIArgPBRHCk7uvNNEL4Nr3+fmWMws+8+A/
uKjcmS+NGPNgJLtqp9opI6Slkc=
Content-Length: 0

GET /login?
SAMLRequest=fZLNTsMwEITvPIXlexrHCSRYTVABgUBQFVqQ4IKMswRLiV28TgVvjxMK4kfiaGtndvbb
nR68di3ZgENtTUmTCaMEj
LK1Nk1Jb1YnUUEPqp0pyq5di1nvn801vPSAnswQwfkgO7IG
+w7cEtxGK7i5vijps/drFHEMxr/
phE8AbW07qc1E2U7kRZbGCKp32r9FnTSyARcPPeSnqdqaUnIcmmkj/Rhw8A22rTb96y9TXrCCx
a1ttKHk7LikD2la70uWPKZ1nSaZ2ntU6X6mMs6eFCt2cxnKcBE66g2U9Em2CMMP9nBm0EvjS8pZwiKWR
8neinHBdgXjkyTL7ylZOL
vRNbi57IL41NqmBbLcjkQuP0ai5PaTLB/
IBtYGxciypL0zwkrUKEzwQOGVWM4uL0SoFGtnvVW2pdVIXoyp3Df9//IvirTaAmvGgAO
nYSXg1k4jxA3KeMWjec4Pr6744u58eb76s5dp/C1C9fH6eQrVOw==&SigAlg=http://www.w3.org/
2000/09/xmldsig#rsa-
sha1&Signature=tj/elo8bd/9/
ydMfPUfr1EucF6bUHZtdBwVza5v7+v6i9GsfZjURiXBpBFko8N5iIzLk5RSFPeMgCRkR1HJbNV
pnCog+/7mv39kvMuG21nEyiGvdIArgPBRHCk7uvNNEL4Nr3+fmWMws+8+A/
uKjcmS+NGPNgJLtqp9opI6Slkc= HTTP/1.1
Host: idp.yourdomain.com:28080
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.0.19) Gecko/
2010031422
Firefox/3.0.19 (.NET CLR 4.0.20506)
Accept: text/html,application/xhtml xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://gsa.yourdomain.com/search?
site=default_collection&client=default_frontend&output=xml_no_dtd&proxystyleshee
t=default_frontend&pr
oxycustom=
```

The SAMLRequest is first DEFLATE-compressed, then Base 64 encoded, then URL encoded. It must be decoded and parsed if using the security manager. The following lines of code in python and c# demonstrate the conversion:

```
python:
  def dec_b64_inflate(bstring):
    d_data = base64.b64decode(bstring)
    return zlib.decompress(d_data, -15)

C#:
  public string Decompress(byte[] input)
    {
      using (MemoryStream inputStream = new MemoryStream(input))
      {
        using (DeflateStream gzip =
          new DeflateStream(inputStream, CompressionMode.Decompress))
        {
          using (StreamReader reader =
            new StreamReader(gzip, System.Text.Encoding.UTF8))
          {
            return reader.ReadToEnd();
          }
        }
      }
    }
```

After decompression, the &SAMLRequest= becomes:

```
<?xml version="1.0" encoding="UTF-8"?>
<samlp:AuthnRequest AssertionConsumerServiceURL=
  "https://gsa.yourdomain.com/security-manager/samlassertionconsumer"
  Destination="http://spi.yourdomain.com:28080/login"
  ID="_33d9a01b3dd314c6bc394c420fc0857a"
  IsPassive="false" IssueInstant="2010-07-16T02:05:02.147Z"
  ProviderName="Google security manager"
  Version="2.0" xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol">
  <saml:Issuer xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
    http://google.com/enterprise/gsa/T2-I02BQQ2PYJSJT/security-manager
  </saml:Issuer>
</samlp:AuthnRequest>
```

[4] [5] Authenticate User

The IdP challenges the user to provide credentials to authenticate. If an SSO token is already present on the user's browser, that cookie could be used by the IdP to automatically authenticate without prompting. The IdP is free to use any authentication mechanism available (certificate, SSO, NTLM, Basic, Kerberos).

Figure 3: IdP challenging user for credentials.

After Authentication, the IdP can either use Artifact Binding or POST Binding depending on what mechanism its setup for.

HTTP Artifact Binding

[6a] With Artifact Binding, the IdP generates a random-looking string called an artifact, then associates it with the response from the authentication step. This association is used later to look up the response.

Now the IdP redirects the user back to the security manager (`/security- manager/samlassertionconsumer`) and passes on this artifact in the GET query string. How the IdP determines the host/port of the security manager is up to the IdP. The Referer HTTP header MUST never be used to determine the Google Search Appliance or security manager host/URL. An IdP can simply hard-code the SecurityManager URL in code or in a config file. Although the `AssertionConsumerServiceURL` does properly indicate the URL of the security manager, per specifications, if the request isn't integrity protected, an IdP MUST not rely on its value. From the SAML Core 2.0 Specifications:

```
AssertionConsumerServiceURL [Optional]:
Specifies by value the location to which the Response message MUST be
returned to the requester. The responder MUST ensure by some means that
the value specified is in fact associated with the requester.
```

One way to derive the SecurityManager URL from the `&SAMLRequest=` is to store the URL along with the `<Issuer>`. In the example above, the Issuer is `http://google.com/enterprise/gsa/T2-I02BQQ2PYJSJT/security-manager` which could be stored internally on the IdP in a lookup table associated with URL `https://gsa.yourdomain.com/security-manager/samlassertionconsumer`.

Figure 4: Idp Redirecting with the Artifact Profile.

```
HTTP/1.x 302 Found
Date: Fri, 16 Jul 2010 02:05:06 GMT
Content-Length: 122
Content-Type: text/html
Location: https://gsa.yourdomain.com/security-manager/
samlassertionconsumer?SAMLart=emwjzal36b2dfyoc8en74xmvg9kps5qr
Server: CherryPy/3.1.0

GET /security-manager/samlassertionconsumer?SAMLart=
emwjzal36b2dfyoc8en74xmvg9kps5qr HTTP/1.1
Host: gsa.yourdomain.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.0.19)
Gecko/2010031422 Firefox/3.0.19 (.NET CLR 4.0.20506)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://gsa.yourdomain.com/search? site=
default_collection&client=default_frontend&output=
xml_no_dtd&proxystylesheet=default_frontend&proxycustom=%3CHOME/%3E
Cookie: JSESSIONID=DBF3B9029E3F442CDB78FABDFD4CFDE6;
COOKIETEST=1; GSA_SESSION_ID=f233e6451746aceec55a60e1a8f9708e
```

[6b] [6c] The security manager gets the artifact as the `SAMLart` parameter's value, and sends a SOAP POST to the Identity Provider over a (preferably mutually authenticated) HTTPS connection:

Figure 5: Security manager requesting Artifact Resolve

```
POST /artifact_service HTTP/1.0
Host: idp.yourdomain.com


<?xml version="1.0" ?>
<soap11:Envelope xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/">
  <soap11:Body>
    <samlp:ArtifactResolve ID="_19abdb7e3ada0f44ba2935c8ab53ef54"
      IssueInstant="2010-07-16T02:05:06Z" Version="2.0"
      xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol">
      <saml:Issuer xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
        http://google.com/enterprise/gsa/T2-I02BQQ2PYJSJT/security-manager
      </saml:Issuer>
      <samlp:Artifact>
        emwjzal36b2dfyoc8en74xmvg9kps5qr
      </samlp:Artifact>
    </samlp:ArtifactResolve>
  </soap11:Body>
</soap11:Envelope>
```

The IdP receives this `<ArtifactResolve>` request and looks up the response associated with the artifact. Also, the `Issuer` field in the response must match what was configured in the security manager's admin console. Take special note of the ID field correlation (items in red and purple).

An artifact must not be reusable. Once an artifact is dereferenced, the Identity Provider must reject attempts to dereference the same artifact again.

```
<?xml version="1.0" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
  <samlp:ArtifactResponse ID="amnwsiqxpzamrbl58fce2y1hvgdk3to9"
    InResponseTo="_19abdb7e3ada0f44ba2935c8ab53ef54"
    IssueInstant="2010-07-16T02:05:06Z" Version="2.0"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol" >
    <saml:Issuer>
      myauthn
    </saml:Issuer>
    <samlp:Status>
      <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
    </samlp:Status>
```

```
<samlp:Response ID="mo1sijfykc8mwn7eu9lbzpr2va6godhx"
  IssueInstant="2010-07-16T02:05:06Z" Version="2.0"
  Destination="https://gsa.yourdomain.com/security-manager/
    samlassertionconsumer">
<samlp:Status>
  <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
</samlp:Status>
  <Assertion ID="aup1f3hnzo7gv9kwrds86abc45jxqtye"
    IssueInstant="2010-07-16T02:05:06Z" Version="2.0">
    <saml:Issuer>
      myauthn
    </saml:Issuer>
    <saml:Subject>
      <saml:NameID>
        user1
      </saml:NameID>
      <saml:SubjectConfirmation
        Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
        <saml:SubjectConfirmationData
          InResponseTo="_33d9a01b3dd314c6bc394c420fc0857a"
          NotOnOrAfter="2010-07-16T02:05:11Z"
          Recipient="https://gsa.yourdomain.com/security-manager/
            samlassertionconsumer"/>
      </saml:SubjectConfirmation>
    </saml:Subject>
    <saml:Conditions NotBefore="2010-07-16T02:05:06Z"
      NotOnOrAfter="2010-07-16T02:05:11Z">
      <saml:AudienceRestriction>
        <saml:Audience>
          http://google.com/enterprise/gsa/T2-I02BQQ2PYJSJT/security-manager
        </saml:Audience>
      </saml:AudienceRestriction>
    </saml:Conditions>
    <saml:AuthnStatement AuthnInstant="2010-07-16T02:05:06Z"
      SessionIndex="aup1f3hnzo7gv9kwrds86abc45jxqtye">
      <saml:AuthnContext>
        <saml:AuthnContextClassRef>
          urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
        </saml:AuthnContextClassRef>
      </saml:AuthnContext>
    </saml:AuthnStatement>
  </saml:Assertion>
</samlp:Response>
</samlp:ArtifactResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

HTTP POST Binding

[6d] [6e] With POST Binding, the IdP sends a digitally signed authentication `Assertion` identifying the user to the security manager via an HTML form POST through the browser. That is, the Assertion is digitally signed (XML Digital Signature) using the private key of the IdP. The signed assertion is embedded as a hidden variable (`SAMLResponse`) in an HTML form and transmitted directly to the user's browser. The HTML page that includes a form should auto-submit (automatically form-submit) to the security manager URL which is listening for the POST Binding (`/security-manager/samlassertionconsumer`). As with the Artifact Binding, the IdP needs to determine the URL of the security manager on its own (using hard-coded URL or a lookup table based on the `ISSUER`)

Figure 6: IdP Redirecting with the POST Profile

The following section is taken from the SAML 2.0 Bindings (line 900) [https://www.oasis-open.org/standards#saml].

Note: The RelayState parameter is optional and not specified in this redirect. The IdP sends down the following HTML page to the browser which will automatically POST to the security manager.

```
HTTP/1.1 200 OK
Date: 21 Jan 2004 07:00:49 GMT
Content-Type: text/html; charset=iso-8859-1


<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">

<body onload="document.forms[0].submit()">

<noscript>
<p>
<strong>Note:</strong> Since your browser does not support JavaScript,
you must press the Continue button once to proceed.
</p>
</noscript>

<form action="https://gsa.yourdomain.com/security-manager/samlassertionconsumer"
method="post">
  <div>
    <input type="hidden" name="RelayState" value=""/>
    <input type="hidden" name="SAMLResponse"
value="Jmx0O3NhbWxwOlJlc3BvbnNlIAoJRGVzdGluYXRpb249JnF1b3Q7aHR0cHM6L..."/>
  </div>
  <noscript>
  <div>
    <input type="submit" value="Continue"/>
  </div>
  </noscript>
</form>

</body>
```

With the base64 encoded form of the signed SAML Response:

```
<samlp:Response
  Destination="https://gsa.yourdomain.com/security-manager/samlassertionconsumer"
  ID="mff72db63f0ffa71ac12e40a0ee9270e6be2c58c3" IssueInstant="2014-11-24T23:39:24Z"
  Version="2.0" xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol">
  <saml:Issuer>authn.py</saml:Issuer>
  <samlp:Status>
    <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success" />
  </samlp:Status>
  <saml:Assertion ID="hc2d8aa0d8b55e40560e9ad0fb786483456a17222"
    IssueInstant="2014-11-24T23:39:24Z" Version="2.0">
    <saml:Issuer>authn.py</saml:Issuer>
    <saml:Subject>
      <saml:NameID>admin</saml:NameID>
      <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
        <saml:SubjectConfirmationData
          InResponseTo="_bec64735759b67b8edf15cbbdb1de4f7" NotOnOrAfter="2014-11-
          24T23:40:24Z"
           Recipient="https://gsa.yourdomain.com/security-manager/samlassertionconsumer" />
      </saml:SubjectConfirmation>
    </saml:Subject>
    <saml:Conditions NotBefore="2014-11-24T23:39:24Z"
      NotOnOrAfter="2014-11-24T23:40:24Z">
      <saml:AudienceRestriction>
        <saml:Audience>http://google.com/enterprise/gsa/T2-N7M6W8WPYLSAB/security-manager</
         saml:Audience>
      </saml:AudienceRestriction>
    </saml:Conditions>
    <saml:AuthnStatement AuthnInstant="2014-11-24T23:39:24Z"
      SessionIndex="hc2d8aa0d8b55e40560e9ad0fb786483456a17222">
      <saml:AuthnContext>
        <saml:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtected
        Transport</saml:AuthnContextClassRef>
      </saml:AuthnContext>
    </saml:AuthnStatement>
  </saml:Assertion>
  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:SignedInfo>
       <ds:CanonicalizationMethod
        Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315" />
      <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
      <ds:Reference URI="#mff72db63f0ffa71ac12e40a0ee9270e6be2c58c3">
        <ds:Transforms>
          <ds:Transform
            Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
        </ds:Transforms>
        <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
        <ds:DigestValue>pP4G6cP1a/0vyx/VEnNyJbK3D9Y=</ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>NfhTVkk88qnOJqglor0gPe9vMrvmcPbQW2ROei7Kbl/QCpGEcB/Xq2lnBREaElJW
      ugZgXfeS68BYe39DKdeUJK6qprgQ68Ir9X+C8dHc596lTG5uQluzhC6IdA9cqOdj
      V++XvxOYDNYE6fQoxTaSCynC3QmIkiXwILAJF5l/iTI=</ds:SignatureValue>
    <ds:KeyInfo>
      <ds:X509Data>
       <ds:X509Certificate>MIICdDCCAd2gAwIBAgIJAM1waOO0jGjeMA0GCSqGSIb3DQEBCwUAMFMxCzAJBgNV
          BAYTAlVTMQswCQYDVQQIDAJDQTETMBEGA1UECgwKR29vZ2xlLmNvbTEMMAoGA1UE
          CwwDR1NBMRQwEgYDVQQDDAt0ZXN0LXNlcnZlcjAeFw0xNDExMjQyMjEwMTBaFw0y
          NDExMjEyMjEwMTBaMFMxCzAJBgNVBAYTAlVTMQswCQYDVQQIDAJDQTETMBEGA1UE
          CgwKR29vZ2xlLmNvbTEMMAoGA1UECwwDR1NBMRQwEgYDVQQDDAt0ZXN0LXNlcnZl
          cjCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEA9Y4EvouLKp+U9nPw+VGGJmwg
          7ZqDVa+Fu9+vGWLF5B8EBEybM6VQblJJA29nuKVZ6zAAzltHlTt8fF38VhqqSNT/
          c2xPLpah/tm5i7c2jec6fyTnVXQM/jjGgH6pVWtwdi1iGOuS1RZ2uKHRZCO3eCwp
          TD+V4u0YJ5+nCNZBx4cCAwEAAaNQME4wHQYDVR0OBBYEFGVstcwuuqtKqlziTkYS
```

```
            95BffNamMB8GA1UdIwQYMBaAFGVstcwuuqtKqlziTkYS95BffNamMAwGA1UdEwQF
            MAMBAf8wDQYJKoZIhvcNAQELBQADgYEAuEUabzRjpHoYzrIc6ibc3RsaUJDGZsu9
            2oYPeDeCOjTR4zry9rDUqGocxgGmPjb6Mrb53x8nqih8I+S+DzDvClvhGbLdN5m2
            r9RugFehef7jLAlttvoqioko2Yy+layR+oJe3mrB/nfMpdlFUvz0m/4+Nln6KDLu
            XhPwBUh2PLI=</ds:X509Certificate>
        </ds:X509Data>
      </ds:KeyInfo>
    </ds:Signature>
</samlp:Response>
```

Identity transfer to Google Search Appliance

[7] After parsing the `Assertion`, the security manager now knows the Identity of the user performing the search. It needs to transfer this credential back to the Google Search Appliance using the HTTP-Redirect and HTTP-Artifact Binding. The security manager generates a brand new SAMLArtifact for use in context with this communication and issue a 302 Redirect back to the search appliance. The security manager knows what the user's initial `&RelayState=` is from step [2][3] (using the security-manager's JSESSIONID).

Figure 7: Security manager redirecting via HTTP-Redirect back to the Google Search Appliance

[8] [9] Once the Google Search Appliance has the artifact, it acquires the verified identity by communicating with the security manager internally. This part of the protocol is not shown for clarity.

Authorization

Purpose of the Authorization SPI

The Google Search Authorization SPI is exposed to allow a customer's web service to communicate between the Authorization SPI and the customer's server that provides access control services, which this document refers to as the Policy Decision Point (PDP). The PDP provides a layer between the Google Search Appliance and the customer's access-control system. The PDP will be implemented, tested, and maintained by the customer.

When a user performs a search over access-controlled documents, the user must first authenticate to the search appliance. This allows the search appliance to reference the user's identity when making authorization checks, and to include the search user's identity in search logs.

There is an option to turn off cache links and snippets for access-controlled documents. This allows the administrator to assess the risk of storing access-controlled documents on the search appliance.

As with authentication, the protocol used between the search appliance, the browser, and the PDP is taken from SAML 2.0, an XML-based standard, whose primary use case is inter-domain single sign-on. For example, suppose a user is logged in at organization A, and wants to access content at organization B. Instead of forcing the user to log in again, SAML provides a way for the SSO system at A to vouch for the user by communicating with the SSO system at B. In our scenario, the PDP acts as organization A, while the search appliance acts as organization B.

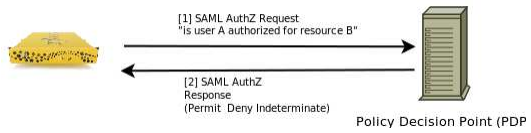

Figure 9: SAML Authorization calls between the Google Search Appliance and the PDP.

Version 6.0 provides batched SAML authorization requests, which you can use under the following conditions:

- Your SAML provider supports the Google SAML batch authorization extension.
- Your system has not been using SAML in any previous search appliance software version.

How the Authorization SPI Works

When the search appliance needs to check whether a search user has access to a URL, it creates a message containing the user identity and the URL, and sends it to an authorization server. This authorization server is the Policy Decision Point (PDP), a service provided by the customer. In response to authorization check requests, the Policy Decision Point responds with a message that says either "Permit," "Deny," or "Indeterminate." (these terms are defined by the SAML standard).

For each URL in a search results list, the search appliance issues an `<AuthorizationDecisionQuery>` element, containing the identity of the user and the URL in question, to the Policy Decision Point. The PDP sends back a `<Response>` message, which contains an `<AuthzDecisionStatement>`, and indicates whether the user is authorized for the URL. These messages are exchanged using the SAML SOAP binding over HTTPS.

The format of these messages are defined by SAML, and they are sent over SOAP over HTTPS. How the SAML messages are embedded in SOAP is also defined by SAML, as the "SAML SOAP binding". For complete details, please refer to the SAML standard.

When the search appliance makes an authorization check, it caches the result. The time that this information is valid is configurable in the Admin Console.

Here are the relevant portions of the SAML schema (see http://www.oasis-open.org/committees/download.php/11903/saml-2.0-os-xsd.zip) for the request:

```
<complexType name="RequestAbstractType" abstract="true">
  <sequence>
    <element ref="saml:Issuer" minOccurs="0"/>
    <element ref="ds:Signature" minOccurs="0"/>
    <element ref="samlp:Extensions" minOccurs="0"/>
  </sequence>
  <attribute name="ID" type="ID" use="required"/>
  <attribute name="Version" type="string" use="required"/>
  <attribute name="IssueInstant" type="dateTime" use="required"/>
  <attribute name="Consent" type="anyURI" use="optional"/>
</complexType>
```

```
<element name="Extensions" type="samlp:ExtensionsType"/>
<complexType name="ExtensionsType">
  <sequence>
    <any namespace="##other" processContents="lax" maxOccurs="unbounded"/>
  </sequence>
</complexType>

<element name="SubjectQuery" type="samlp:SubjectQueryAbstractType"/>
<complexType name="SubjectQueryAbstractType" abstract="true">
  <complexContent>
    <extension base="samlp:RequestAbstractType">
      <sequence>
        <element ref="saml:Subject"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<complexType name="BaseIDAbstractType" abstract="true" mixed="true">
    <complexContent>
        <extension base="anyType">
            <attribute name="NameQualifier" type="string" use="optional"/>
            <attribute name="SPNameQualifier" type="string" use="optional"/>
        </extension>
    </complexContent>
</complexType>

<element name="NameID" type="saml:NameIDType"/>
<complexType name="NameIDType" mixed="false">
    <simpleContent>
        <restriction base="saml:BaseIDAbstractType">
            <simpleType>
                <restriction base="string"/>
            </simpleType>
            <attribute name="Format" type="anyURI" use="optional"/>
            <attribute name="SPProvidedID" type="string" use="optional"/>
        </restriction>
    </simpleContent>
</complexType>
```

```
<element name="Subject" type="saml:SubjectType"/>
  <complexType name="SubjectType">
    <choice>
        <sequence>
            <choice>
                <element ref="saml:BaseID"/>
                <element ref="saml:NameID"/>
                <element ref="saml:EncryptedID"/>
            </choice>
            <element ref="saml:SubjectConfirmation" minOccurs="0"
              maxOccurs="unbounded"/>
        </sequence>
        <element ref="saml:SubjectConfirmation" maxOccurs="unbounded"/>
    </choice>
</complexType>

<element name="AuthzDecisionQuery" type="samlp:AuthzDecisionQueryType"/>
<complexType name="AuthzDecisionQueryType">
  <complexContent>
    <extension base="samlp:SubjectQueryAbstractType">
      <sequence>
        <element ref="saml:Action" maxOccurs="unbounded"/>
        <element ref="saml:Evidence" minOccurs="0"/>
      </sequence>
      <attribute name="Resource" type="anyURI" use="required"/>
    </extension>
  </complexContent>
</complexType>
```

The `<Subject>` element contains the identity of the search user. For the `<Subject>` element, the `<NameID>` element is used. The format of this identity is whatever is passed to the Google Search Appliance from the Authentication portion of the Authorization Server/PDP. The `Resource` attribute is the URL for which we are checking authorization.

For the `Action` element, the attribute for the namespace has the value `urn:oasis:names:tc:SAML:1.0:action:ghpp`. The value for the text of the `Action` element is GET.

The following elements are not sent to the Policy Decision Point by the search appliance.

- `<saml:Issuer>` element
- `<ds:Signature>` element
- `<samlp:Extensions>` element
- `Consent` attribute
- `<SubjectConfirmation>` element
- `NameQualifier` attribute
- `SPNameQualifier` attribute
- `Format` attribute
- `SPProvidedID` attribute
- `<Evidence>` element

Here are some relevant portions of the SAML schema for the response:

```
<element name="Response" type="samlp:ResponseType"/>
<complexType name="ResponseType">
  <complexContent>
    <extension base="samlp:StatusResponseType">
      <choice minOccurs="0" maxOccurs="unbounded">
        <element ref="saml:Assertion"/>
        <element ref="saml:EncryptedAssertion"/>
      </choice>
    </extension>
  </complexContent>
</complexType>

<complexType name="StatusResponseType">
  <sequence>
    <element ref="saml:Issuer" minOccurs="0"/>
    <element ref="ds:Signature" minOccurs="0"/>
    <element ref="samlp:Extensions" minOccurs="0"/>
    <element ref="samlp:Status"/>
  </sequence>
  <attribute name="ID" type="ID" use="required"/>
  <attribute name="InResponseTo" type="NCName" use="optional"/>
  <attribute name="Version" type="string" use="required"/>
  <attribute name="IssueInstant" type="dateTime" use="required"/>
  <attribute name="Recipient" type="anyURI" use="optional"/>
</complexType>

<element name="Status" type="samlp:StatusType"/>
<complexType name="StatusType">
  <sequence>
    <element ref="samlp:StatusCode"/>
    <element ref="samlp:StatusMessage" minOccurs="0"/>
    <element ref="samlp:StatusDetail" minOccurs="0"/>
  </sequence>
</complexType>

<element name="StatusCode" type="samlp:StatusCodeType"/>
<complexType name="StatusCodeType">
  <sequence>
    <element ref="samlp:StatusCode" minOccurs="0"/>
  </sequence>
  <attribute name="Value" type="anyURI" use="required"/>
</complexType>
```

```
<element name="Assertion" type="saml:AssertionType"/>
<complexType name="AssertionType">
  <sequence>
    <element ref="saml:Issuer"/>
    <element ref="ds:Signature" minOccurs="0"/>
    <element ref="saml:Subject" minOccurs="0"/>
    <element ref="saml:Conditions" minOccurs="0"/>
    <element ref="saml:Advice" minOccurs="0"/>
    <choice minOccurs="0" maxOccurs="unbounded">
      <element ref="saml:Statement"/>
      <element ref="saml:AuthnStatement"/>
      <element ref="saml:AuthzDecisionStatement"/>
      <element ref="saml:AttributeStatement"/>
    </choice>
  </sequence>
  <attribute name="Version" type="string" use="required"/>
  <attribute name="ID" type="ID" use="required"/>
  <attribute name="IssueInstant" type="dateTime" use="required"/>
</complexType>

<complexType name="StatementAbstractType" abstract="true"/>

<element name="Issuer" type="saml:NameIDType"/>

<element name="AuthzDecisionStatement" type="saml:AuthzDecisionStatementType"/>
<complexType name="AuthzDecisionStatementType">
  <complexContent>
    <extension base="saml:StatementAbstractType">
      <sequence>
        <element ref="saml:Action" maxOccurs="unbounded"/>
        <element ref="saml:Evidence" minOccurs="0"/>
      </sequence>
      <attribute name="Resource" type="anyURI" use="required"/>
      <attribute name="Decision" type="saml:DecisionType" use="required"/>
    </extension>
  </complexContent>
</complexType>

<simpleType name="DecisionType">
  <restriction base="string">
    <enumeration value="Permit"/>
    <enumeration value="Deny"/>
    <enumeration value="Indeterminate"/>
  </restriction>
</simpleType>

<element name="Action" type="saml:ActionType"/>
<complexType name="ActionType">
  <simpleContent>
    <extension base="string">
      <attribute name="Namespace" type="anyURI" use="required"/>
    </extension>
  </simpleContent>
</complexType>
```

The namespace set in the `Action` element attribute is `urn:oasis:names:tc:SAML:1.0:action:ghpp`. If the string in an `Action` element is "GET", the search appliance displays the URL in the search results, along with snippets and the cache link.

Since the URL found in the cache link (the cache URL pointed to by the cache link, not the URL that points to the original document) is not secret, we must again check the "GET" authorization for a document when the user tries to access the corresponding cache link URL.

If the value for the `Decision` attribute in `AuthzDecisionStatement` is "Indeterminate", rather than "Permit" or "Deny", the search appliance then tries to check authorization using Basic Authentication, NTLM, or Forms Authentication, if they are configured. If they aren't configured, an answer of "Indeterminate" is treated as if authorization was denied.

The following is an example of a message the search appliance sends to the Policy Decision Point:

```
POST /authz HTTP/1.1
Host: pdp.yourdomain.com
Content-Type: text/xml
SOAPAction: http://www.oasis-open.org/committees/security
Content-length: nnn

<?xml version="1.0" ?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <samlp:AuthzDecisionQuery ID="kijcfklibdkjeopfobgifdbknijdjgooccdfaigc"
      IssueInstant="2010-07-16T02:05:07Z"
      Resource="http://content2.yourdomain.com/doc.html"
      Version="2.0" xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
      xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol">
      <saml:Issuer>
        http://google.com/enterprise/gsa/T2-IO2BQQ2PYJSJT
      </saml:Issuer>
      <saml:Subject>
        <saml:NameID>
          user1
        </saml:NameID>
      </saml:Subject>
      <saml:Action Namespace="urn:oasis:names:tc:SAML:1.0:action:ghpp">
        GET
      </saml:Action>
    </samlp:AuthzDecisionQuery>
  </soapenv:Body>
</soapenv:Envelope>
```

The following is an example of a possible response from the Policy Decision Point:

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


<?xml version="1.0" ?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <samlp:Response ID="ew2o7aqtn6ycjwzr5ibh9uef8xl4smpd"
      IssueInstant="2010-07-16T02:05:08Z" Version="2.0"
      xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
      xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol">
      <samlp:Status>
        <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
      </samlp:Status>
      <saml:Assertion ID="ak1qc9vzni72exb4hyw8ugtof6jd3mr5"
        IssueInstant="2010-07-16T02:05:08Z" Version="2.0">
        <saml:Issuer>
          myauthn
        </saml:Issuer>
        <saml:Subject>
          <saml:NameID>
            user1
          </saml:NameID>
        </saml:Subject>
        <saml:AuthzDecisionStatement "Decision="Permit"
          Resource="http://content2.yourdomain.com/doc.html">
          <saml:Action Namespace="urn:oasis:names:tc:SAML:1.0:action:ghpp">
            GET
          </saml:Action>
        </saml:AuthzDecisionStatement>
      </saml:Assertion>
    </samlp:Response>
        </saml:AuthzDecisionStatement>
      </saml:Assertion>
    </samlp:Response>
  </soapenv:Body>
</soapenv:Envelope>
```

SAML Batch Authorization Requests

SAML batch authorization requests enable the search appliance to cache authorization requests for users. For each user who performs a search query that involves secure content, the search appliance first determines the relevant URLs and then determines whether the user has access to the content. The search appliance makes an authorization request to the appropriate web servers and then stores the authorization data. The search appliance uses the cached authorization information for subsequent searches, making those searches faster.

You can use batched SAML authorization requests if your SAML provider supports the Google SAML batch authorization extension. If not, do not use batched SAML authorization requests.

You can enable this feature in the Admin Console:

1. Click **Search > Secure Search > Flexible Authorization**.
2. Select **SAML** from the pull-down menu and click **Add another rule**.
3. Enter the information for the rule, including the **Authorization Service URL** so that the system can access the service when authorization is needed.
4. If you want to use batched SAML requests and your system meets the conditions described in this section, click the **Use batched SAML Authz Requests** checkbox.
5. Click **Save**.

SAML Batched Authorization Usage

With Batched Authorization enabled, the search appliance performs authorization requests by inserting multiple AuthzDecisionQuery elements into a SOAP envelope.

The following is an example of a message the search appliance sends to the Policy Decision Point:

```
POST /authz HTTP/1.1
Host: pdp.yourdomain.com
Content-Type: text/xml
SOAPAction: http://www.oasis-open.org/committees/security
Content-length: nnn
```

```
<?xml version="1.0" ?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <samlp:AuthzDecisionQuery ID="kijcfklibdkjeopfobgifdbknijdjgooccdfaigc"
      IssueInstant="2010-07-16T02:05:07Z"
      Resource="http://content2.yourdomain.com/doc.html"
      Version="2.0" xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
      xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol">
      <saml:Issuer>
        http://google.com/enterprise/gsa/T2-IO2BQQ2PYJSJT
      </saml:Issuer>
      <saml:Subject>
        <saml:NameID>
          user1
        </saml:NameID>
      </saml:Subject>
      <saml:Action Namespace="urn:oasis:names:tc:SAML:1.0:action:ghpp">
        GET
      </saml:Action>
    </samlp:AuthzDecisionQuery>
    <samlp:AuthzDecisionQuery ID="kaaapjecdbephgcciodkdighcaglaojmejkojblg"
      IssueInstant="2010-07-16T02:05:07Z"
      Resource="http://site.yourdomain.com/secure2.html"
      Version="2.0" xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
      xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol">
      <saml:Issuer>
        http://google.com/enterprise/gsa/T2-IO2BQQ2PYJSJT
      </saml:Issuer>
      <saml:Subject>
        <saml:NameID>
          user1
        </saml:NameID>
      </saml:Subject>
      <saml:Action Namespace="urn:oasis:names:tc:SAML:1.0:action:ghpp">
        GET
      </saml:Action>
    </samlp:AuthzDecisionQuery>
  </soapenv:Body>
</soapenv:Envelope>
```

In return, the search appliance expects to receive one or more SAML Response elements inside a SOAP envelope from the Policy Decision Point. The PDP should return the same number of Response elements to correspond with the number of AuthzDecisionQuery elements that the search appliance sent in its request. The ordering of the responses within the SOAP envelope does not matter, but the ID attributes of the AuthzDecisionQueries must be preserved in the Response elements. The following is an example of a possible response from the Policy Decision Point:

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

```
<?xml version="1.0" ?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <samlp:Response ID="ew2o7aqtn6ycjwzr5ibh9uef8xl4smpd"
      IssueInstant="2010-07-16T02:05:08Z" Version="2.0"
      xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
      xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol">
      <samlp:Status>
        <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
      </samlp:Status>
      <saml:Assertion ID="ak1qc9vzni72exb4hyw8ugtof6jd3mr5"
        IssueInstant="2010-07-16T02:05:08Z" Version="2.0">
        <saml:Issuer>
          myauthn
        </saml:Issuer>
        <saml:Subject>
          <saml:NameID>
            user1
          </saml:NameID>
        </saml:Subject>
        <saml:AuthzDecisionStatement "Decision="Permit"
          Resource="http://content2.yourdomain.com/doc.html">
          <saml:Action Namespace="urn:oasis:names:tc:SAML:1.0:action:ghpp">
            GET
          </saml:Action>
        </saml:AuthzDecisionStatement>
      </saml:Assertion>
    </samlp:Response>
    <samlp:Response ID="jli3u2o8cqhsa9nmz4vtxl6rkg7dejpw"
      IssueInstant="2010-07-16T02:05:08Z" Version="2.0"
      xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
      xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol">
      <samlp:Status>
        <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
      </samlp:Status>
      <saml:Assertion ID="zh4x26snw9qzjcpuoy35f1tl7dhgmeak"
        IssueInstant="2010-07-16T02:05:08Z" Version="2.0">
        <saml:Issuer>
          myauthn
        </saml:Issuer>
        <saml:Subject>
          <saml:NameID>
            user1
          </saml:NameID>
        </saml:Subject>
        <saml:AuthzDecisionStatement Decision="Deny"
          Resource="http://site.yourdomain.com/secure2.html">
          <saml:Action Namespace="urn:oasis:names:tc:SAML:1.0:action:ghpp">
            GET
          </saml:Action>
        </saml:AuthzDecisionStatement>
      </saml:Assertion>
    </samlp:Response>
    </saml:AuthzDecisionStatement>
   </saml:Assertion>
  </samlp:Response>
 </soapenv:Body>
</soapenv:Envelope>
```

SPI CallFlow Diagram

The following diagram is the complete call flow for SPI showing both the Artifact and POST Bindings:

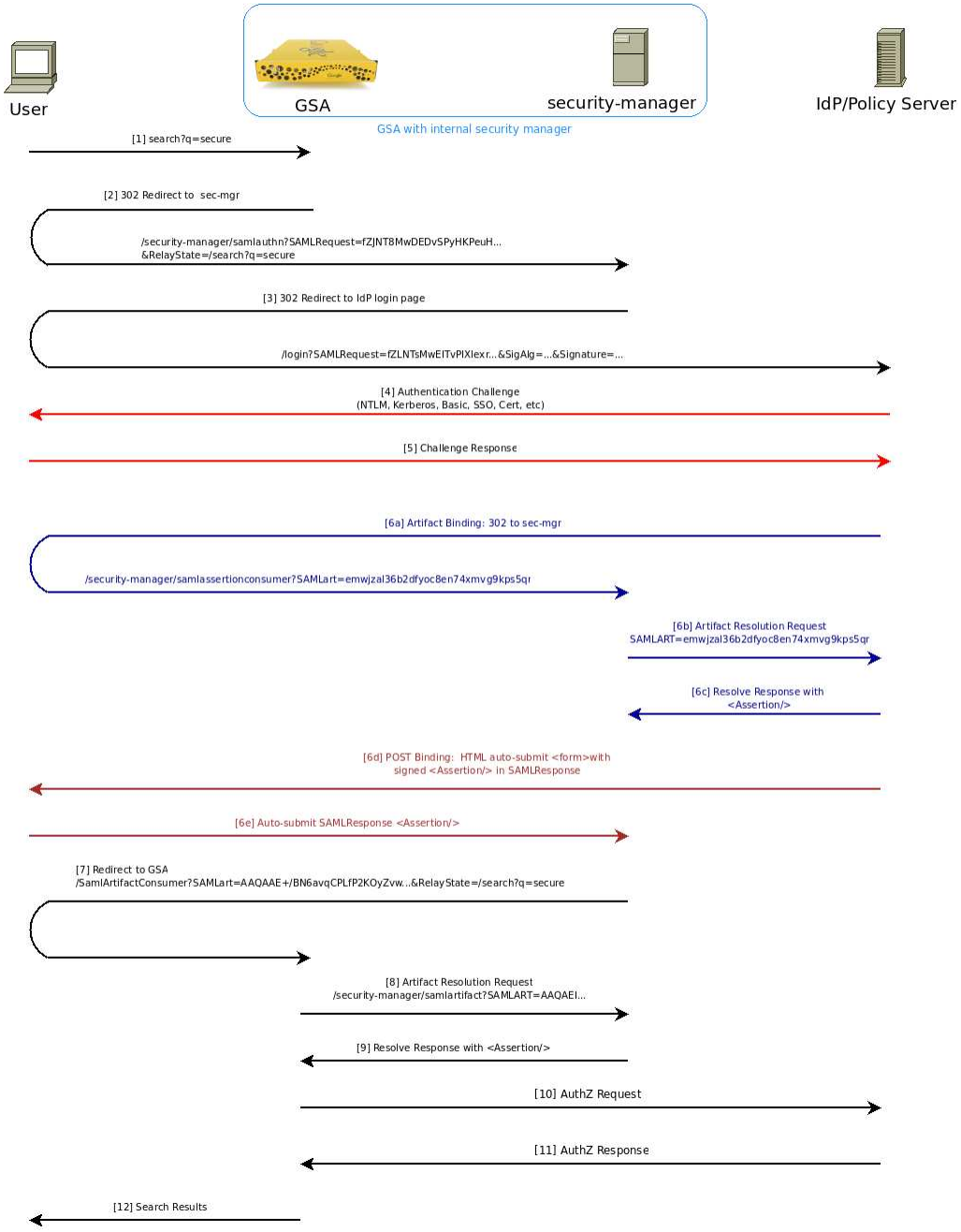

Figure 10: Full SAML SPI calls between Google Search Appliance, security manager, IdP and PDP

Reference

- **GSA Admin Toolkit:** Sample SPI for authentication and authorization [http://code.google.com/p/gsa-admin-toolkit/]
- **SAML 2.0:** [http://www.oasis- open.org/specs/#samlv2.0]
- **Google Search Appliance Universal Login with SPI**: The GSA's security-manager providing universal login forms/SPI. ["The SAML Authentication Service Provider Interface (SPI)" in *Managing Search for Controlled-Access Content*]
- **XML Digital Signatures**: Used for integrity protection of SAML Assertions. [http://www.w3.org/TR/xmldsig-core/]

Troubleshooting

This section provides information for solving problems you might encounter with the SAML Authentication and Authorization SPIs.

Using xmllint to Validate SOAP Messages

If you receive a 500 error from the search appliance during any part of the AuthN/AuthZ process, make sure that the SOAP messages being sent to the search appliance are valid. To check if messages are valid, use `xmllint`.

Index