# Federation in the Enterprise

Patrick Lunney, Product Owner - Single Sign-On & Multi Factor Authentication
Capital One



*To comment on this article, please visit our GitHub repository and submit an issue.*

## Table of Contents

## Abstract

This article describes the fundamentals of enterprise identity federations, focusing on SAML and OpenID Connect (a protocol built on OAuth2.0). It will also contain common scenarios where federations are used and high-level terminology. Academic identity federations are out of scope but are mentioned briefly for comparison.


## Introduction

This article describes identity federation in the context of single sign-on in enterprises and outlines some use cases for enterprise federation integrations. Enterprises have various ways to manage federation connections: the connections may be full service within the enterprise, self-service with controls in place for governance, or manual integrations. Each integration model has its strengths and weaknesses, which will be discussed in turn below.

### Terminology

| Term | Definition |
|---|---|
| Identity Federation | An identity federation is a group of computing or network providers that agree to operate using standard protocols and trust agreements. In a **Single Sign-On (SSO)** scenario, identity federation occurs when an **Identity Provider (IdP)** and **Service Provider (SP)** agree to communicate via a specific, standard protocol. The enterprise user will log into the application using their credentials from the enterprise rather than creating new, specific credentials within the application. By using one set of credentials, users need to manage only one credential, credential issues (such as password resets) can be managed in one location, and applications can rely on the appropriate enterprise systems (such as the HR system) to be the source of truth for a user's status and affiliation.<br><br>Identity federations can take several forms. In academia, **multilateral federations**, where a trusted third party manages the metadata of multiple IdPs and SPs, are fairly common.[i] This article focuses, however, on the enterprise use case where **bilateral federation** arrangements, where the agreements are one-to-one between an IdP and an SP, are the most common form of identity federation in use today. |
| Bilateral Federation | A bilateral federation is one that consists of only two entities: one **Identity Provider (IdP)** and one **Service Provider (SP)**. This is the most common model for an enterprise identity federation. |

| | |
|---|---|
| Identity Provider (IdP) | An Identity Provider (IdP) performs a service that sends information about a user to an application. This information is typically held in a user store, so an identity provider will often take that information and transform it to be able to be passed to the service providers, AKA apps. The OASIS organization, which is responsible for the SAML specifications, defines an IdP as "A kind of SP that creates, maintains, and manages identity information for principals and provides principal authentication to other SPs within a federation, such as with web browser profiles." [ii] |
| Multilateral Federation | A federation that consists of multiple entities that have agreed to a specific trust framework. There are several forms of multilateral federations, including hub-and-spoke and mesh. Multilateral federations are the most common model for academic identity federations. |
| OAuth 2.0 | OAuth 2.0 is an open-source protocol that allows Resource Owners such as applications to share data with clients by facilitating communication with an Authorization Server.[iii] That data takes the form of credentials given to applications to obtain information/data from other applications. The Authorization Server is usually the Identity Provider (IdP). The Authorization Server (AS) may provide authorization directly or indirectly. For example, the AS may supply attributes or profile data of the Resource Owner or provide access to data that can later be used for authorization purposes, such as entitlements from an Identity Management or Governance Solution. |
| OpenID Connect | OpenID Connect is an authentication and authorization framework built on top of OAuth2.0, specifically the authorization_code grant type. It was created to allow not only to authorize clients to obtain information but also includes the ability for clients to obtain information about the user after the user is authenticated. |
| Security Assertion Markup Language (SAML) | SAML is an XML-based communication protocol between SPs and IdPs.[iv] Usually, the enterprise hosts the IdP, whereas applications (including cloud services) are the SPs. |
| Service Provider (SP) | Defined by the OASIS organization, which is responsible for the SAML specification, as "A role donned by a system entity where the system entity provides services to principals or other system entities." This usually takes the form of an application that offers services requiring authentication and authorization to a user. |
| Single Sign-On | Single Sign-On is a centralized portal that enables SPs to verify the identities of **End Users** by facilitating communication with IdPs. SSO acts as a bridge to decouple SPs and IdPs. This can happen via |

| | |
|---|---|
| | numerous protocols such as agent-based integrations, direct LDAP integration, SAML, and OpenID Connect, to name a few. |

## Exploring Identity Federation in the Enterprise

There are several common scenarios where an identity practitioner is likely to encounter identity federation in an enterprise context. This section explores the most common protocols, OpenID Connect, and SAML.

### Use Case 1: SAML

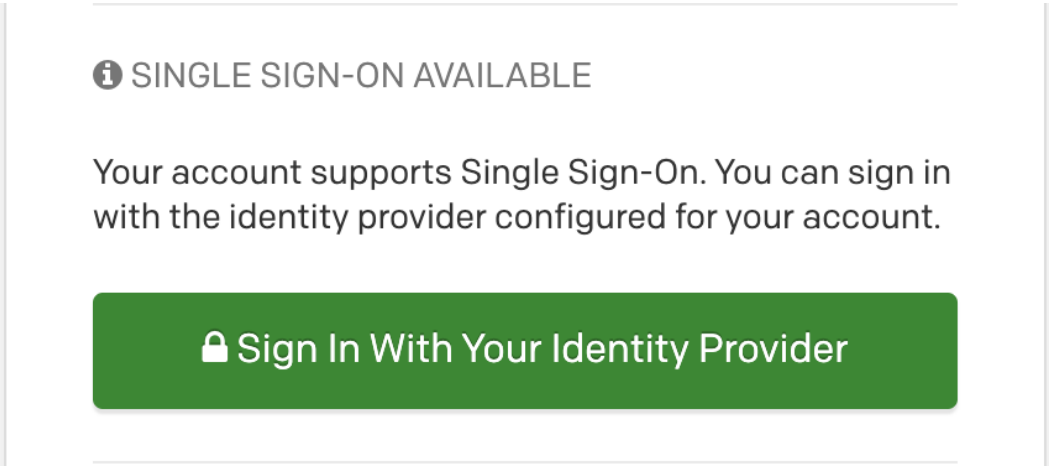


*Figure 1 - Example of a Single Sign-On User Interface*

SAML is most often found in SaaS (Software as a Service) applications. An application is purchased or created by an enterprise to do "something" and employees need to log into the application. The application will need to exchange information with the enterprise to form this federation. Usually, an IdP (the enterprise) and an SP (the app) will exchange metadata, allowing them to set up the connections in the SSO system. Metadata exchange can be done manually, but that often takes time and can cause headaches for IdPs and SPs.

See Appendix Item 1 for an example of a metadata file from an IdP. In that example, the IdP operator will give this metadata to the SP operator. The SP can then input this information manually (or import it, depending on their SSO platform) into their SSO system to allow the enterprise's users to sign in to the application using their SSO accounts. The IdP operator will need to do the same, either by importing an SP metadata file or manually updating the configuration of the IdP.

An IdP metadata file contains the enterprise's entityID, the various URLs used in SAML, and the attributes that will be passed in the SAML assertion (the data that is passed to the app).

An entity ID is a unique name for a SAML entity, both an IdP and an SP. No two IdPs or SPs can share the same entityID.

Think of a SAML assertion as a voucher or ticket. The IdP gives the user a voucher to the user to get into the SP, and the SP is validating the voucher using certificate validation. After the voucher is validated, the SP will look at the attributes to see what the user can do. For example, in Appendix Item 2, you can see a user's username and email address were passed to this SP.

For more information on the details of SAML assertions and components, see the SAML specification and associated supporting documents.[v]

One last piece of information regarding enterprise SAML federations: there are two different types of URLs for applications. Sometimes it is the SP's URL, for example, 'https://myhrapp.com/enterprise'. This is known as an SP-initiated request. Other times, the IdP will initiate the request. For example, 'https://authn.enterprise.com/idp/SAML20=myhrapp'. In both cases, the user will be logging into the same app tenant for the enterprise. Some applications only support IdP-initiated login requests.

Here is a diagram flow of a standard SAML authentication:

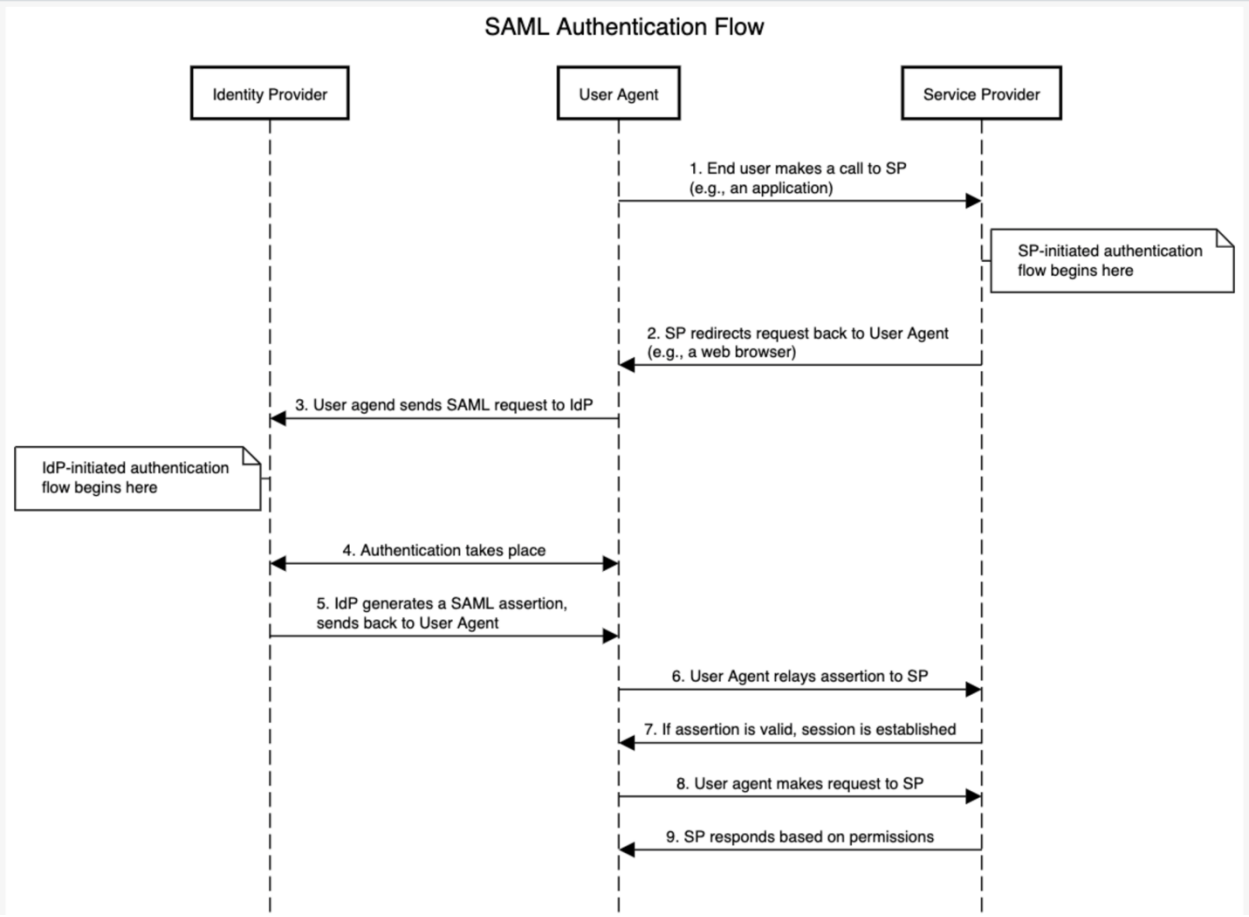


*Figure 2 - SAML Authentication Flow*

It should be noted that the *authentication* of the user is completed at step 5; the IdP has validated the user's credentials and is now passing the SAML assertion back to the browser. Federation is completed at step 7; the browser forwards the assertion to the application so that the application can know the user has been authenticated and create a session for that user. In steps 8 and 9, that is where *authorization* takes place. Based on the information provided by the IdP, the application will allow or deny the user access to certain parts of the application.

## Use Case 2: OpenID Connect

Another common type of identity federation is internal to the enterprise. Previously, enterprises would use "agents," which they would install on web servers hosting applications. The agents would communicate with something called a policy server to determine what a user could do, if anything at all. That agent/policy server technology is old and not used as much in enterprises anymore.

Instead, a popular protocol that is increasingly being used is OpenID Connect. OpenID Connect is newer than SAML and based on the OAuth2.0 protocol; most in-house

enterprise apps are based on APIs and microservices, which is why OIDC is favored.[vi] It should be noted that some SaaS applications do support OpenID Connect.

OpenID Connect uses the authorization_code grant type of OAuth2.0. It is important to note that OpenID Connect is meant to share user attributes, so it will be the only part of OAuth2.0 in this document. There are many other grant_types in OAuth2.0 which authenticate users or clients in different ways but are not part of user *authentication* and *authorization* and are outside the scope of this document.

### Authorization_Code Flow

The authorization_code grant type is explained in the OAuth2.0 spec.[vii] OpenID Connect 1.0 is based on this flow. An important consideration to note involves the scopes in OpenID Connect: they must contain openid (and most often include profile). Here is a diagram of that authorization_code flow:[viii]

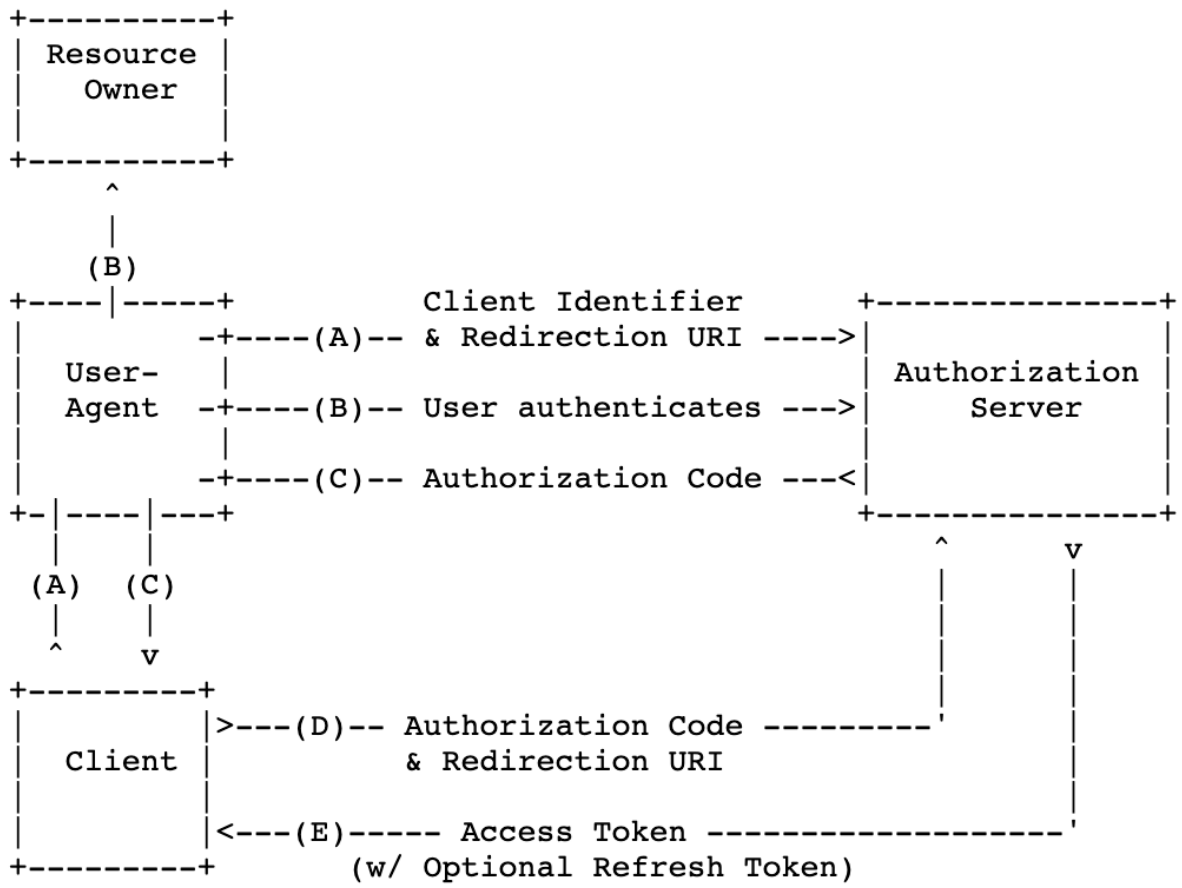


*Figure 3 - OAuth 2.0 authorization_grant Flow*

In this diagram, we can see that the user will first go to a browser (user agent) and initiate a request against the authorization server. The authorization server will then prompt the

user to enter their credentials (B). After collecting the credentials, the browser will send that information to the authorization server, which then will respond with a code to the browser (C). The backend of the application (Client, C) will take that code and exchange it for an access token (D, E). In OpenID Connect, there is an optional step F in which the client may request additional information about the user (attributes) by making an API request against a ‘userinfo’ endpoint. With this API request, the AS will return the user's information allowing the client to *authorize* the user.

To see the API calls, please see Appendix Item 3.

# Challenges in Enterprise Federations

## When to Use SAML versus OpenID Connect

The short answer to this question is: it depends. Sometimes there are limitations as to what SPs can do, as well as IdPs. There are pros and cons to both integrations, so it really is just a matter of choice (or limitation) between the IdPs and SPs.[ix]

The IDPro Body of Knowledge article "Introduction to Identity - Part 2: Access Management" by Pamela Dingle offers an interesting view of the evolution of authentication and access control tools.[x] In particular, the section 'Mobile & API Innovation Gave Us OAuth & Delegated Authorization Frameworks' offers some interesting insights into the evolution that led to the development of OAuth despite the existence of SAML.

## Attributes - Data and Formatting

Applications require different names for attributes. Sometimes an attribute must be called firstname, where other applications may need firstName, or perhaps even givenName. This can cause issues, as the application might not be able to pick up the attribute in the SAML Assertion / userinfo endpoint it needs to authorize the user. This is where the IdP and SP need to collaborate to determine how the attributes should be sent. In some enterprises, the attribute names do not change; the enterprise forces the application to adopt its formatting of the attribute. Other times, the application forces the IdP to change the attributes. There is also something called attribute mapping which can take place. Most SAML and OpenID Connect plugins allow this to take place in attribute mapping files, like Shibboleth.[xi] The IdP will send attributes, and upon receiving them, the SP can transform them into the correct format.

## Assertion Sizing

Quite a bit of information can be passed to SPs, and the assertion can become so large that it will break the SP. This is somewhat common when applications authorize users via Active Directory or LDAP groups (also known as SID bloat, essentially a large data blob of information about the user), and the IdP sends an array of all Active Directory groups. The SAML assertion will contain so much information that the SP will not be able to parse it out, and the user will not be able to get into the application. Resolving this issue often requires custom integrations, where there needs to be a special configuration within the IdP to manage assertions for that single application. Additionally, assertion sizes can be limited based on web servers, browsers, and even proxies. This problem can be alleviated via identity governance processes that limit the number of Active Directory groups and removes memberships no longer required.

## Cross-Origin Resource Sharing (CORS)

Cross-Origin Resource Sharing, commonly known as CORS, causes issues in many enterprises. CORS is a standard that allows a server to relax the same-origin policy.[xii]

Usually, an API call from one application cannot be returned to a separate application. For example, if I make a request to application1.com/api, I would expect the request to come back to me and not be sent to application2.com/api. These are two different domains and application1.com could potentially be sending malicious data to application2.com.

CORS is used to explicitly allow some cross-origin requests while rejecting others. For example, if a site offers an embeddable service, it may be necessary to relax certain restrictions. If I attempt to load application1.com, and that application requires resources from application2.com, my browser will make that request through application1.com into application2.com, thus making it a cross-domain API call.  CORS allows the request to pass through and retrieve information so I can visit the application.

Setting up such a CORS configuration is a challenge. It is also potentially not secure. What most IdPs can do is relax their policies to allow sharing between top-level domains, for example, *.enterprise.com or *.partner.com. This way, there will be no restrictions on the origin of requests.[xiii]

# Conclusion

This document is a high-level review of application federations in the enterprise. The most common protocols used are SAML and OpenID Connect. Both are widely used today in the enterprise world as well as the consumer world as well. When you see this screen:

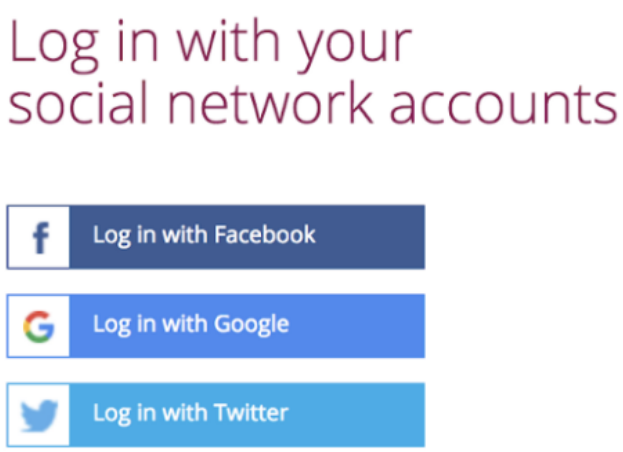


*Figure 4 - Sample Social Login Screen*

you are actually selecting the IdP you'd like to sign into the SP with. You also have the ability (in most cases) to sign up in the app directly. One thing to note, when you do sign in to an application using an Identity Provider such as social media sites, you are passing information about yourself, the same way your enterprise passes information about you to SPs in the enterprise. On social networks, it is important to understand the terms and conditions of what can be done with this data. In enterprise applications, this is usually done by legal teams to ensure there will be no data exfiltration.

With more and more applications becoming SaaS applications, enterprises are creating more and more federations. With that, there will continue to be innovations in the single sign-on community to make them safer, such as adding multifactor authentication into the flow.

## Author Bio

My name is Patrick Lunney. I have managed/owned identity providers in two fortune 50 companies over the past eight years. In that time, I've worked with 100s of SaaS applications as well as in-house applications to ensure federations are set up securely and properly. Currently, I am the product owner for Capital One's internal workforce Single Sign-On and Multi Factor Authentication products. All applications in our enterprise must use either OpenID Connect or SAML for SSO, with very few exceptions. I have held this role since July of 2019.

# Appendix:

### Item 1: SAML Request

```
<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata" ID="mzWO1kVu-
dAmFIdmN.08s9bOaCH" cacheDuration="PT1440M" entityID="IdProvider">
   <md:IdPSSODescriptor protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol"
   WantAuthnRequestsSigned="false">
      <md:KeyDescriptor use="signing">
         <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
            <ds:X509Data>
               <ds:X509Certificate>
               </ds:X509Certificate>
            </ds:X509Data>
         </ds:KeyInfo>
      </md:KeyDescriptor>
      <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified</md:NameIDFormat>
      <md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
      Location="https://authn.enterprise.com/idp/SSO.saml2"/>
      <md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
      Location="https://authn.enterprise.com/idp/SSO.saml2"/>
      <saml:Attribute xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" Name="firstname"
      NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified"/>
      <saml:Attribute xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" Name="groups"
      NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified"/>
      <saml:Attribute xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" Name="lastname"
      NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified"/>
      <saml:Attribute xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" Name="userid"
      NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified"/>
      <saml:Attribute xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" Name="email"
      NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified"/>
   </md:IdPSSODescriptor>
   <md:ContactPerson contactType="administrative"/>
</md:EntityDescriptor>
```

### Item 2: SAML Response

```
<samlp:Response Destination="https://serviceprovider.com/acs"
   ID="HpiyLr_zVMK.jxdUHXxRvjJ8Fwy" IssueInstant="2020-11-24T01:53:06.809Z" 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">IDprovider</saml:Issuer>
   <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/2001/04/xmldsig-more#rsa-sha256"/>
         <ds:Reference URI="#HpiyLr_zVMK.jxdUHXxRvjJ8Fwy">
            <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#"/>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
            <ds:DigestValue>PwJICHFA1QIlML2p5MyJaRib5TDY4TWj5J7IEAjn1Yo=</ds:DigestValue>
         </ds:Reference>
      </ds:SignedInfo>
      <ds:SignatureValue> Signature
      </ds:SignatureValue>
      <ds:KeyInfo>
         <ds:X509Data>
            <ds:X509Certificate>
</ds:X509Certificate>
</ds:X509Data>
         <ds:KeyValue>
            <ds:RSAKeyValue>
                        <ds:Modulus>
                        </ds:Modulus>
               <ds:Exponent>AQAB</ds:Exponent>
            </ds:RSAKeyValue>
         </ds:KeyValue>
      </ds:KeyInfo>
</ds:Signature>
<samlp:Status><samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
</samlp:Status>
<saml:Assertion ID="bJUFiJZEXV0rDgdTh9HnF2CbrIq" IssueInstant="2020-11-24T01:53:07.104Z"
      Version="2.0" xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
      <saml:Issuer>IDprovider</saml:Issuer>
      <saml:Subject>
         <saml:NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">ztl593</saml:NameID>
         <saml:SubjectConfirmation
Method="urn:oasis:names:tc:SAML:2.0:cm:bearer"><saml:SubjectConfirmationData NotOnOrAfter="2020-11-
24T01:58:07.104Z"
            Recipient="https://serviceprovider.com/acs"/></saml:SubjectConfirmation>
      </saml:Subject>
      <saml:Conditions NotBefore="2020-11-24T01:48:07.104Z" NotOnOrAfter="2020-11-24T01:58:07.104Z">
         <saml:AudienceRestriction>
            <saml:Audience>http://www.serviceprovider.com/</saml:Audience>
         </saml:AudienceRestriction>
      </saml:Conditions>
      <saml:AuthnStatement AuthnInstant="2020-11-24T01:53:07.103Z"
         SessionIndex="bJUFiJZEXV0rDgdTh9HnF2CbrIq">
         <saml:AuthnContext>
```

```
<saml:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:Telephony</saml:AuthnContextClassRef>
          </saml:AuthnContext>
       </saml:AuthnStatement>
       <saml:AttributeStatement>
          <saml:Attribute Name="mail" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
             <saml:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">Patrick.Lunney@idprovider.com</saml:AttributeValue>
          </saml:Attribute>
       </saml:AttributeStatement>
    </saml:Assertion>
</samlp:Response>
```

## Item 3: OpenID Connect

To begin the process the user agent will first make a GET request against the authorization server, passing along information about the application the user wishes to go to.

```
curl --request GET \
--header ‘content-type: application/x-www-form-urlencoded' \
 --url
“${sso_prefix}/authorization?response_type=code&redirect_uri=${redirect_uri}&scope="op
enid profile”&client_id=${client_id}
```

What will return from this request is the login page (assuming there is no session), and a user will enter their credentials so the authorization server can authenticate the user. Afterward, an authorization_code is sent to the application in the browser. The application backend must take that authorization_code and exchange it for an access token.

To exchange the authorization_code for the access token:

```
curl --request POST \
      --url “https://${sso_prefix}/token” \
      --header 'content-type: application/x-www-form-urlencoded' \
      --header 'Authorization: Basic base64(urlencode("${client_id}:${client_secret}))' \
      --data “code=${code}” \
      --data “grant_type=authorization_code” \
      --data “redirect_uri=${redirect_uri}” \
      --data 'scope=openid profile'
```

After this exchange, the application can then make a backend API call to the authorization server to obtain additional information about the user for further authorization.

```
curl --request GET \
--header ‘content-type: application/x-www-form-urlencoded' \
--header 'Authorization: Bearer ${token}
 --url “${sso_prefix}/userinfo
```

This will give applications information like this:

```
{
  "sub"       : "83692",
  "name"      : "Alice Adams",
  "email"     : "alice@example.com",
  "department" : "Engineering",
  "birthdate"  : "1975-12-31"
}
```

---

[i] “Multilateral federation,” InCommon Federation wiki, last updated 17 February 2020, https://spaces.at.internet2.edu/display/federation/Multilateral+federation.

[ii] Hodges, Jeff, Rob Philpott, Eve Maler, eds. “Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0,” OASIS Standard, 15 March 2005, https://docs.oasis-open.org/security/saml/v2.0/saml-glossary-2.0-os.pdf.

[iii] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[iv] Ragouzis, Nick, John Hughes, Rob Philpott, Eve Maler, Paul Madsen, Tom Scavo, eds. “Security Assertion Markup Language (SAML) V2.0 Technical Overview,” OASIS Committee Draft, 25 March 2008, https://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.pdf.

[v] OASIS Standards landing page, https://www.oasis-open.org/standards/.

[vi] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[vii] Ibid, see Section 4.1.

[viii] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, Section 4.1, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[ix] For further discussion on the pros and cons between SAML and OAuth, see https://www.okta.com/identity-101/whats-the-difference-between-OAuth-openid-connect-and-saml/ or https://auth0.com/intro-to-iam/saml-vs-openid-connect-oidc/

[x] Dingle, Pamela, “Introduction to Identity – Part 2: Access Management,” IDPro Body of Knowledge, 17 June 2020, https://bok.idpro.org/article/id/45/.

[xi] Shibboleth Consortium, https://www.shibboleth.net/.

[xii] “Same-origin Policy,” MDM Web Docs, https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy.

[xiii] For additional information, see https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS and https://web.dev/cross-origin-resource-sharing/.