**Web App SSO was Created to Help Active Directory**

Before web-based applications were created, the typical user utilized Microsoft Windows® for their system, Microsoft Office for productivity software, and Microsoft Exchange for email. IT admins used Microsoft Active Directory on Windows Servers to manage users and systems. Suffice it to say, the office was dominated by on premises , Microsoft technology. The dominant identity provider at the time, Active Directory, wasn’t built to connect with these non-Microsoft, web-based applications. Web app SSO providers have done well, and both [Okta](https://jumpcloud.com/blog/okta-competition/) and [OneLogin SSO](https://jumpcloud.com/blog/google-idaas-onelogin/) have matured over the last ten years. Both SSO solutions were built well, and both support thousands of applications. They also integrate seamlessly with Microsoft Active Directory and OpenLDAP™ – the core identity providers most organizations are using.

**List of Idas(Identity as a service) providers**

Google, facebook, Linkedin, yahoo. As well as Okta and Onelogin are also worked as identity provider.

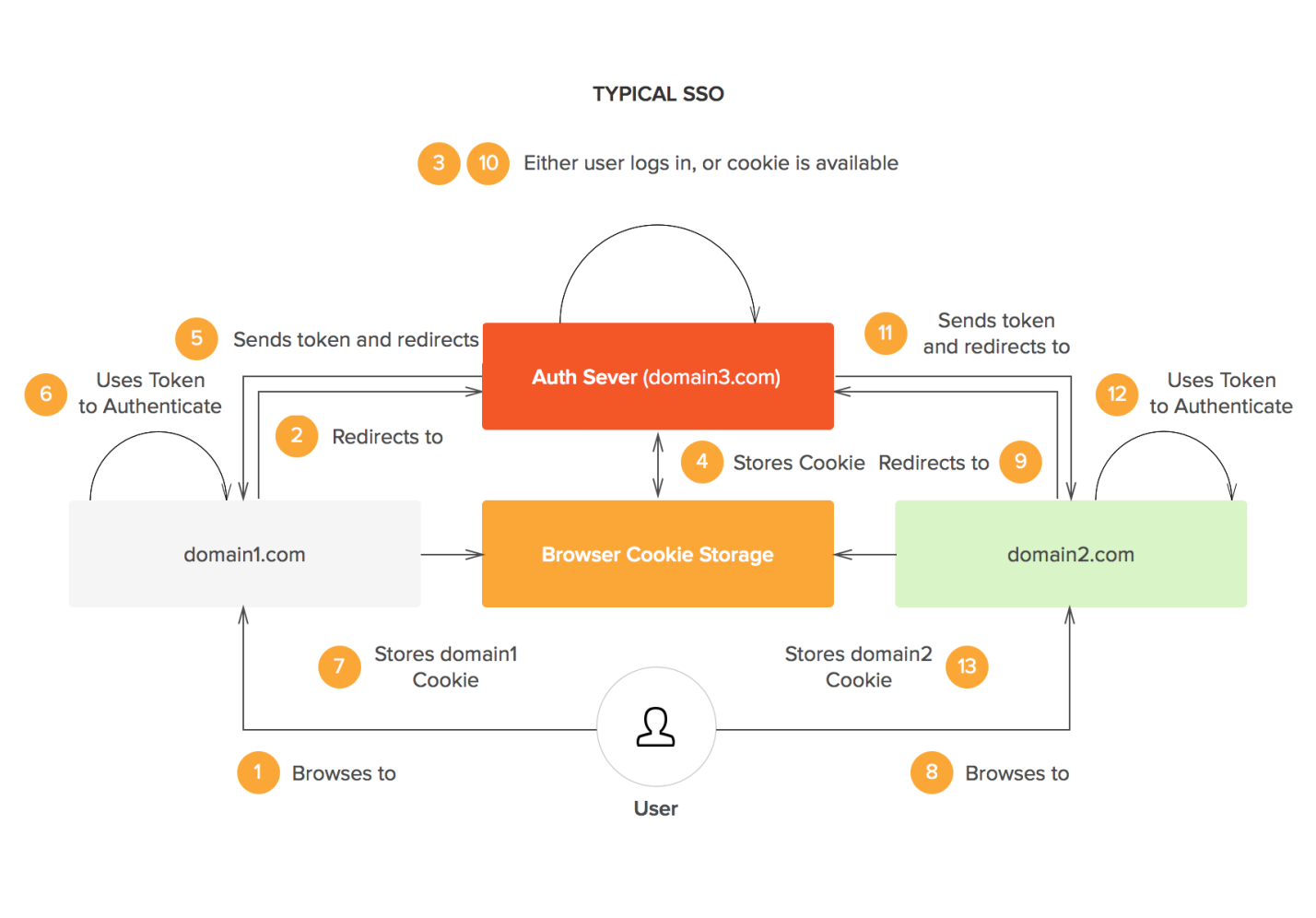
**Difference between LDAP, ADFS and SAML**

LDAP and AD(Active directory) are same, both are identity directory used for on premises single sign on. On other hand for SAML assertion would be leveraged by a service provider—or web application—via a secure XML exchange. Vendors used SAML to create software that could extend one user identity from AD to a host of web applications, creating the first generation of Identity-as-a-Service (IDaaS)—single sign-on (SSO) solutions. So LDAP and AD is used as Daas or directory as a service and SAML used for Idaas Identity as a service.

**Single sign on basics**

With SSO login, a *central authentication server* needs to exist. Let's call our central authentication server **foobarbaz.com**. This is how the process flow will look in this scenario:

1. The user accesses **foo.com**.
2. The user is redirected to **foobarbaz.com**, where an authentication-related cookie is generated.
3. The user navigates to **bar.com**.
4. The user is redirected to **foobarbaz.com**.
5. **foobarbaz.com** checks whether the user already has an authentication-related cookie and redirects the user back to **bar.com**, providing access to its features and content.
6. The same process applies to **baz.com**.



**Different SSO mechanism ?**

Open Id connect (Built on top of Oauth2.0)

SAML2.0

**What is Oauth2.0 and OpenId connect ?**

<https://www.youtube.com/watch?v=LyqeHAkxVyk>

<https://developers.google.com/identity/protocols/OAuth2>

Oauth2.0 playground

**Difference between OAuth2.0 and SAML2.0**

<https://www.gluu.org/resources/documents/articles/oauth-vs-saml-vs-openid-connect/>

**When to use which mechanism ?**

Mobile applications: no question–use OpenID Connect

If you are writing a new application, use OpenID Connect–skate to where the puck is going!

For only webapp we can use SAM2.0 but Openid can also be used.

**What is SAML metadata file ?**

SAML Metadata is used to exchange configuration information between Service Provider and Identity Provider and vice versa. The information can include:

* Binging location
* Organization name
* Contact Person
* Single Sign On Url
* Single Logout Url

The Metadata can be signed and encrypted so that the information is sent securely. The other side may need the corresponding public key to validate and decrypt it and then can be used to understand and establish the connection with the SP or IdP.

**How SAML2.0 works ?**

The user accesses the remote application using a link on an intranet, a bookmark, or similar and the application loads.

The application identifies the user’s origin (by application subdomain, user IP address, or similar) and redirects the user back to the identity provider, asking for authentication. This is the authentication request.

The user either has an existing active browser session with the identity provider or establishes one by logging into the identity provider.

The identity provider builds the authentication response in the form of an XML-document containing the user’s username or email address, signs it using an X.509 certificate, and posts this information to the service provider.

The service provider, which already knows the identity provider and has a certificate fingerprint, retrieves the authentication response and validates it using the certificate fingerprint.

The identity of the user is established and the user is provided with app access.

**How SLO or Single logout works for SAML2.0 ?**

<https://www.portalguard.com/blog/2016/06/20/saml-single-logout-need-to-know/>

**Microservice authentication**

<https://dzone.com/articles/authentication-and-authorization-in-microservices>

**What is LDAP ?**

Ldap is a light weight directory which stores user related informations. like this :

ldapAttributes.add(*createAttribute*("cn", user.getEmailAddress()));

ldapAttributes.add(*createAttribute*("sn", user.getLastName()));

ldapAttributes.add(*createAttribute*("uid", user.getEmailAddress()));

ldapAttributes.add(*createAttribute*("userPassword", *generatePassword*(***PASSWORD\_LENGTH\_MIN***, ***PASSWORD\_LENGTH\_MAX***)));

ldapAttributes.add(*createAttribute*(props.m\_username, user.getEmailAddress()));

ldapAttributes.add(*createAttribute*("givenName", user.getFirstName()));

ldapAttributes.add(*createAttribute*(props.m\_firstName, user.getFirstName()));

ldapAttributes.add(*createAttribute*(props.m\_lastName, user.getLastName()));

ldapAttributes.add(*createAttribute*(props.m\_displayName, UserDAO.*getDisplayName*(user.getFirstName(), user.getLastName())));

ldapAttributes.add(*createAttribute*(props.m\_email, user.getEmailAddress()));

ldapAttributes.add(*createAttribute*(props.m\_phone, user.getPhoneNumber()));

ldapAttributes.add(*createAttribute*(props.m\_mobile, user.getMobileNumber()));

ldapAttributes.add(*createAttribute*(props.m\_title, user.getPosition()));

ldapAttributes.add(*createAttribute*(props.m\_language, user.getLocale()));

This is a separate server which we can attach with weblogic or any cloud environment.

**How login works in Prime application ?**

When we create any user we create a user object at prime and create ldap user with auto-generated password in Ldap server. add a foreign key field in User table as ldapid.

Then when we attach permissions it will add some policydata with the ldapid in Authorization microservices and the security assignment in Prime database.

Step 1 : In submit form action = “j\_security\_check” after submit it will hit weblogic authentication form or in OCI authentication form.

Step 2 : Weblogic or OCI will check the user from Ldap server and send success response , In response set the session id and set attribute ldapid in backend session storage.

Step3 : For any call UI make call to the application with the jsessionid and prime microservice create a jwtToken with some expirydate and send the ldapid and a key to authorization microservice, get the permissions.

Step4: Depending on the permissions send the details to client.

**Logging mechanism using ELK stack in Microservice system**

<https://dzone.com/articles/deploying-springboot-in-ecs-part-1>

**How exception handled in your system ?**

1. Create messages key in properties files which will work as resource bundles.
2. In BasicException call ResourceBundle and convert the message key and error code to actual message
3. Create a ExceptionMapper class which will implelemts jersey ExceptionMapper and override toResponse(Exception e) to Response whenever any exception thrown.