International Journal of Management, IT & Engineering
Vol.14 Issue 9, September 2024,
ISSN: 2249-0558 Impact Factor: 7.119
Journal Homepage: http://www.ijmra.us, Email: editorijmie@gmail.com
Double-Blind Peer Reviewed Refereed Open Access International Journal - Included in the International Serial Directories Indexed & Listed at: Ulrich's Periodicals Directory ©, U.S.A., Open J-Gate as well as in Cabell's Directories of Publishing Opportunities, U.S.A

Managing Identity in Salesforce CRM

Prajeet Gadekar*

Abstract

Identity solutions in Salesforce are crucial for ensuring secure and seamless access to the platform. They help in managing user identities, enforcing security policies, and providing a single sign-on (SSO) experience, which enhances user productivity and reduces the risk of unauthorized access. By integrating identity solutions, organizations can streamline user management, improve compliance with regulatory requirements, and protect sensitive data. Two common identity protocols used in Salesforce are SAML (Security Assertion Markup Language) and OIDC (OpenID Connect).

SAML is an XML-based protocol that allows secure web domains to exchange user authentication and authorization data. It enables SSO by allowing users to authenticate once and gain access to multiple applications without re-entering credentials. OIDC, on the other hand, is a modern authentication protocol built on the OAuth 2.0 framework. It provides an identity layer on top of OAuth 2.0, allowing clients to verify the identity of the end-user based on the authentication performed by an authorization server. Both SAML and OIDC play a vital role in enhancing security and user experience in Salesforce environments.

In this paper we look in the details of both approaches and compare them and come to a conclusion and which approach works best in which use cases.

Keywords:

Salesforce;
Identity Solutions;
SAML;
Open ID Connect;
OIDC.

Author correspondence:

Prajeet Gadekar,
Salesforce Inc,
1095 Avenue of the Americas,Newyork city, New York, 10036, USA
Email: prajeetgadekar@gmail.com

1. Introductionto Identity Methods

1.1. SAML

Lets first start with **SAML**. At the very core of this concept is trust. There is a trusted relationship between SP's (applications) and the Identity Provider (IDP). The SP's will rely on the IDP to make the decision from them. When a user requests access to an application, the first thing the app will do is check if the user already has a running session on the app, if so then great, the user just gets the access. If the user does not have an ongoing session the application will generate a SAML request for the IDP. The user will then be authenticated by the IDP, perhaps their credentials could be validated against an Identity store. The IDP could also challenge the user and ask them to go through a second form of authentication (MFA). Once it has completed all the checks it is supposed to, the

IDP will generate an assertion. This is where trust plays a role. The application or the SP trusts the assertion. Trust needs to be secure so there are certificates involved in signing the request and encoding to establish this trust (more on that later).. The SAML assertion identifies the user that is then matched by the SP to identify a user in the application and access is granted to the user.

Here is the SP initiated flow

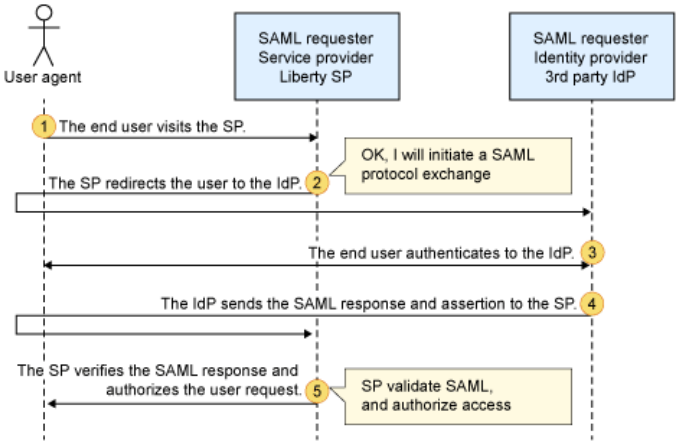

Figure 1. *SP initiated SAML flow*

A more simpler but similar flow is the IDP initiated flow. In this flow the user authenticates himself with the IDP and then clicks on a link to log in to the application. The flow is initiated by the IDP so there is no need to send the SAML request to the IDP.

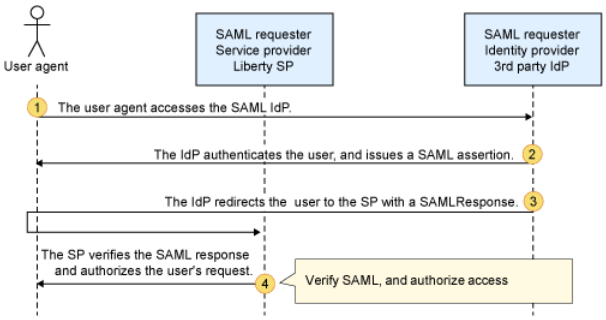

Figure 2. *IDP initiated SAML flow*

Here are some key terms to understand in a SAML flow

Issuer : The unique identifier of the identity provider

Entity ID : Unique URL that identifies who the SAML assertion is intended for, essentially the SP. Note that both the issuer and entity id values in the SAML assertion has to match the values in the settings in Salesforce and the IDP. So unless it does match we are going to get an error. Specifically the Audience attribute in the SAML has to match entity ID on the SSO settings page.

Subject : The nameidentifier element in the Subject statement will specify the user. Most common is for the SAML User ID type to be Dalesforce User objects federationID. You need to make sure you have this same value in the Federation id field of your User Object. It could be an email or something like an employee id of an organization. This is the primary mechanism to map users between the IDP and your Salesforce org. There are alternate attributes in which you can specify the user, and how it relates to Salesforce,

Salesforce username is one of them but it's unusual to expect your IDP to know the Salesforce user name. This is how subject looks like when using federationID

IDP's authentication certificate : Remember the assertion is signed. So your IDP is going to use a private key that no one else has, to sign the request. If you want to validate the signature, the IDP will provide a public key to all its recipients. This public key is stored in salesforce and referred to in the SAML settings. This certificate is the primary mechanism to establish trust between the IDP and Salesforce. If Salesforce is able to validate the signature with the public key it knows that the assertions could only have come from the IDP because no one else has the private key for that signature.

Assertion Decryption Certificate: The contents of the assertion, that is the user's information etc can be encrypted by the IDP. So they are essentially encrypting with the public key and they will give the private key to decrypt the content to only parties that they want to decrypt the content. So essentially the IDP will provide a Salesforce with the private key that will be uploaded in the settings as a decryption certificate.

Request signing Certificate: When Salesforce does not have a session and generates a SAML request to be sent to the IDP, that request can be signed. It has to be signed by a private key, so this cert represents the private key to signing the request. For the SP initiated flow we will have to provide a public key to the IDP so they can validate the signature.

Request binding : Based on the preference of your IDP the SAML request could either be a HTTP POST sending a SAML message using a bse64-encoded HTML form OR a HTTP Redirect sending a base64-encoded and URL encoded SAML messages with URL parameters.

Single Logout: If the desire is to log out of all applications that the IDP supports when they click login out of any one application, generally a chaining mechanism is used. In Salesforce we specify a URL to redirect after the Salesforce session is logged out. This URL could point to the IDP single logout mechanism which can make sure every application's logout method is invoked.

Just in Time: If we want to provision users in real time when they try to login for the first time or update the user record when they login the concept of just in time provisioning can be used. IT consists of a JIT Handler apex class that has the logic to create the user. Note that JIT is useless when it comes to deprovisioning.

Identity Provider Login URL : This is where Salesforce or the Service provider will redirect when there is not active session and needs IDP's intervention.

ACS URL's : This is where the IDP will redirect with the assertion. The SAML request usually sends the ACS URL to the IDP. A list of ACS URL's are also added to the IDP's settings, a request cannot have an ACS URL that is not already in the IDP settings. When the SAML setting is saved in Salesforce ACS URLs for core and any experience sites are generated.

RelayState parameter: It's an important parameter that stores the exact resource location within the SP. You generally do not have to deal with it until you are doing something custom in your solution like posting/redirecting a request to a custom endpoint instead of the IDP directly. Your solution will have to make sure it keeps propagating the relaystate so the exact resource on the SP can be reached when the assertion is posted.

1.2. OIDC

Now lets understand **Open ID Connect.**

OpenID Connect (OIDC) is an identity layer built on top of the OAuth 2.0 protocol, which allows clients to verify the identity of an end-user based on the authentication performed by an authorization server. It also enables clients to obtain basic profile information about the end-user in an interoperable and REST-like manner. OIDC uses OAuth 2.0 methods such as the Authorization Code Flow, Implicit Flow, and Hybrid Flow to facilitate secure authentication and authorization. The Authorization Code Flow is the most commonly used method, where the client first obtains an authorization code from the authorization server and then exchanges it for an access token and an ID token. The ID token is a JSON Web Token (JWT) that contains information about the user, such as their identity and authentication details.

In Salesforce, OpenID Connect is implemented to allow users to log in using their credentials from an external identity provider (IdP) that supports OIDC, such as Google or Facebook. This is achieved by configuring Salesforce as a relying party (RP) in the OIDC flow. Salesforce requires the use of certificates to ensure secure communication between the IdP and Salesforce. These certificates are used to sign and validate the tokens exchanged during the authentication process. When a user attempts to log in, Salesforce redirects them to the IdP's authorization endpoint, where they authenticate. Upon successful authentication, the IdP sends an authorization code back to Salesforce, which is then exchanged for an ID token and access token. Salesforce uses the ID token to verify the user's identity and grant access to the appropriate resources.

It is important to distinguish between authorization and authentication in this context. Authentication is the process of verifying the identity of a user, ensuring that they are who they claim to be. This is achieved through the OIDC protocol, where the IdP authenticates the user and issues an ID token. Authorization, on the other hand, is the process of determining what resources the authenticated user is allowed to access. This is managed through OAuth 2.0, where the access token obtained during the OIDC flow is used to authorize the user's access to specific resources and services. By combining these two processes, OIDC provides a secure and seamless way to authenticate users and authorize their access to resources in Salesforce.

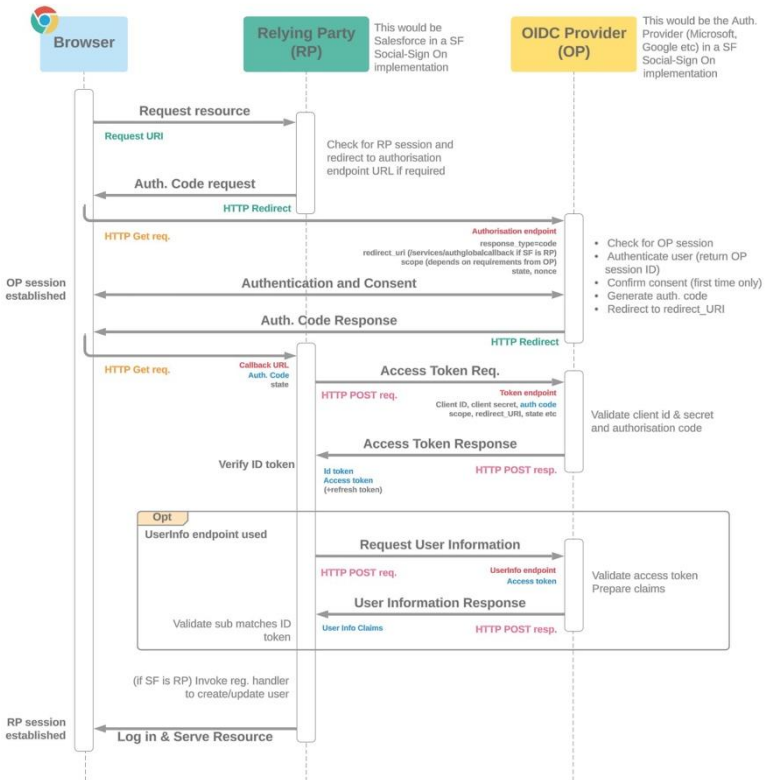

Figure 3. *OIDC flow*

2. ResearchMethod

For our study we implemented both SAML and OIDC based authentications with Saelsforce as the Service provider and Okta as the Identity provider.

The process involved configurations on either side and a set of certificates to establish trust. We used self signed certificates. We also tried using both methods to authenticate in to mobile apps and API integrations both soap and rest.

Salesforce was configured both as an SP and IDP in SAML set ups. Similarly Salesforce was set up as a relying party and it another scenario it as also set up as identity provider that was used to connect to external apps.

As we proceded we documented certain considerations that could help in the comparison and assist upfront when trying to make a decision either way.
These are some of those considerations
1. Where will be the identity store
2. Is there already an IDP that is used to log in to other applications
3. How will users be provisioned in Salesforce
4. How do we inactivate users in Salesforce
5. Do Salesforce profiles, permission sets, roles etc map to anything that the IDP determines.
6. What protocol should we be using
7. Can Salesforce act as an identity provider for itself and some sales or service related applications.

8. How do we provide multi factor authentication (Remember now it is mandatory according to Salesforce's client contract)
9. Is there a requirement for a single logout (logging out of all applications when logging out of the IDP).

3. ResultsandAnalysis

Here are the results on our comparison. At a very high level. Both methods are able to solve most use cases but are better in specific use cases. SAML is very limited in mobile use cases.

Table 1. Comparison on SAML and OIDC

Feature	SAML	OIDC (OpenID Connect)
Protocol Type	XML-based	JSON/REST-based
Primary Use Case	Enterprise SSO (Single Sign-On)	Consumer SSO and API authentication
Complexity	More complex	Simpler and more modern
Token Format	XML (SAML Assertions)	JSON Web Tokens (JWT)
Identity Provider	IdP (Identity Provider)	OP (OpenID Provider)
Service Provider	Service Provider | SP (Service Provider)	RP (Relying Party)
Mobile Support	Limited	Excellent
Standardization	Standardization. Older, well-established	Newer, rapidly growing
Use in APIs	Rarely used, Not suitable	Commonly used
Session Management	Session Management | More complex	Simpler
Adoption	Widely adopted in enterprise environments	Widely adopted in enterprise environments | Increasingly adopted in modern apps
Security	Strong, Mature security features	Strong and modern security features
Interoperability	High but requires more config	Higher and easier to config
User experience	Redirects can make experience less optimal	Seamless user experience

4. Conclusion

Both SAML (Security Assertion Markup Language) and OIDC (OpenID Connect) are robust protocols for authentication and authorization, each with its own strengths and ideal use cases.
SAML is a mature, XML-based protocol that has been widely adopted in enterprise environments for Single Sign-On (SSO). It excels in scenarios where complex, secure, and federated identity management is required, particularly within large organizations. SAML's

extensive support for enterprise applications and its well-established security features make it a reliable choice for internal corporate systems, legacy applications, and environments where XML-based communication is already in place.
OIDC, on the other hand, is a more modern, JSON/REST-based protocol designed with simplicity and flexibility in mind. It is particularly well-suited for consumer-facing applications, mobile apps, and API authentication. OIDC's use of JSON Web Tokens (JWT) and its seamless integration with OAuth 2.0 make it an excellent choice for modern web applications, microservices architectures, and scenarios where quick and easy integration is a priority.
Here are some recommendations specific to use cases
1. Enterprise SSO and Legacy Systems:
Use SAML: If you are dealing with enterprise-level Single Sign-On (SSO) requirements, especially in environments with existing SAML infrastructure or legacy systems, SAML is the preferred choice. Its robust security features and extensive support for enterprise applications make it ideal for these scenarios.
2. Modern Web Applications and APIs:
Use OIDC: For modern web applications, especially those that require API authentication, OIDC is the better option. Its simplicity, use of JSON, and seamless integration with OAuth 2.0 make it highly suitable for these use cases. OIDC is also ideal for microservices architectures and applications that need to scale quickly.
3. Mobile Applications:
Use OIDC: If you are developing mobile applications, OIDC is the clear winner. Its design is mobile-friendly, and it provides excellent support for mobile authentication flows, making it easier to implement and manage.
4. Consumer-Facing Applications:
Use OIDC: For consumer-facing applications where user experience and ease of integration are critical, OIDC is the recommended choice. Its modern approach and support for social logins (e.g., Google, Facebook) enhance the user experience and simplify the authentication process.
5. Hybrid Environments:
Use Both: In some cases, you may need to support both SAML and OIDC, especially in hybrid environments where both legacy enterprise systems and modern applications coexist. Many identity providers support both protocols, allowing you to leverage the strengths of each where appropriate.
In summary, choose SAML for enterprise-level, complex, and secure SSO needs, and opt for OIDC for modern, flexible, and scalable authentication solutions, particularly in consumer-facing and mobile applications.

References`

[1] SAML with Salesforce as SP https://help.salesforce.com/s/articleView?id=sf.sso_saml_setting_up.htm&type=5.
[2] Configuring Salesforce as IDP - https://help.salesforce.com/s/articleView?id=sf.sso_saml.htm&type=5
[3] SF as openID provider https://help.salesforce.com/s/articleView?id=sf.service_provider_define_oid.htm&type=5
[4] SF as relying party https://help.salesforce.com/s/articleView?id=sf.sso_authentication_providers.htm&type=5