

In a modern authentication and authorization system, the process typically involves an **Identity Provider (IdP)**, a **Service Provider (SP)** (often referred to as the Application), an **Identity Store**, an **Application Data Store**, and a **Client/User Agent**. Here’s how they fit together:

**Key Components**

**Client/User Agent**: This refers to the entity (e.g., a web browser or mobile app) that the end-user interacts with. The client sends requests to the application on behalf of the user.

**Identity Provider (IdP)**: This is a trusted service that performs the authentication of the user. It holds the user’s credentials (password, certificates, etc.) and manages user identity, often issuing authentication tokens.

**Identity Store**: This is a repository (like a database or directory) that holds the user's identity information (e.g., usernames, passwords, and other attributes). The Identity Provider uses this store to validate credentials.

**Service Provider (SP)** or Application: This is the application or service the user is trying to access. The SP relies on the Identity Provider to handle authentication and authorize the user’s access to resources.

**Application Data Store**: This holds the actual application’s data, such as user-generated content, profiles, transactions, etc. The SP determines which users can access which data based on the authorization process.

**Process of Authentication and Authorization:**

**1. User Requests Access to the Application (SP):**

The **client/user agent** (browser, mobile app) attempts to access a protected resource in the **Service Provider** (the application).

**2. Redirect to Identity Provider (Authentication):**

The **SP** detects that the user is not authenticated and redirects the client to the **Identity Provider (IdP)**.

The **IdP** prompts the user for credentials (username and password, for example) and authenticates them against the **Identity Store**.

**3. Authentication by Identity Provider:**

The **IdP** checks the credentials against the **Identity Store** (e.g., LDAP, Active Directory, or database).

If the credentials are valid, the IdP creates a secure **token** (such as JWT, SAML assertion, or OAuth token) which asserts the user’s identity.

**4. Redirect Back to the Service Provider (Authorization):**

The **IdP** redirects the client back to the **SP** with the token.

The **SP** verifies the token, extracts user information, and checks what resources the user is authorized to access.

**5. Authorization Decision by Service Provider:**

The **SP** checks the user’s roles/permissions (which may have been included in the token or queried from the **Identity Provider**).

Based on this information, the **SP** authorizes the user to access specific resources in the **Application Data Store**.

**6. Access Granted:**

If the authorization is successful, the **SP** grants the user access to the requested resources.

The **Application Data Store** serves the requested data to the user through the **SP**.

**Common Protocols Used:**

**OAuth 2.0**: Used for authorization. It issues tokens that allow the client to access resources on behalf of the user.

**OpenID Connect (OIDC)**: A layer on top of OAuth 2.0, used for authentication and to retrieve user profile information from the IdP.

**SAML (Security Assertion Markup Language)**: A protocol used for Single Sign-On (SSO) between the **IdP** and the **SP**.

**Example Flow:**

**Client** (browser) tries to access a **Service Provider** (web app).

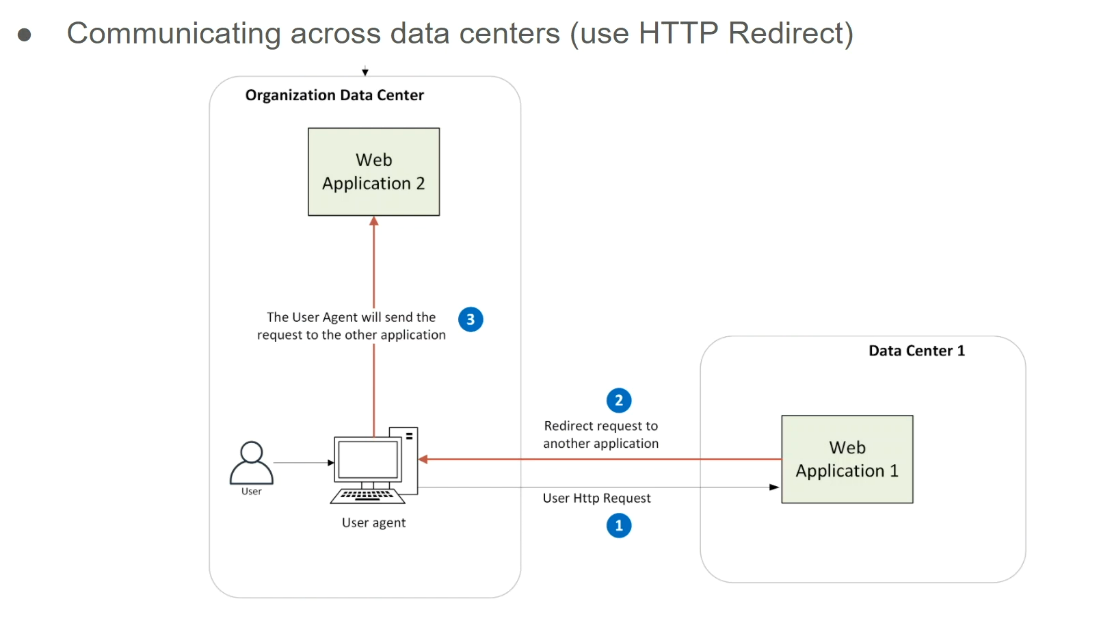
The **SP** redirects to the **Identity Provider** (e.g., Google, Okta) for authentication.

The **IdP** validates the credentials against the **Identity Store** (e.g., LDAP) and issues an authentication **token** (e.g., JWT).

The **Client** is redirected back to the **SP** with the **token**.

The **SP** validates the token, checks permissions, and fetches data from the **Application Data Store** (e.g., user profile, transactions) if authorized.

This process ensures secure and scalable authentication and authorization, often used in Single Sign-On (SSO) environments.

SAML and Single Sign On  
  


LDAP had an advantage of centralizing the users in one place, however it had two significant disadvantages:

1. the LDAP had to be in the same data centre as the application. this is a significant disadvantage for cloud applications.

2. users had to enter their credentials manually. this can be a security issue

but let us see how SAML addresses both these problems.

Technique that is used by SAML to perform Authentication across data centres.

So, you have a web application one which is in a completely different data centre and web

application two which is in the organization data centre in the organization.

Some application from outside cannot call web application 2 so **first problem** how does the web application one in some way communicate with web application 2 at least send some requests to it?

There is no direct path from data centre one to data centre two (organization data centre).

The answer to that lies in the HTTP redirect.

It makes use of the browser to communicate web application one and web application two

together. in a sense what happens is that:

Let us see the user sends a request to web application one so it is an HTTP request on the wire sending to web application 1, so web application 1 wants to send something to web application 2 which is in a different data centre, it will simply request the user agent which is the browser to redirect the request to web application 2 and, in the process, also add some more data to it. so that is what those red lines represent.

It is a redirect request by the web application 1 to forward it to web application 2, so 2 and

3 basically represents the **redirect mechanism**, now this redirection can happen many times, for example web application 1 might redirect to web application 2 and web application 2 can redirect back within its same application to some other process and finally when web application 2 is ready to send the data back to web application 1, it does not make a direct call to web application 1.

So, what it does it again does a redirect so it will ask the user agent again to redirect the

request back to web application one. so that is how the communication between two entities which is in this case two HTTP servers happen across data centres.

Now let us look at how SAML solves the **second problem** which is the user entering the credentials manually how can we avoid this.

In a typical enterprise an employee who is logged into the corporate network using his laptop is already in the LDAP. in most cases this is the Active Directory.

SAML Identity Provider leverages this fact so instead of asking the user to enter the login

credentials - SAML Identity Provider recognizes the fact that the request is being sent by the same

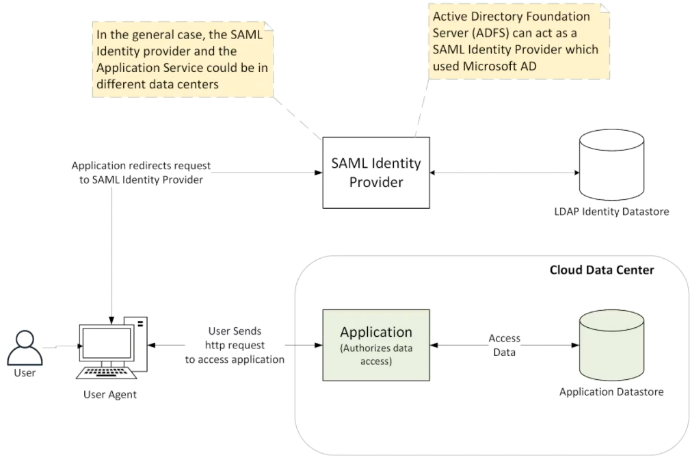
user who has logged into the corporate network. SAML then passes back the user information to the

application without asking for the credentials. This is great because the user now does not

have to enter the credentials and still is able to log into the application. This is a significant

improvement in user experience.

SAML Single Sign-On Flow



Look at the flow of the request between the user the Identity Provider and the application provider.

You will see the application and the application data store in a separate cloud data centre.

There is no direct connection between the application and the SAML Identity Provider.

In most cases the user agent the SAML Identity Provider the Identity data store would be in the same corporate data centre.

User here could be an internal user of the organization, so when if you look at it from

the Identity perspective or the Authentication perspective or Authorization the Identity of the

user is stored in the LDAP Identity store.

Then you have the SAML Identity Provider which is nothing but ADFS from Microsoft which is built on top of the Active Directory.

The user agent is a browser (Google chrome, Firefox, Internet explorer) but it is a browser which can redirect requests from one application to another.

In this case from the application to the SAML Identity Provider that is how it gets

around the restriction of application not able to directly talk to the SAML Identity Provider.

Authentication is done by SAML Identity provider in this case ADFS

and

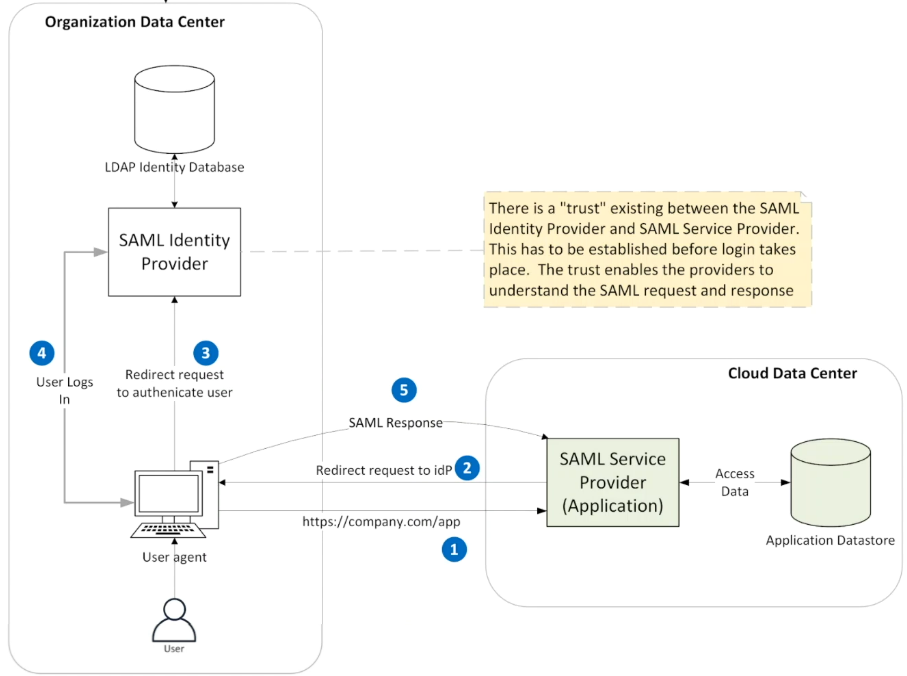
Authorization as always is done by the application because it is the only entity which knows

how to authorize and control different functions within the application to users.

In the case of LDAP architecture when an application talked to LDAP it would get the

LDAP groups and then it would do the Authorization.

For the SAML architecture the LDAP groups will come back via the user agent, it will come to the application and then in a similar way it can do Authorization within its application.



Now you have the SAML Service Provider which is nothing but the application and we will soon see why this is called as the Service Provider and the application data store all within the same cloud data centre which is separate than the organization.

Application does not have a direct connection to the SAML Identity Provider, so it has to use the HTTP redirect in order to talk to it.

Assume that the user here is now going to send an initial request to the application, so it could be company.com/app whatever that url is you are sending the request to the application.

The application at this point says you were never logged in I do not know who you are so I am going to send back this request to the SAML Identity Provider via the user agent.

**Before this whole thing takes place there must be a trust which has to be established between the**

**SAML Identity Provider and the SAML service provider and the part of the trust is an exchange**

**of SAML metadata.**

**If the application request is coming from a particular URL it is mapped to an Identity Provider**, so in this case the application would redirect it to the SAML Identity Provider so that is marked by request number 2 and number 3 that represents the redirection and this redirection is happening in two steps.

The application is sending a response 2 to request 1 as a request 3 to the browser to redirect to the SAML Identity Provider the request is initiated from the browser to the SAML Identity Provider sending it the SAML request. So as a part of 2 the application is also sending the SAML request attaching that information to the SAML request and saying send it to the SAML Identity Provider so the browser as we know before is going to send a HTTP request to the SAML Identity Provider using the SAML request.

**SAML Request is a request to get the user information from SAML Identity Provider (No Password sent)**

When the request 3 comes to the SAML Identity provider as we had talked before it

recognizes the fact that it is coming from a user who is already logged into the network and because

of that what it does it automatically logs in the user 4 and it sends a redirect response back to

the SAML Service Provider.

So, this is not a print in the picture but the SAML Identity Provider will talk back to the application again using a redirect so it will be a redirect back to the application and as a part of the response the user information is also sent 5 and they are called as **claims.**   
  
The attributes of the users are sent as SAML claims and entire thing is bundled as a sample response and within the SAML response is also contained the SAML token.

Within the SAML token is information about the user.

SAML response is encrypted and is also signed by the Identity Provider but to decrypt or to recognize the signature the SAML Service Provider needs the certificates of the SAML Identity Provider.

In the same way the SAML Identity Provider needs the certificate of the SAML Service Provider. this is what is called as the **trust**.

So the SAML Identity Provider and the Service Provider needs to know each other's information and that's the trust which is an exchange of the SAML metadata which basically is an xml file containing the certificates which needs to be exchanged between the two so that those two entities know about each other.

That is the reason why this application is called as the SAML Service Provider because it knows about the SAML Identity Provider and that is the information about it and in the same way the SAML Identity Provider knows about the application which is going to send it the Authentication request and because of that SAML metadata the application can take that response, decrypt it and

recognize the signature or validate the signature.

Only thing I need to we did talk about the SAML metadata file, the trust between the SAML Identity Provider and the Service Provider and the **SAML response contain SAML token** and the **SAML token contains claims**.

The **claims are attributes of the user** so the first name, the last name, email id all of those and the groups.

All of those are considered as SAML claims and the **SAML token and the SAML claims are encrypted**

for security reasons.

The user is **federated user** meaning that you have one Identity in one place

inside the organization and all the applications in other data centres can use the same Identity

for all these applications and that is one way of saying it is nothing but a federated

user.

It is the same user Identity which has been used by all these.