Technical Documentation for

Directory Management with OpenLDAP, AD and SSO Integration with ADFS

Prasanth.V

01/01/2025

TABLE OF CONTENTS

GLOSSARY

AD	Active Directory
SSO	Single Sign On
ADFS	Active Directory Federation Services
RBAC	Role Based Access Control
LDAP	Light Weight Directory Access Protocol
SP	Service Provider
IDP	Identity Provider
SAML	Security Assertion Markup Language
OU	Organizational Unit
CN	Common Name
DN	Distinguish Name
UI	User Interface
RP	Relay Party
EJS	Embedded JavaScript Templates
MVC	Model View Controller

INTRODUCTION

This technical document provides a comprehensive **overview of the architecture, integration, and configuration of the platform** that integrates OpenLDAP, Active Directory (AD), and Active Directory Federation Services (ADFS) for centralized directory management and authentication.

The system leverages SSO (Single Sign-On) to provide seamless user access across multiple services and applications.

It covers the technical details of the architecture, key components, and integration points between OpenLDAP, AD, and ADFS, along with the necessary configurations for successful SSO implementation.

This guide includes:

- Detailed explanations of the underlying technologies and LDAP protocol-based application.
- Configuration steps for integrating OpenLDAP and Active Directory with ADFS
- Role-based access control (RBAC) implementation for user actions.

USE CASE

1. Simplified Management

- By centralized access to user accounts, OU and groups admin can able to efficiently access and manage directory.

2. Enhance Security

- Ensures users can access resources only when they are authorized through RBAC.

3. Multiple Authentication Experience

- Enable users to login in both traditional authentication and SSO based authentication for seamless login experience.

- For SSO, ADFS provides a claim contains the user employee ID, which is validated by system across directory and validating that employee ID with user data present in the directory service and proceed for appropriate dashboard.

4. Scalability

- Works seamlessly with multiple LDAP based systems like OpenLDAP, AD & ADFS.

SYSTEM ARCHITECTURE

The Model-View-Controller (MVC) architecture is used in this project to maintain a clear separation of concerns, enabling a well-organized, modular, and maintainable structure.

Here's how the different layers of MVC are implemented in the project:

1. Model:

- The model represents the data and business logic of the application. In this project, the **Model** layer is represented by the **services** and **utils** directories. These are responsible for interacting with external servers like LDAP or AD (Databases) and performing the necessary operations like user, group management.
- **OpenLDAP Services:** The services in this directory handle business logic related to user, group, and organization management specifically for OpenLDAP. These services connect to the OpenLDAP server, validate data, and perform operations like creating, updating, or deleting users.
- **Active Directory Services:** Similar to OpenLDAP services, these handle Active Directory-specific business logic for managing users, groups, and organizations within an AD environment.

2. View:

- The **View** represents the user interface of the application. It is responsible for presenting the data and providing an interactive experience to the user. In this project, the **views** are created using **EJS**.
- **Static Assets**: The UI folder contains static files like CSS, JavaScript, and images, which are used to style the templates and enable frontend interactions.

3. Controller:

- The **Controller** layer acts as an intermediary between the **Model** and the **View**, processing user requests, interacting with the model, and rendering the appropriate views.

- **OpenLDAP Controllers**: These controllers define the logic for handling requests related to OpenLDAP operations. For example, userController.js manages user-specific actions like creating, updating, and listing users.
- **AD Controllers**: Similar to OpenLDAP controllers, these controllers handle Active Directory-specific operations for users, groups, and organizations.

While the core structure of the project follows the **MVC** design pattern, additional components like **Middleware** and **Routing** play important roles in ensuring the overall functionality of the application. These components enhance the maintainability of the codebase, allowing for more efficient request handling and ensuring proper file structuring.

4. Middleware

- The **Middleware** layer is responsible for handling requests before they reach the controller. It’s used for tasks like authentication, session management, rate-limiting, and error handling.
- **Session Middleware:** This middleware manages user sessions and ensures that the user is authenticated before accessing protected routes.
- **Error Middleware:** This handles centralized error management, logging errors, and providing a unified error response to the user.
- **API Limiter:** This middleware helps in limiting the number of API requests a user can make, preventing abuse and overloading of the system.

5. Routing

- Routing in this project defines how the HTTP requests are handled and mapped to controllers for specific actions.
- **User Routes**: These routes handle the operations related to user management in OpenLDAP and Active Directory.
- **Group Routes:** These routes manage group operations such as creating groups, adding users to groups, etc.

- **Common Routes:** This includes routes that can be used across the system, like session management and general API endpoints.

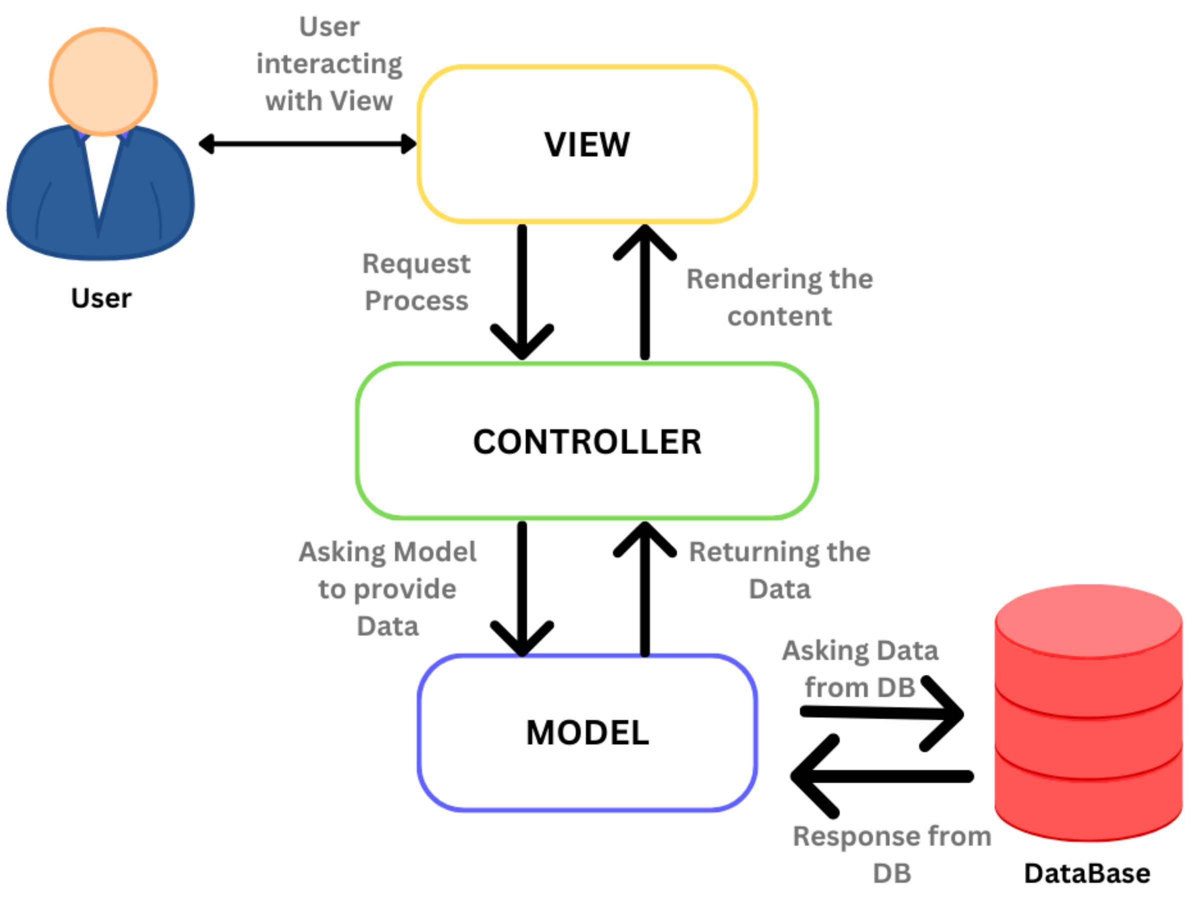


Figure 1: MVC Structure

ROLE-BASED ACCESS CONTROL

RBAC is the key feature to implemented in our project. Each role is assigned to set of actions to be performed.

1. Admin

- Admin user has full access and responsible for managing the directory service.
- Permissions:
  - Create, update, delete user accounts.
  - Manage permission to users.

    - Create & manage group members.
    - Reset user password.

2. User

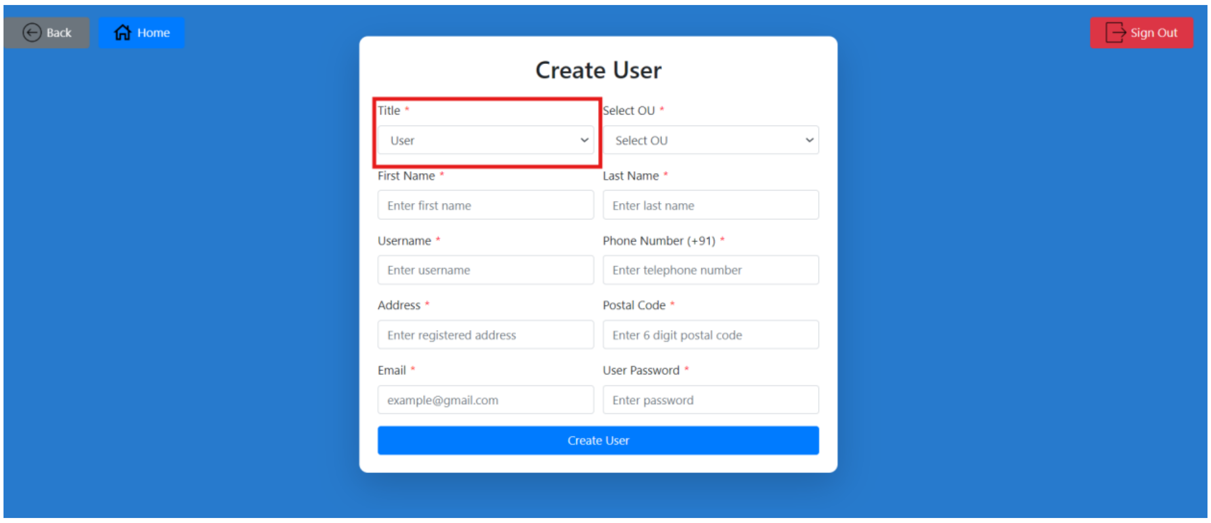
- Regular user has limited access and can perform only related to their own account and general queries.
- Permissions:
    - Change Password.
    - Search users within the directory.

ROLE PERMISSION MAPPING

The process of mapping roles to users are unique method across respective LDAP applications.

1. OpenLDAP

- When creating users in OpenLDAP, admin need to mention Title field (default is set to user).
- After creating of users, the title is used for role against OpenLDAP which is also been validated while authentication (Figure2).



Create User

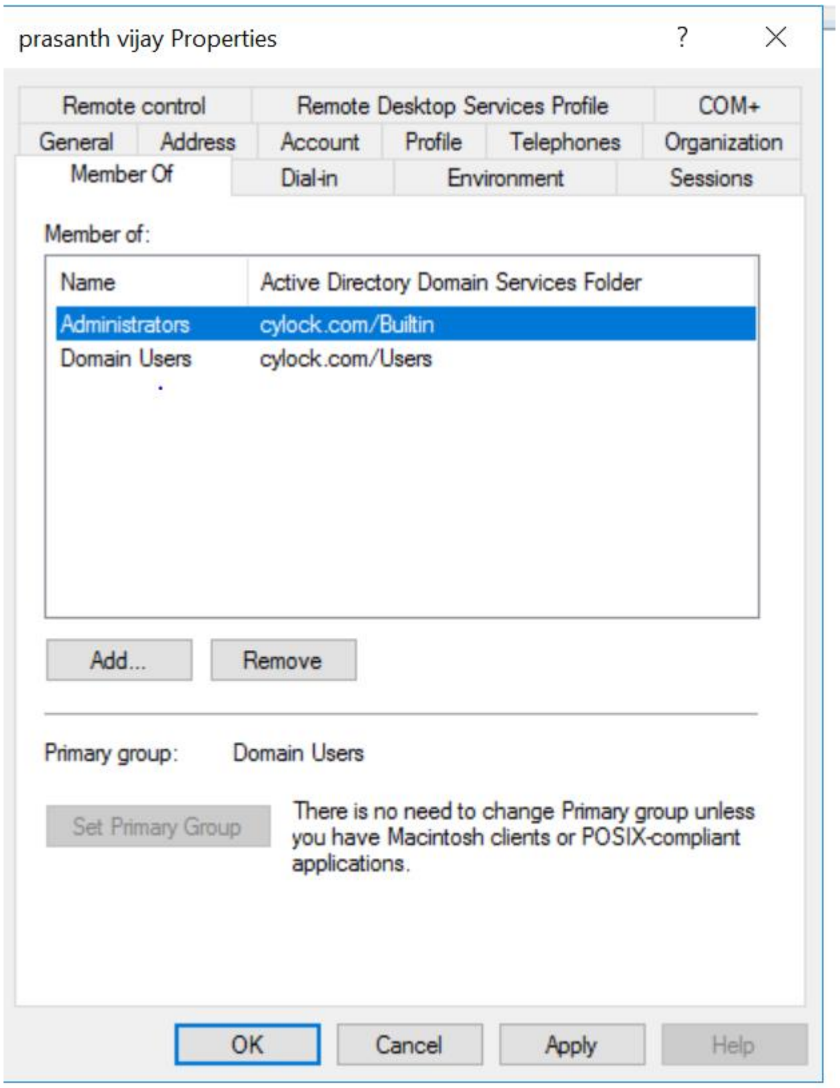
Title *	Select OU *
User	Select OU
First Name *	Last Name *
Enter first name	Enter last name
Username *	Phone Number (+91) *
Enter username	Enter telephone number
Address *	Postal Code *
Enter registered address	Enter 6 digit postal code
Email *	User Password *
example@gmail.com	Enter password

Create User

Figure 2: Role Mapping in OpenLDAP

2. AD

- When creating users in AD, all new users are default set to user (non admin). The admin role access can be gained only by adding to admin type of groups.
- While authentication with the necessary details against AD server, code flow will check the user is member of any admin groups (group details are mentioned in backend).
- If any of the group matches with the authenticated user, they will be accessed to admin privileges.



Member of:

Name	Active Directory Domain Services Folder
Administrators	cylock.com/Builtin
Domain Users	cylock.com/Users

Primary group: Domain Users

There is no need to change Primary group unless you have Macintosh clients or POSIX-compliant applications.

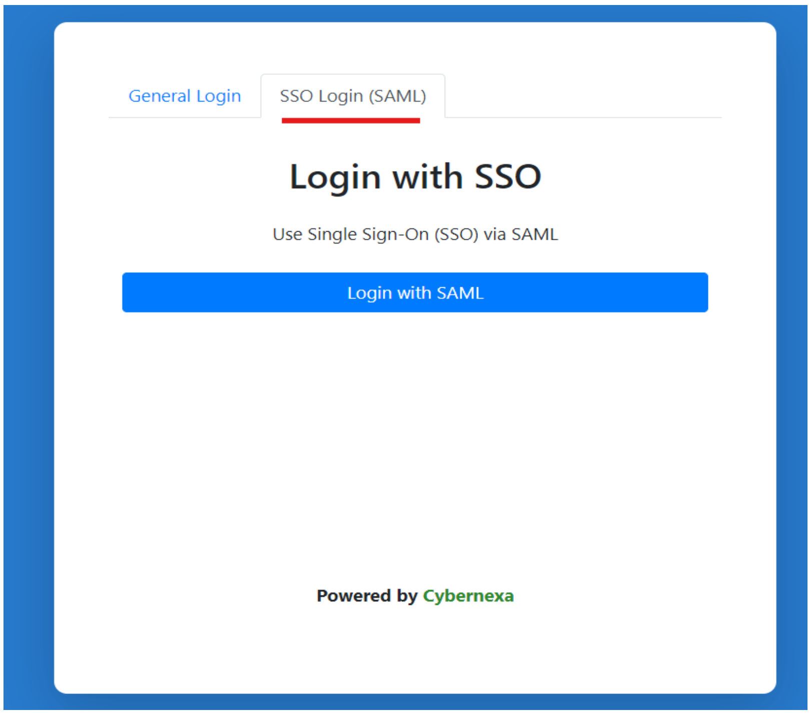
Figure 3: Role Mapping for AD

SINGLE SIGN ON (SSO)

- Single Sign-On (SSO) allows users to authenticate once and access multiple systems or applications without re-entering their credentials. This reduces the complexity of managing multiple passwords and enhances security by centralizing authentication through a trusted Identity Provider (IdP).

SP-initiated Flow

- In an SP-initiated SSO flow, our application acts as the **Service Provider (SP)** that interacts with an Identity Provider (IdP) to authenticate users (Figure 4). Here's how the process works:
- **Authentication Request**
  - The user accesses our application, triggering a **SAML Authentication Request** to the Identity Provider (IdP). The request, signed for security, redirects the user to the IdP via their browser.
- **Response and Validation**
  - After user authentication at the IdP, a **SAML Response** (containing user details and roles) is sent back to our application's **Assertion Consumer Service (ACS)** endpoint. The response is validated for signature, conditions, and integrity.
- **Session and Access**
  - Upon successful validation, a session is created for the user in our application, granting access to resources based on their roles and attributes. All subsequent actions operate within this session.

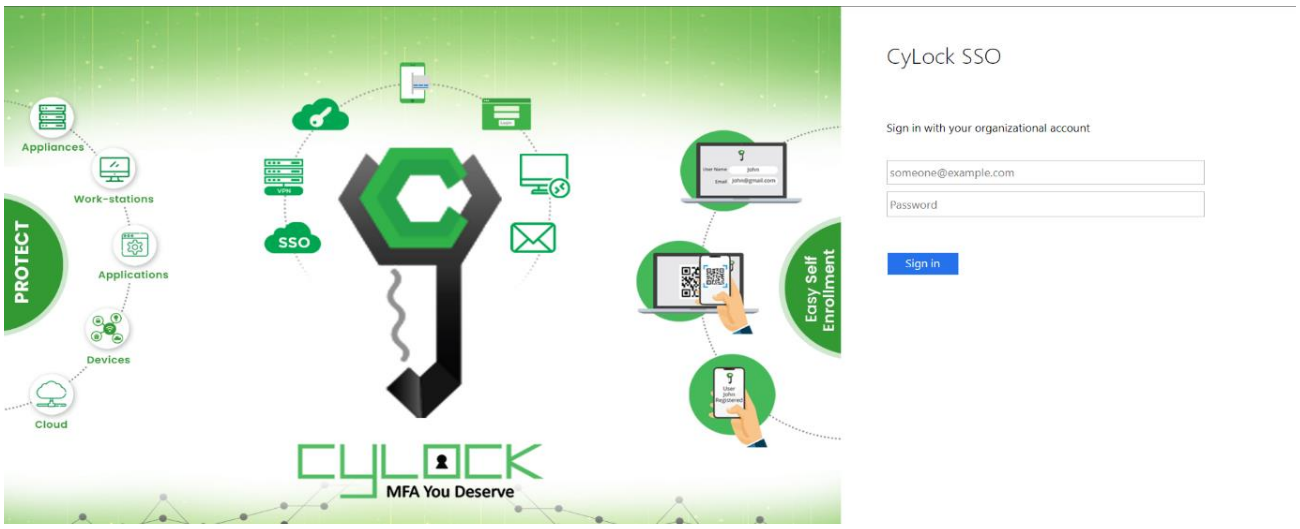


Login with SSO

Use Single Sign-On (SSO) via SAML

Powered by Cybernexa

Figure 4: SSO login page (SP-initiated).



CyLock SSO

Sign in with your organizational account

Figure 5: ADFS Login page

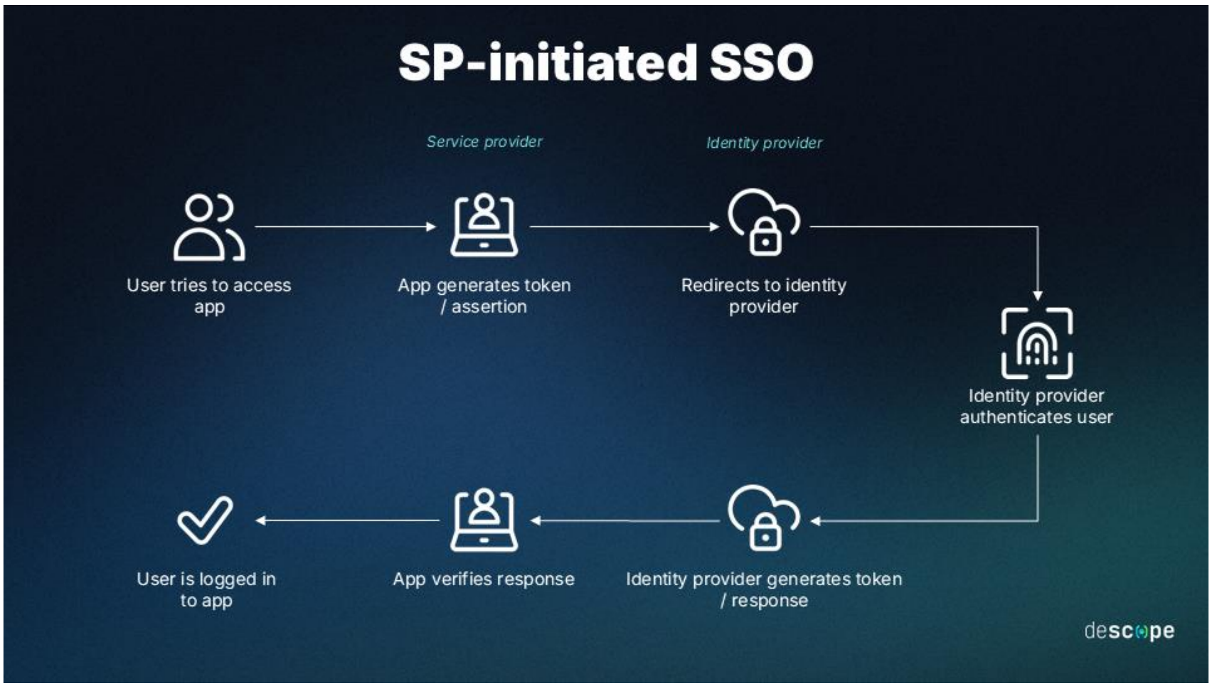
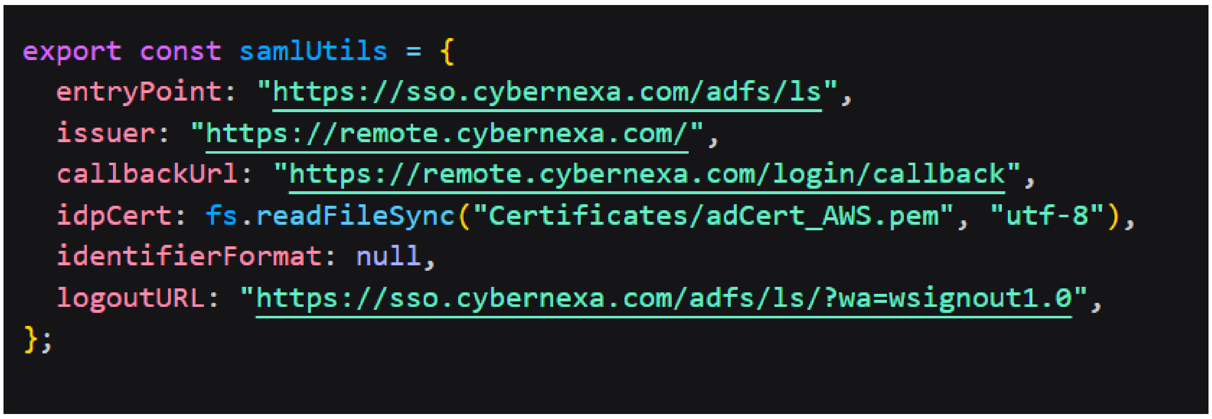

SP-initiated SSO
Service provider
Identity provider
User tries to access app
App generates token / assertion
Redirects to identity provider
Identity provider authenticates user
Identity provider generates token / response
App verifies response
User is logged in to app
descope

Figure 6: SP- initiated SSO

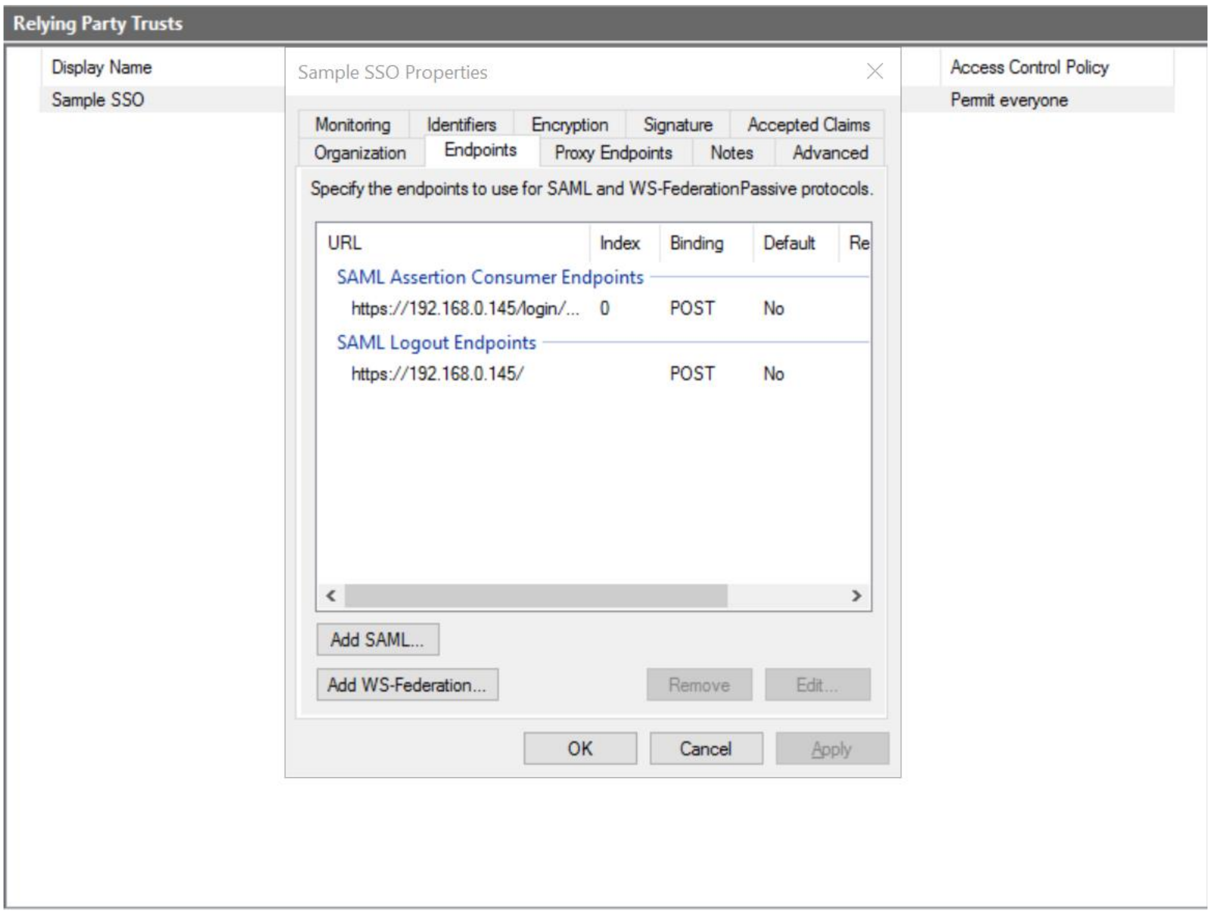
SSO Connection Establishment

- In the context of SAML, Relying Party (RP) is used to create connection establishment across SP to IdP.
- The common fields to be matched in both SP and IdP to create connection for authentication.



```
export const samlUtils = {
  entryPoint: "https://sso.cybernexa.com/adfs/ls",
  issuer: "https://remote.cybernexa.com/",
  callbackUrl: "https://remote.cybernexa.com/login/callback",
  idpCert: fs.readFileSync("Certificates/adCert_AWS.pem", "utf-8"),
  identifierFormat: null,
  logoutURL: "https://sso.cybernexa.com/adfs/ls/?wa=wsignout1.0",
};
```

Figure 7: SAML RP setup for SP - initiated



Relying Party Trusts

Display Name	Access Control Policy
Sample SSO	Permit everyone

Sample SSO Properties

Specify the endpoints to use for SAML and WS-FederationPassive protocols.

URL	Index	Binding	Default	Re
SAML Assertion Consumer Endpoints				
https://192.168.0.145/login/...	0	POST	No	
SAML Logout Endpoints				
https://192.168.0.145/		POST	No	

Figure 8: Setup in IdP.