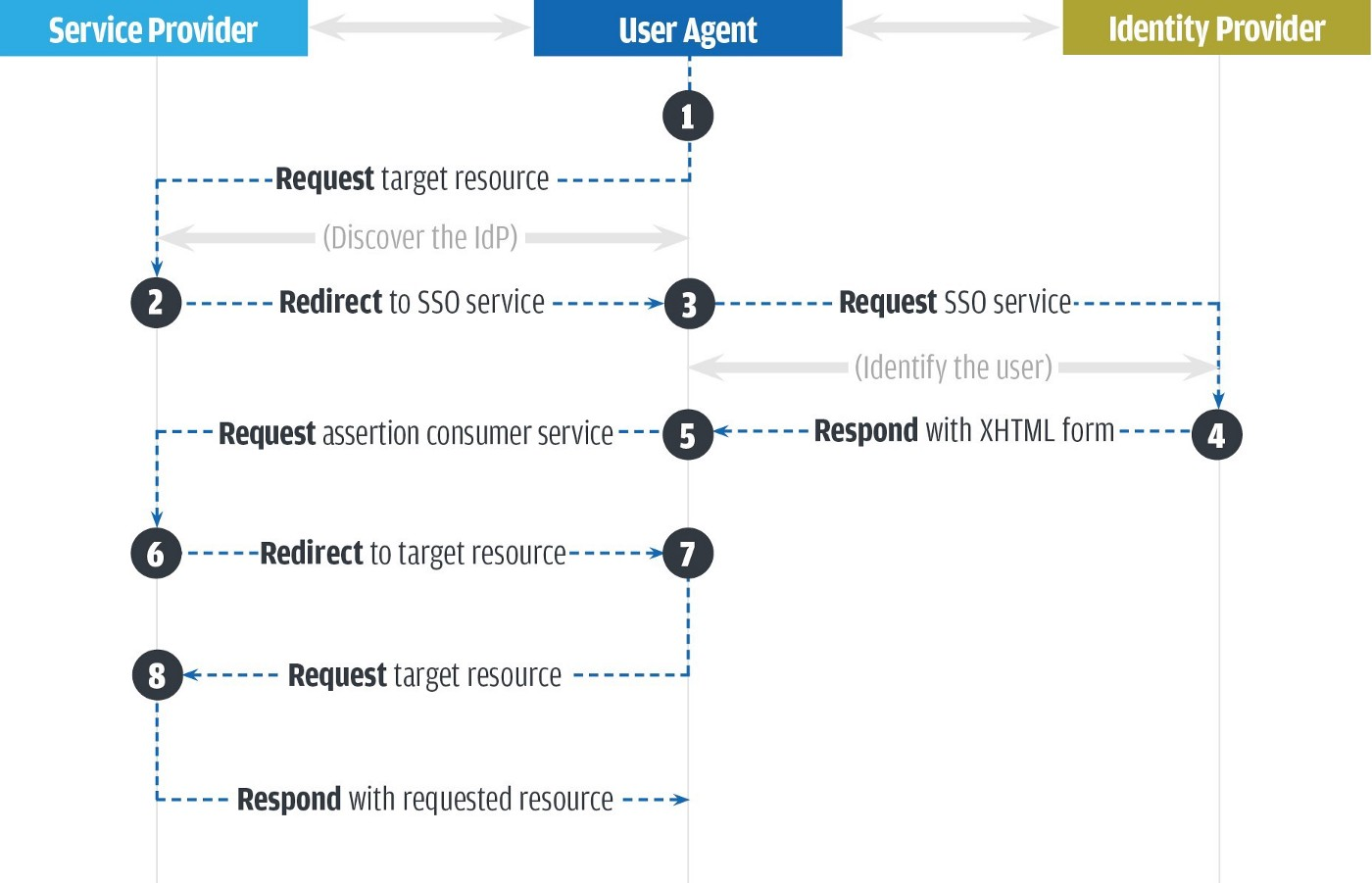
### SAML

Security Assertion Markup Language (SAML) is an XML-based open standard used for single sign on (SSO) implementations. SAML 2.0 was released in 2005 and is the current version of the standard.

SAML is used for both authentication & authorization between two parties: a Service Provider (Office365, Salesforce, G Suite, etc.) & an Identity Provider (Okta, OneLogin, Ping Identity, etc.). The Service Provider (SP) agrees to trust the Identity Provider (IdP) in the authentication process. This is done through a SAML XML document sent by the IdP containing the user authorization & authentication and then redirected to the service provider.



Let’s consider this example:

* Identity Provider (IdP): Active Directory Federation Services
* Service Provider (SP): EquityClear application

1. User tries to login to EquityClear application from Web browser.
2. EquityClear application responds by generating a SAML request
3. Web browser redirects the user to an SSO URL, Active Directory Federation Services parses the SAML request, authenticates the user and generates a SAML response.
4. Active Directory Federation Services resends the encoded SAML response to the Web browser
5. Web browser redirects the SAML response to the EquityClear application
6. If the verification is successful, the user will be logged in to the EquityClear application and granted access to all the various resources.

## Standards comparison

|  |  |  |  |
| --- | --- | --- | --- |
|  | **OpenID Connect** | **OAuth2** | **SAML 2.0** |
| **What is it?** | Open standard for authentication | Open standard for authorization and pseudo- authentication | Open standard for authorization and authentication |
| **History** | Developed by the OpenID Foundation in 2014 | Developed by Twitter and Google in 2006 | Developed by OASIS in 2001 |
| **Primary use case** | Single Sign-On for Consumers apps | API Authorization Between Applications | Single Sign-On for Enterprise apps |
| **Protocols Used** | XRDS, HTTP | JSON, HTTP | SAM, XML, HTTP, SOAP |
| **Format** | JSON | JSON | XML |
| **Security Risks** | Phishing  Identity providers have a log of OpenID logins, making a compromised account a bigger privacy breach | Phishing  OAuth 2.0 does not support signature, encryption, channel binding, or client verification. Instead, it relies completely on TLS for confidentiality. | XML Signature Wrapping to impersonate any user |
| **Support client** | Web browser based, native applications, sensor | Web browser based, native applications | Web browser based |
| **Ease of implementation** | Harder implementation | Easier implementation | Harder implementation  from a technical perspective, an XML-based SAML is very complex. |
| **Interoperability** | Supported by ADFS (latest version) | Supported by ADFS (latest version) | Supported by ADFS |
| **Discovery service** | Single discovery | Single discovery | No (requires pre-agreed metadata), does not support API management scenarios (discoverability/client registration) and heavy weight tokens |
| **Infrastructure and dependency** | No infrastructure | No infrastructure | Requires infrastructure.  There is a dependency on the identity provider, so that access to the service provider is not possible without the availability of the identity providers. |
| **Signing** | Yes (JWT) | Yes | Yes |
| **Privacy** | Yes (JOSE) | Yes | Yes |
| **Advantages** | *Built on top of OAuth 2.0 standard*  *Suitable for mobile devices (and possibly for sensor devices).*  *OpenID Connect tokens are designed for today’s REST-based application development practices.*  *Simpler protocol to realise (great support for open libraries as well). Potentially safer (XML-DSIG changed to JWS what eliminates a range of possible attacks).*  *Foundation for a far more efficient and scalable enterprise federated SSO solution.*  *Compare to SAML: OpenID Connect can satisfy these same use cases but with a simpler, JSON/REST based protocol.*  *Allows to choose of trust provider.* | *Open standard for Authorization, i.e., publicly available, and developed, approved and maintained via a collaborative and consensus driven process.*  *The only framework in its genre and is widely used for similar applications.*  *More like a framework (not a defined protocol), which leaves a lot of implementation freedom that we need because of non-standard requirements. Easy to configure and deploy which saves time, effort and ease development process.*  *Organisations do not need support for password renewal, forgotten password, authentication of users, and support to let users remove themselves from the service, etc.*  *Low risk for ID theft, etc. The service already has good support to prevent this. Authentication takes place at provider; the OAuth tokens are encrypted and not in our application.*  *Gives great possibility to add new services (many well-known industry and education organisations already deployed OAuth 2.0 which allows clients to use resources using their identity).*  *User can prevent access to the application from the OAuth provider* | *SAML tokens provide information about identity of entity and widely used in SSO systems.*  *Dominant protocol for achieving secure attribute exchange and SSO today.*  *Mature and well-defined standard.*  *Widely adopted and provides good security based on signed XML documents.* |
| **Disadvantages** | *So far not widely adopted, but the situation is rapidly changing (many companies are currently in a process of adopting it).* | *No notion of identity, therefore additional effort is needed to validate an identity of requesting parties.*  *No fine-grained authorization (based on scopes only).*  *Requires logic to allow the user to log in with multiple OAuth providers.* | *Does not support embedded and mobile applications.* |

**Poc SAML 2.0/AD FS:**

1. Access request to UCS Or SPH.
2. The SP (SAML FILTER) intercepts access request, case the user has not been authenticated, create a SAML 2.0 authentication request.
3. Send a HTTPS SAML 2.0 request to AD FS IdP.
4. Authentication AD with a trust relationship ADFS/AD.
5. Send a HTTPS SAML 2.0 response to SP (SAML FILTER).
6. Check the SAML 2.0 response message (decoding, verify).
7. Create an authentication token (Shiro) containing the SAML2 response received on the Service Provider Consumer URL configured (/acs): ACS Assertion Consumer Service.
8. Populate authorization realms with roles/features from LAMADM database. The concept of role-based access control (RBAC) will be used, refers to the idea of assigning permissions to users based on their role (token Shiro). It provides fine-grained control.
9. Starting and store SAML session and redirect to initial requested resource.
10. For all access request to protected resources for example with prefix (e.g. /api/\*), the Shiro filter check permission and redirect to requested resource.



**10**

**1**

SPH

UCS

LAMADM

**SHIRO**

**FILTER**

**SAML**

**FILTER**

**8**

**7**

**10**

**2**

**10**

**6**

CW\_ADM\_ROLE

user has already been authenticated

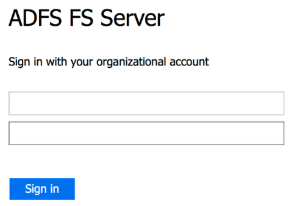
user has not been authenticated

**5**

**3**

**9**

CW\_ADM\_ROLE\_FEAT

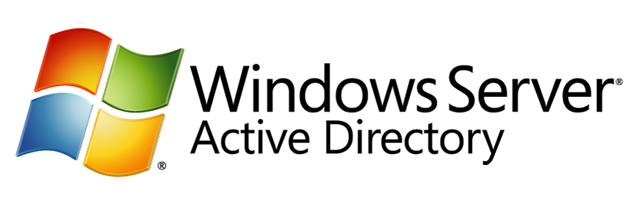


access to a resource



CW\_ADM\_FEAT

**4**



|  |  |
| --- | --- |
| Project:  Owner:  Author:  Document Type:  Version Number:  Date: | SSO Server (Identity gateway with Oauth 2.0/LDAP)  …….  Developer Team  TAD/POC (local)  0.1 (Draft)  2020/06/03 |

\_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_ \_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_ \_\_\_\_

--------------------------------------------------------------------------

SSO-SERVER Version: 1.0

--------------------------------------------------------------------------

# Architecture - an overview

Apache OLTU

Spring (ldap, data-ldap)

**ACTIVE DIRECTORY**

LDAP

**CACHE MANAGER**

**SSO-SERVER**

OAuth2

Apache SHIRO

Search, Authenticate

(spring, ehcache)

*Audit Log*

*OAuth2 Client*

**Register as a client**

For retrieve data from a resource server using OAuth2, you must register as a client of the authorization server.

Therefore, a relational database will be used:

* *Client name*: the application name
* *Client id: the application id*
* *Client secret: the application secret*
* *Redirect URLs: URLs of the client for receiving* authorization code and access token

💡 Classic CRUD via an IHM apply to this topic

**Authorization grant types**

💡 OAuth2 defines 4 grant types depending on the nature of the client:

* **Code Grant(🗸)**

It allows you to obtain a long-lived access token since it can be renewed with a refresh token (if the authorization server enables it).

* **Implicit Grant** **(X)**

It is typically used when the client is running in a browser using a scripting language such as Javascript.

* **Resource Owner Password Credentials Grant** **(X)**

the credentials (and thus the password) are sent to the client and then to the authorization server. It is mainly used when the client has been developed by the same authority as the authorization server.

* **Client Credentials Grant** **(X)**

This type of authorization is used when the client is himself the resource owner. There is no authorization to obtain from the end-user.

## Technologies

*We have several models of architectures (3-tier, Struts MVC, GWT,...) and also the solution will be designed to facilitate the integration of old/new applications, and high availability.*

{ Java, Spring Boot (web, cache, tomcat), Spring (ldap, data-ldap), Apache shiro (spring, ehcache), JSTL, Spring, Tomcat embed, Apache oltu, Maven, Logback, Page login/Admin (jsp, html, css), Yml }

Git :

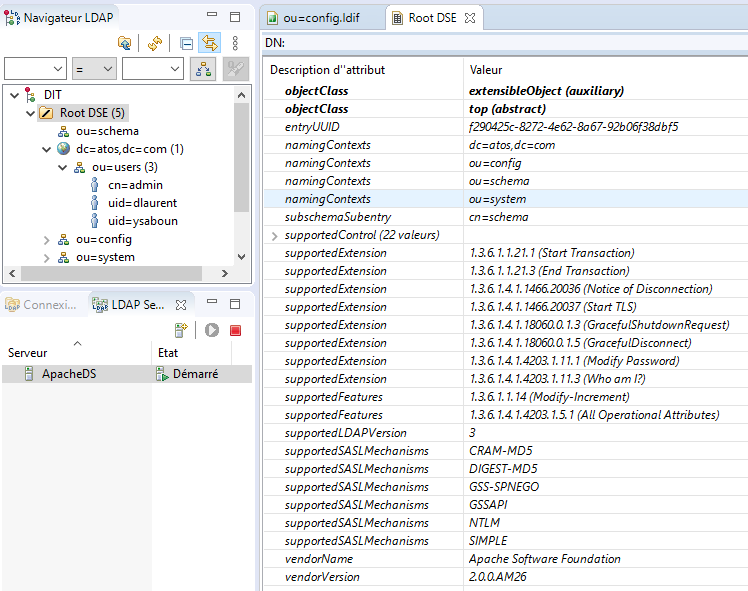
**User Story Mapping**

# DONE

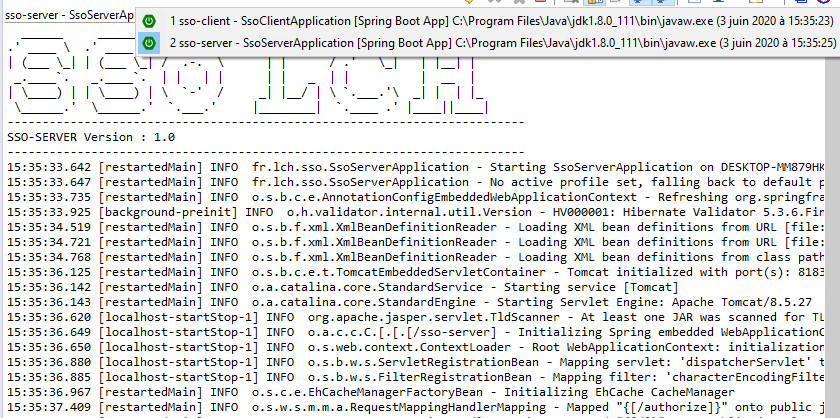
1. Poc Keycloak (integration a plugin Logging event listener), high cost
2. Poc sso-server core (test Oauth 2.0)
3. Poc sso-server core (test LDAP ApacheDS)

**LDAP**: ou=users,dc=atos,dc=com

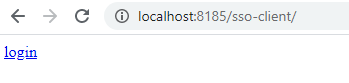
LDAP (ApacheDS)



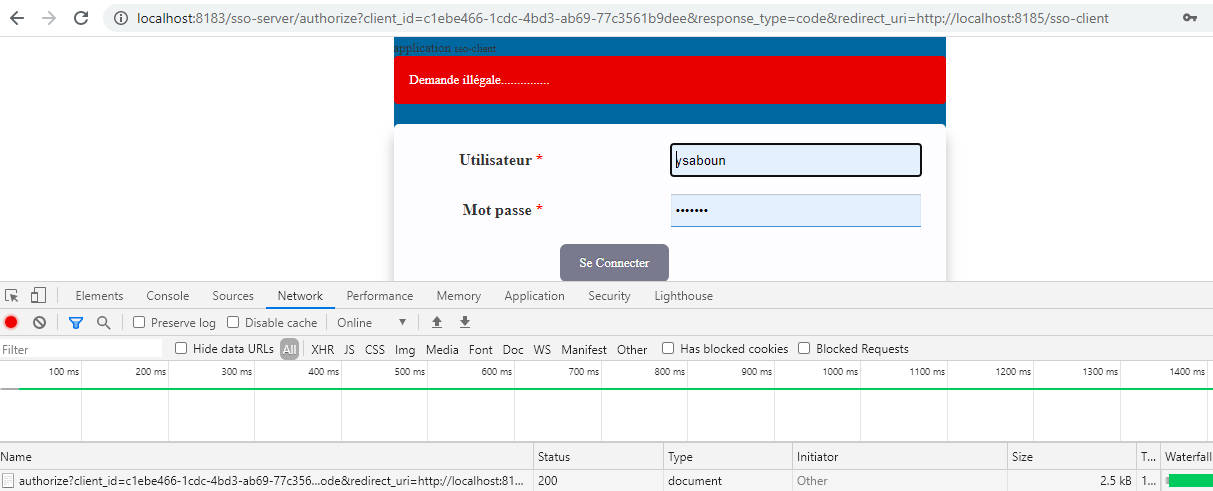
Run sso-server

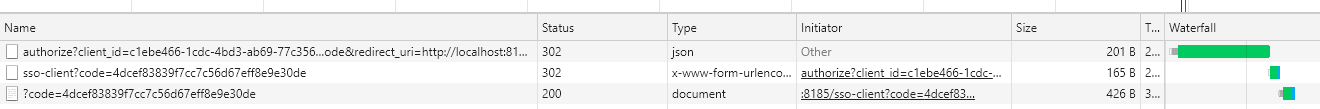


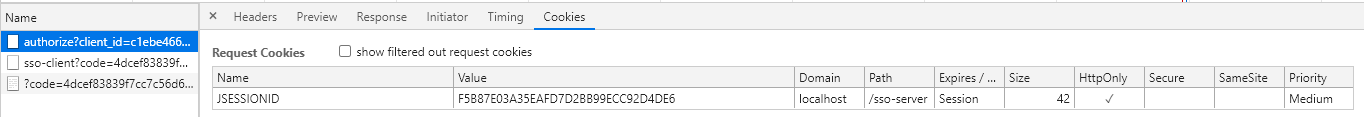
Test login



*Redirect URL to sso-server (login page)*







**General**

Request URL: <http://localhost:8183/sso-server/authorize?client_id=c1ebe466-1cdc-4bd3-ab69-77c3561b9dee&response_type=code&redirect_uri=http://localhost:8185/sso-client>

Request Method: POST

Status Code: 302

Remote Address: [::1]:8183

Referrer Policy: no-referrer-when-downgrade

**Response Headers**

Content-Length: 0

Content-Type: application/json;charset=utf-8

Date: Wed, 03 Jun 2020 12:56:10 GMT

Location: http://localhost:8185/sso-client?code=4dcef83839f7cc7c56d67eff8e9e30de

**Request Headers**

Accept:text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.9

Accept-Encoding: gzip, deflate, br

Accept-Language: fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7

Cache-Control: max-age=0

Connection: keep-alive

Content-Length: 175

Content-Type: application/x-www-form-urlencoded

Cookie: JSESSIONID=F5B87E03A35EAFD7D2BB99ECC92D4DE6

Host: localhost:8183

Origin: http://localhost:8183

Referer: http://localhost:8183/sso-server/authorize?client\_id=c1ebe466-1cdc-4bd3-ab69-77c3561b9dee&response\_type=code&redirect\_uri=http://localhost:8185/sso-client

Sec-Fetch-Dest: document

Sec-Fetch-Mode: navigate

Sec-Fetch-Site: same-origin

Sec-Fetch-User: ?1

Upgrade-Insecure-Requests: 1

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.61 Safari/537.36

**Query String Parameters**

client\_id: c1ebe466-1cdc-4bd3-ab69-77c3561b9dee

response\_type: code

redirect\_uri: <http://localhost:8185/sso-client>

**Form Data**

client\_id: c1ebe466-1cdc-4bd3-ab69-77c3561b9dee

response\_type: code

redirect\_uri: http://localhost:8185/sso-client

username: ysaboun

password: \*\*\*\*\*\*\*

submit: Se Connecter

***SSO Server***

*Resources (TSI)*

*Web Agents*

***Active Directory***

*Claims Provider (CP)*

***,,***

***ADFS***

*Identity Provicer (IdP)*

***nginx***

***Load Balance***

*SP Federation Server*

nregistrer manuellement l'application ou utiliser le fichier FederationMetadata.xml

configuration demande un nom pour la Relying Party.

protocole SAML 2.0

Il faut ensuite sélectionner un certificat si l'on souhaite encrypter le token renvoyé par ADFS.

enseigner l'URL de l'application/ fédération passive (par redirection vers le STS) nginx

## Configuration des claims (rules

+ Transform an Incoming Claim : transformer un Claim envoyé à l'ADFS (par les Identity Providers) et le renvoyer

+ Send LDAP Attributes as Claims : renvoyer des attributs LDAP en tant que Claims

User Informations), select the attribute store Active Directory

Mapping of ldap attributes to outgoing clain type

+ Send Claims Using a Custom Rule

issue(Type = "access", Value = "true");

**Secure hash algorithm**. You can choose SHA-256 or SHA-1

HTTPS

public certificate, you can download the Federation Metadata at:  
    <https://YOUR_ADFS_SERVER/FederationMetada/2007-06/FederationMetadata.xml>

For the public certificate, you can download the Federation Metadata at: FederationMetadata.xml

You will need to extract from the XML file:Public Certificate: X509Certificate from the ds:Signature section.

Browse to the Endpoints tab and add a **SAML Assertion Consumer** with a **Post** binding and a URL of https://company.service-now.com/navpage.do.

Identifier Entity id

Reply url (addertsion consumer service url)

Identity Provider (IdP)

Service Provider (SP)

**SAML Bindings :**

1. HTTP Redirect Binding
2. HTTP Post Binding
3. HTTP Artifact Binding

ADFS allows federation which comprises two sides viz. the IDP (Claims Provider) (the owner of the identity repository - in this case AD) and the RP (Relying Party) which is another STS or application that wishes to outsource authentication to the IDP.

Trusts are handled via certificates based on the ownership of private keys e.g. SAML tokens are signed by the IDP.

So the RP trust is the trust between the RP and the IDP - a token signed by the IDP must originate from the IDP and therefore the claims inside the token can be trusted.

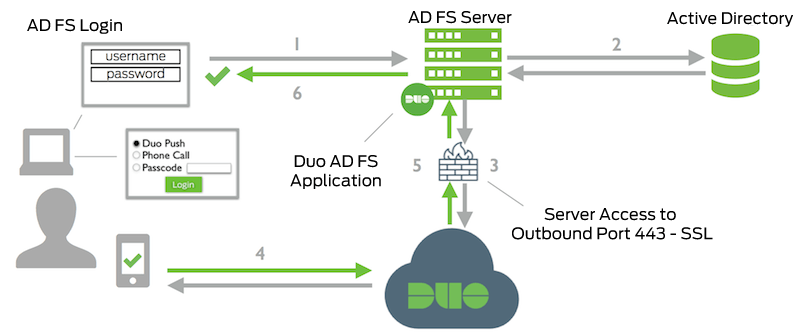
STS (Security Token Service)

# SP Initiated SSO

## Pre-requisites

## ADFS 4.0 in Windows Server® 2012 R2.

<https://blogs.oracle.com/dcarru/sp-vs-idp-initiated-sso>



1. AD FS connection initiated
2. Primary authentication to AD
3. AD FS connection established to Duo Security over TCP port 443
4. Secondary authentication via Duo Security’s service
5. AD FS receives authentication response
6. AD FS session logged in

***AuthN and AuthZ SAML/ADFS:***



***Active Directory***

*Claims Provider (CP)*

***AD FS***

*Identity Provicer (IdP)*



***SAML Proxy***

*Service Provider (SP)*



***SPH***

***TSI***

***UCS***



Trust

*AuthN CFS ~ 42*

*AuthZ CFS ~ 8*

*AuthN CFS ~ 6*

*AuthN CFS ~ 19*

*AuthZ CFS ~ 38*

*SAML/HTTPS*



***Σ CFS (AuthN+AuthZ) ~ 113***

***More details:***



***all application security layers on role/actions, front-end or back-end will not be changed (review and re-certification).***

1. **Scenario n°1 SAML 2.0**
2. **Scenario n°2** **OAuth 2 SAML 2.0 Bearer Assertion Flow**

* Client appliaction (Client Controller) redirects to the SAML server for authentication (SAML request).
* The SAML server checks the user credentials (SAML Controller), creates and signs a SAML Response.
* Request (redirection) to the Authorisation server (OAuth Controller) with the SAML Response in the payload and a state parameter that will be used to access the user’s session.
* The Authorisation server checks SAML Response stores a hash of the SAML response so that it can be used later for verifying the user identity, creates an application cookie identity with the information provided by SAML and redirects the authorisation endpoint. The Authorisation server creates a new bearer type identity for the user with all of the rights that the user authorised and redirects to the Client application (redirect URL) with parameters authentication code and the application state.
* The Client application uses the state parameter to access the user’s session state and issues a POST back to the Authorisation server in order to exchange the authorisation code for an OAuth access token . The request includes the username and SAML hash to be compared as means of authentication.
* The Authorisation server compares the provided SAML hash with the one stored in the cache to authenticate the user and returns an OAuth access token (OAuth Token) to the Client application.
* The Client application stores the OAuth access token in a cookie. For any Request the Client application uses a handler (OAuth Request Handler) and adds the OAuth Token as a URL parameter.

**Autorization**

**Server**

**OAuth 2.0**

**Resource**

**Server**

**Resource**

**Owner**

**OAuth 2.0**

**Client**

Request Access

Request Access

Token/ OAuth Code

Use Access

Token

**SAML 2.0 IdP**

**Assertion Generator**

**ADFS Proxy Handler**

SAML request

Bearer Assertion

*The SAML grant type for OAuth doesn’t provide a refresh token. The lifetime of the access token issued by the OAuth authorization must match the lifetime of the SAML token used in the authorization grant.*

*If the SAML token expires, but the user still has a valid browser session. In such a scenario, the web application has to redirect the user back to the SAML 2.0 identity provider, get a new SAML token, and exchange that token for a new access token. If the session at the SAML 2.0 identity provider is still live, then this redirection can be made transparent to the end user.*

**Scenario n°3 OpenID Connect**

/!\ Using a version of AD FS that supports OpenId Connect, namely either (AD FS 4.0 on Windows Server 2016).

**Java Certified OpenID Provider Libraries (**<https://openid.net/developers/certified/>)

#### [Connect2id Server 6.1.2a](https://connect2id.com/products/server)

* **Delivers OpenID Connect and OAuth 2.0 to the enterprise**
* *Target Environment:* Java in Apache Tomcat web server
* *License:* TBD
* *Certified By:* Connect2id
* *Conformance Profiles:* Basic OP, Implicit OP, Hybrid OP, Config OP, Dynamic OP

#### [Gluu Server 2.3](http://gluu.org/)

* **The Gluu Server is a free open source identity and access management platform for single sign-on, mobile authentication, and API access management that includes a comprehensive implementation of an OpenID Connect Provider and Relying Party**
* *Target Environment:* The Gluu Server OpenID Provider is written in Java. Packages are available for Centos, Red Hat, Ubuntu, and Debian.
* *License:* See <https://gluu.org/docs/#license>
* *Certified By:* Michael Schwartz
* *Conformance Profiles:* Basic OP, Implicit OP, Hybrid OP, Config OP, Dynamic OP

#### [Gluu Server 3.1.1](http://www.gluu.org/)

* **The Gluu Server is a free open source identity and access management platform for single sign-on, mobile authentication, and API access management that includes a comprehensive implementation of an OpenID Connect Provider and Relying Party**
* *Target Environment:* Java
* *License:* See <https://gluu.org/docs/ce/3.1.1/#license>
* *Certified By:* Michael Schwartz
* *Conformance Profiles:* Basic OP, Implicit OP, Hybrid OP, Config OP, Dynamic OP

#### [MITREid Connect](https://github.com/mitreid-connect/OpenID-Connect-Java-Spring-Server)

* **Customizable Java-based implementation of OAuth 2, OpenID Connect, and UMA designed for personal and enterprise scenarios**
* *Target Environment:* Java Spring backend, JavaScript front-end management UI
* *License:* Apache 2.0
* *Certified By:* Justin Richer
* *Conformance Profiles:* Basic OP, Config OP, Dynamic OP

#### [OIDC OP Overlay for Shibboleth IdP v3.2.1 version 1.0](https://github.com/uchicago/shibboleth-oidc)

* **This module adds OIDC support to the Shibboleth Identity Provider**
* *Target Environment:* Java
* *License:* Apache 2.0
* *Certified By:* University of Chicago
* *Conformance Profiles:* Basic OP, Config OP

#### [Cobalt V1.0](http://viewds.com/products/cobalt.html)

* **Cobalt is an identity and access management (IAM) platform for the cloud. It includes a federated identity service that supports both OIDC and SAML 2.0, as well as a cloud identity store with an integrated identity data management service based on OData and a fine-grained authorization service based on XACML.**
* *Target Environment:* Java on Vert.x
* *License:* Proprietary software licensed by subscription
* *Certified By:* ViewDS
* *Conformance Profiles:* Basic OP, Implicit OP, Config OP

## Scenarios

**AuthN and AuthZ SAML/ADFS:**



***Active Directory***

*Claims Provider (CP)*

***AD FS***

*Identity Provicer (IdP)*



***SAML Proxy***

*Service Provider (SP)*



***SPH***

***TSI***

***UCS***



Trust

*AuthN CFS ~ 42*

*AuthZ CFS ~ 8*

*AuthN CFS ~ 6*

*AuthN CFS ~ 19*

*AuthZ CFS ~ 38*

*SAML/HTTPS*



***Σ CFS (AuthN+AuthZ) ~ 113***

***All application security layers on role/actions, front-end or back-end will not be changed (review and re-certification).***

## Scenario n°1

Authentication LDAP/AD

#### 

#### UCS (Cash, Drv) / SPH: Authentication LDAP

AWFM Realms is a common security module based on Apache Catalina (JDBC Realm, Single Sign On authenticator)

### Prerequisites

* LDAP realm configuration (connectionURL, connectionName...)
* Review and re-certification entitlements matrix
* Application of a password policy

### Steps

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step id** | **Description** | **Complexity** | **Cost (md)** | |
| 0 | Development environment (add an embedded configuration and populate LDAP) | 1 | 1 | |
| 1 | Add a configuration JndiRealm Apache Catalina in AWFM Realms with LDAP Attributes | 5 | 8 | |
| 2 | Mapping roles AD (ldap attributes)/app features in relational database | 3 | 5 | |
| 3 | App (Backend native security) authorization review | 5 | 8 | |
| 4 | App Admin (HMI) review | 3 | 5 | |
| 5 | Logout control | 2 | 3 | |
| 6 | Log review | 2 | 3 | |
| 7 | Unit Testing | 2 | 3 | |
| 8 | No regression test | 2 | 3 | |
| 9 | Test to Qualif LDAP/AD | 2 | 3 | |
| TOTAL | | | | 42 md | |

#### TSI: Authentication LDAP

* The user authentication based on Apache Shiro security (nsi-fwk-\* module).
* The security is delegated to Tomcat. Once authentication has been carried and the session authorizes access to the TSI HMI,
* Shiro retrieves the user's groups by login.

### Prerequisites

* LDAP realm configuration (connectionURL, connectionName...)
* Review and re-certification entitlements matrix
* Application of a password policy

### Steps

|  |  |  |  |
| --- | --- | --- | --- |
| **Step id** | **Description** | **Complexity** | **Cost (md)** |
| 0 | Development environment (add an embedded configuration and populate LDAP) | 1 | 1 |
| 1 | Add a configuration JndiRealm Apache Catalina in AWFM Realms with LDAP Attributes | 5 | 7 |
| 2 | Mapping roles AD (ldap attributes)/app features in relational database | 3 | 5 |
| 3 | App (Backend/Apache Shiro security) authorization review and checks | 5 | 7 |
| 4 | App Admin (IHM) review | 3 | 5 |
| 5 | Logout control | 2 | 3 |
| 6 | Log review | 2 | 3 |
| 7 | Cache optimisation (optional) | 2 | 3 |
| 8 | Unit Testing | 2 | 3 |
| 9 | No regression test | 2 | 3 |
| 10 | Test to Qualif LDAP/AD | 2 | 3 |
| TOTAL | | | 43 md |

**TOTAL: 85 md**

## Scenario n°2 - OAuth2/SAML ADFS (2.x, 3.x, 4.x)

* For this section the recommended strategy is SP SSO initiated module
* RBAC will be used
* For this guideline and estimates we have assumed that the infrastructure (ADFS & AD) is already installed

#### 

### Authentication

### Prerequisites

* AD FS Configuration
* Application of a password policy
* Review and re-certification entitlements matrix
* Application of a password policy

### Common Steps

* SAML SP roles: generate SAML auth request, parse & verify SAML assertion, generate SAML logout request, parse & verify SAML logout response, audit logging …
* SAML IdP roles:parse & verify SAML auth request, generate SAML assertion (auth response), verify SAML logout request, generate SAML logout response,..

### Steps

|  |  |  |  |
| --- | --- | --- | --- |
| **Step id** | **Description** | **Complexity** | **Cost (md)** |
| 0 | Development environment (lab/Dev ADFS or another IdP solution: e.g. ssocircle) | 1 | 1 |
| 1 | Configure IdP (ADFS) | 6 | 8 |
| 1.1 | Configuring relying party (SAML 2.0 WebSSO protocol, entity id identify, reply url: addertsion consumer service url) | 1 | 1 |
| 1.1 | Register application (TSI, UCS and SPH)manually or with xml federation metadata file | 1 | 1 |
| 1.2 | Configure IdP Metadata | 1 | 1 |
| 1.3 | 1.3. Configuration of claims (rules, user information, attribute store Active Directory, mapping of ldap attributes, SAML attribute statement)  ```Transform an Incoming Claim: transform a Claim sent to ADFS (by Identity Providers) and send it back ```  ```Send LDAP Attributes as Claims: return LDAP attributes as Claims ```  ```Send Claims Using a Custom Rule: issue (Type = "access", Value = "true")``` | 1 | 1 |
| 1.4 | Configure a secure hash algorithm (SHA-512 or SHA-256 or SHA-128) | 1 | 1 |
| 1.5 | Configure a multi-factor authentication MFA | 1 | 1 |
| 2 | SP Proxy SAML (e.g. OpenSAML) | 42 | 25 |
| 2.1 | Development a common module proxy/filter SP SAML (e.g. OpenSAML common-sp-proxy) | 5 | 7 |
| 2.2 | Generate SP Metadata | 3 | 5 |
| 2.3 | Security configuration (signature, encryption and trust keys)  ```digital signatures, encryption, JKS keystore, Private/Public key, SSL/TLS certificates, X509Certificate``` | 5 | 7 |
| 2.4 | Single sign-on discovery (IdP selection, default/local/remote) (optional)  ```mechanism used to determine which identity provider should be used to authenticate user```  ```currently interacting with the service provider``` | 5 | 7 |
| 2.5 | Error handling | 3 | 5 |
| 2.6 | Add security event logging | 3 | 5 |
| 2.7 | Logout process (Local/Global) | 5 | 7 |
| 2.8 | Session, cookie and cache management | 2 | 3 |
| 2.9 | Add nginx as reverse proxy and load balancer (optional) | 5 | 7 |
| 2.10 | Customization of validity intervals and cache duration | 3 | 5 |
| 2.11 | Automatically update SP metadata mechanism and new certificat X.509 | 3 | 5 |
| 3 | Integration to applications (TSI, UCS and SPH)  <!--💡 two Options to integrate the common-sp-proxy module: the SAML module can be directly embedded Or ```  ```using the SAML library/dependencies is deploying it as a stand-alone module which transfers information about the ```  ```authenticated user to the target application using a custom mechanism --> | 19 | 15 |
| 3.1 | . Add a client authentication adapter (web agents): add dependencies, configuration filter and POST/REDIRECT Bindings  (Java-based or using XML), web.xml (filter mapping, init param) | 13 | 10 |
| 3.2 | Configure/Customization metadata (IdP/SP metadata) by operating environment | 5 | 7 |
| 4 | Testing SSO (single sign-on), SLO (single logout) | 1 | 1 |
| TOTAL | | | 136 md |

### Authorization

### Prerequisites

* Review and re-certification entitlements matrix ```

### Common: Authorization RBAC

### Steps

|  |  |  |  |
| --- | --- | --- | --- |
| **Step id** | **Description** | **Complexity** | **Cost (md)** |
| 1 | Mapping roles AD/app features  ```use a library RBAC to manage identity profiles (e.g Apache Fortress or Apache Syncope or Apache Shiro)``` | 8 | 9 |

### TSI: Authorization RBAC

### Steps

|  |  |  |  |
| --- | --- | --- | --- |
| **Step id** | **Description** | **Complexity** | **Cost (md)** |
| 1.1 | Setup RBAC Policy and review (service/controller layers, REST API,IHM Front, privileged actions MFA) | 8 | 9 |
| 1.2 | No regression test | 3 | 5 |
| TOTAL | | | 14 md |

### UCS: Authorization RBAC

*💡 If the Apache Shiro/Fortress solutions are not feasible. The solution is to create an application (the bridge) that will be a bridge between oAuth and SAML. When a non-authorized client tries to access the protected resource, it is redirected to the authorization server*

### Steps

|  |  |  |  |
| --- | --- | --- | --- |
| **Step id** | **Description** | **Complexity** | **Cost (md)** |
| 1.1 | Setup RBAC Policy and review (service/controller layers, REST API,IHM Front, privileged actions MFA) | 13 | 10 |
| 1.2 | Non regression test | 3 | 5 |
| TOTAL | | | 15 md |

### SPH: Authorization RBAC

### Steps

|  |  |  |  |
| --- | --- | --- | --- |
| **Step id** | **Description** | **Complexity** | **Cost (md)** |
| 1.1 | Setup RBAC Policy and review (service/controller layers, REST API,IHM Front, privileged actions MFA) | 8 | 9 |
| 1.2 | Non regression test | 3 | 5 |
| TOTAL | | | 14 md |

**TOTAL : 188 md**

## Scenario n°3

### OpenID Connect ADFS 4.0

(In progress)

## Status

29/05 : poc scénario OAuth2                                     ok ****

29/06 : poc scénario SAML 2.0                                  ok ****

29/06 : poc scénario OpenID Connect                    ko **X**

***Limites/Contraintes:***

* Compatibilité ADFS 4.0 / AD windows server 2008 R2 => ?
* Difficulté pour implémenter la partie autorisation SAML => *la combinaison SAML/Librairies RBAC (fortress, Syncope, Shiro) permet un mapping des autorisations (rôle/actions) à granularité fine.*
* Je n’arrive pas à valider une conception OpenID Connect mutualisée pour la prise en charge du parc applicatif (UCH/SPH/TSI, Autres architectures).
* Bien que OpenID Connect a été construit sur la base de l'expérience des anciens protocoles, il n'a pas été exposé au même niveau de contrôle et aux attaques de sécurité que les autres protocoles. Des travaux d’implémentation sont toujours en cours et seules les spécifications de base sont terminés.

## APPENDICES

* OpenSAML (<http://www.opensaml.org/>)

It is an open source library (Java or C++), produce & consume SAML messages, create & validate digital signatures/encryption, generate & parse SAML metadata, encoding and transmitting SAML messages.

* Apache Syncope ( http://syncope.apache.org/ )

Open Source system for managing digital identities in enterprise environments, implemented in JEE technology and released under Apache 2.0 license.

* Apache Shiro ( https://shiro.apache.org/ )

Apache Shiro is an open source powerful and easy-to-use Java security framework that performs authentication, authorization, cryptography, and session management. With Shiro’s easy-to-understand API, you can quickly and easily secure any application – from the smallest mobile applications to the largest web and enterprise applications.

* Apache Fortress ( https://directory.apache.org/fortress/ , https://directory.apache.org/fortress/overview.html )

Apache Fortress is an open source project and a standards-based authorization system, that provides role-based access control, delegated administration and password policy services using an LDAP backend. Supports ANSI INCITS 359 RBAC and more.

Standards implemented:

* Role-Based Access Control (ANSI INCITS 359)
* Administrative Role-Based Access Control (ARBAC02)
* IETF Password Policy (draft)
* Unix Users and Groups (RFC2307)

### Gartner Magic Quadrant

## Scenario n°1 OAuth 2 SAML 2.0 Bearer Assertion Flow

Pac4j 🗸, Nimbus, Spring Security, Scribe, MitreID 🗸, Apache Oltu 🗸 ([*https://oauth.net/code/java/*](https://oauth.net/code/java/))

* Client application redirects to the SAML server for authentication (SAML request).
* The SAML server checks the user credentials, creates and signs a SAML Response.
* Request (redirection) to the Authorisation server with the SAML Response in the payload and a state parameter that will be used to access the user’s session.
* The Authorisation server checks SAML Response stores a hash of the SAML response so that it can be used later for verifying the user identity, creates an application cookie identity with the information provided by SAML and redirects the authorisation endpoint. The Authorisation server creates a new bearer type identity for the user with all the rights that the user authorised and redirects to the Client application (redirect URL) with parameters authentication code and the application state.
* The Client application uses the state parameter to access the user’s session state and issues a POST back to the Authorisation server in order to exchange the authorisation code for an OAuth access token. The request includes the username and SAML hash to be compared as means of authentication.
* The Authorisation server compares the provided SAML hash with the one stored in the cache to authenticate the user and returns an OAuth access token to the Client application.
* The Client application stores the OAuth access token in a cookie. For any request the Client application uses a request Handler and adds the OAuth Token as a URL parameter.

***Autorization***

***Server***

***OAuth 2.0***

***Resource***

***Server***

***Resource***

***Owner***

***OAuth 2.0***

***Client***

*Request Access*

*Request Access*

*Token/ OAuth Code*

*Use Access Token*

***SAML 2.0 IdP***

***Assertion Generator***

***ADFS Proxy Handler***

*SAML request*

*Bearer Assertion*



**

Trust

*OpenSAML*





## Scenario n°2 OpenID Connect

[*https://openid.net/developers/certified/*](https://openid.net/developers/certified/)

* **Java Certified OpenID Provider Libraries (***Connect2id, Gluu, MitreID* ***🗸,*** *Shibboleth Idp***)**
* **Certified OpenID Provider Servers and Services (***ADFS Windows Server 2016* ***🗸****, Keycloak (favors the Openid Connect)****🗸****, Okta OP***)**
* **Certified Financial-grade API (FAPI) OpenID Providers (***WSO2 Open Banking***)**



[*https://openid.net/developers/uncertified/*](https://openid.net/developers/uncertified/)

* Nimbus
* Apache Oltu 🗸

[*https://openid.net/developers/jwt/*](https://openid.net/developers/jwt/) *,* [*https://openid.net/developers/libraries/obsolete/*](https://openid.net/developers/libraries/obsolete/)

*OIDC Provider (OP): may rely on itself, or another Identity Provider (IdP) (e.g. WS-Federation or SAML).*

**4.0**



*Request Access Token*

***OpenId Connect***

***Server Provider***

***(IdP)***

***Resource***

***Server***

***End User***

**Relying Party**

***OIDC Client***

***(RP)***

*Request Access*

*Use ID/JWT Token*

***SAML 2.0 IdP***

***Assertion Generator***

***ADFS Proxy Handler***

*SAML request*

*Bearer Assertion*



*Access/ID/Refresh Token*

Trust

*OpenSAML*





MSAL

[](https://mvnrepository.com/artifact/com.microsoft.identity.client/msal)

Configure OAuth2, OpenID Connect, or Active Directory Graph authentication and SAML 2.0 single sign on with the Microsoft Authentication Library MSAL (<https://docs.microsoft.com/fr-fr/azure/active-directory/develop/msal-java-adfs-support>).

## Scenario n°3 SAML 2.0

/!\ The use of Open Source solutions is important, as the security community can verify the implementation to assess its security.

* The user is trying to connect to the application. The SAML agent intercepts the request. The user is not authenticated, so he is redirected to SAML SP.
* SAML SP receives the authentication request (SAML request) and delegate the authentication to ADFS (IdP).
* If you are on a local network and your browser is configured to send Windows session authentication to ADFS (IdP). Then the user will be authenticated without having to enter their identifiers. If you are on an external network or an internal network but your browser is not configured to send the identifiers of the Windows session, so the ADFS (IdP) authentication target appears. The user enters their information. In both cases, in the event of successful authentication, the ADFS (IdP) issues a response (SAML response).
* SAML SP receives the SAML response. With the information contained in the latter, the user now has an SAML authentication token. It is redirected to the application.
* The user accesses the application.

Use a library as SP and IdP ([*https://en.wikipedia.org/wiki/SAML-based\_products\_and\_services*](https://en.wikipedia.org/wiki/SAML-based_products_and_services)*)*

* ***(1)*** *Libraries are used by developers to integrate applications into SAML federations or to build their own SAML-actors like IdPs.*
* ***(2)*** *Products that provide SAML actors.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ***License*** | ***Roles*** | ***Protocols*** |  |
| *Spring Security SAML* | *Open Source* | *SP,* **(1)** | *All products supporting SAML 2.0 in Identity Provider mode (e.g. ADFS, Okta, Shibboleth, OpenAM)* | *Based on OpenSAML* |
| *Auth10-java* | *Open Source* | *SP,* **(1)** | *WS-Federation, SAML 1.1/2.0, ADFS, Windows Azure Active Directory* | *Based on OpenSAML* |
| *OpenSAML* | *Open Source* | *IdP, SP* **(1)** | *SAML-Library: C++, Java* | *Create SAML messages*  *Parsing SAML objects and exporting them to XML format*  *Signature and encryption*  *Encoding and transmitting SAML messages.* |
| *Pac4j-SAML* | *Open Source* | *SP,* **(1)** | *OAuth, CAS, SAML, OpenID Connect, LDAP, JWT* | *Based on OpenSAML* |
| *OIOSAML* | *Open Source* | *SP,* **(1)** | *SAML-Library* | *Based on OpenSAML* |
| *ADFS* | *Commercial* | *IdP,* **(2)** | *WS-Federation, WS-Trust, SAML 2.0, OAuth2, OpenID Connect* |  |
| *Okta* | *Commercial* | *IdP, SP,* **(2)** | *WS-Federation, WS-Trust, SAML 2.0, OAuth2, OpenID Connect* |  |
| *Shibboleth* | *Open Source* | *IdP, SP, Discovery,* **(2)** | *SAML 1.1, SAML 2.0* |  |
| *Keycloak* | *Open Source* | *IdP, SP,* **(2)** | *OAuth 2.0, OpenID Connect, JSON Web Token (JWT),* [*SAML 2.0*](https://en.wikipedia.org/wiki/SAML_2.0) |  |
| *OpenAM* | *CDDL*  *may rise issues (if the software needs to be extended).* | *ECP (Enhanced Client or Proxy), IdP,* **(2)** | *OpenID Connect, OAuth2, SAML 1.1/2.0, WS-Federation, WS-Trust, Kerberos* | “All-in-one” open source access management solution.  Provides single and federated Single Sign-On (SSO) solutions out of the box (username/password, SAML and OpenID Connect).  Easily integrates into existing systems.  Allows you to build a federated identity and entitlement solution easily.  Entitlements engine that supports the latest XACML protocol.  Full support for OAuth 2.0 and OpenID Connect  Variety of Web/J2EE Agent software that enforces Authorization policy decisions for OpenAM on protected resources (sometimes maybe not suitable for existing infrastructure configurations).  Gateway/reverse proxy complimentary software (OpenIG) that allows to protect resources without modifying existing server configurations.  Easy configurable using GUI and CMD tools.  Combining with CDDL 1.0 license may rise issues (if the software needs to be extended). |
| *WSO2* | *Open Source* | *IdP,SP,* **(2)** | *OAuth2, WS-Trust, OpenID* |  |
| *JOSSO* | *Open Source* | *IdP,SP,* **(2)** | *SAML2, OAuth2, WS-Trust, SPMLV2, Kerberos* |  |
| *SATOSA* | *Open Source* | *Broker,* **(2)** | *Proxy (SAML2, OpenID Connect, OAuth2)* |  |
| *Central Authentication Server (CAS)* | *Open Source* | *IdP,* **(2)** | *SAML 2.0, OAuth2, OpenID, WS-Federation* | Mature developer API (as a result faster development and less prone to errors)  Centralised solution for federated SSO Widely adopted, mainly because it provides centralised authentication mechanism  New versions of CAS provide support for OpenID Connect and OAuth 2.0 (SAML support was native)  CAS server does not provide any concept of Authorization therefore other technologies in line with CAS need to be implemented (additional and potentially very expensive overhead)! |
| *adAS* | *Open Source* | *IdP,SP,* **(2)** | *WS-Federation, WS-Trust, SAML 1.0/2.0, Microsoft365, Kerberos, LDAP, Federation* |  |
| *LemonLDAP::NG* | *Open Source* | *IdP,SP,* **(2)** | *SSO, WS-Federation, CAS, OpenID Connect, SAML 2.0, Protocol Proxy* |  |
| *Gluu* | *Open Source* | *IdP,* **(2)** | *SAML, OIDC, OAuth2, CAS* | A SAML Proxy enables an organization to consolidate inbound SAML authentication from the IDPs of partners to a website or an application.  Allows you to build a federated identity and entitlement solution easily.  Entitlements engine that supports the latest XACML protocol.  Full support for Oauth 2.0 and OpenID Connect.  Lack of some administration, development and tutorial documentation that makes training/integration more complicated, time consuming and more likely more error prone - Currently no solutions for gateways/reverse proxies that will allow to protect resources without modifying existing server configurations (editing existing server configuration is not the best option!) |
| *Aerobase* | *Open Source* | *IdP,* **(2)** | *Federated SSO (LDAP/Active Directory), standard protocols (OpenID Connect, OAuth 2.0, SAML 2.0)* |  |

There are several technologies that provide security solutions out-of-the-box. It is important from various perspectives (resources, time etc.) to reuse well defined.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **CAS** | **OpenAM** | **Gluu** |
| Open Standards | *Yes* | *Yes* | *Yes* |
| Single Sign-On (SSO) | *Yes* | *Yes* | *Yes* |
| Support for modern Authentication mechanisms (e.g. SAML, OpenID Connect etc.) | *Yes* | *Yes* | *Yes* |
| Support for various Authorization mechanisms (e.g. custom user policies, XACML etc.) | Not applicable | *Yes* | *Yes* |
|  |  |  |  |
| Variety of Web Policy Agents to serve as a container for protected resources | Not applicable | Yes (various) | Yes (some) |
| Ready and easy configurable Gateways/Reverse Proxy solutions to serve as a container for protected resources | Not applicable | Yes (well-defined and mature OpenIG Federation Gateway software) | Yes (some) |
| Admin and development documentation | Yes (not applicable for AuthZ) | Yes (detailed) | Yes (limited in some cases) |
| License | Open source (Apache 2.0) | Open source (CDDL1.0. Common Development and Distribution License) | Open source (MIT) |

|  | **Keycloak** | **WSO2 Identity Server** | **Gluu** | **CAS** | **OpenAM** | **Shibboleth IdP** | **LemonLDAP::NG** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *OpenID Connect/OAuth support* | Yes | Yes | Yes | Yes | Yes | Third-party | Yes |
| *Multi-factor authentication* | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| *Admin UI* | Yes | Yes | Yes | Yes | Yes | No | Yes |
| *OpenJDK support* | Yes |  | No | Yes | Yes | Partial | N/A (Perl) |
| *Identity brokering* | Yes | Yes | Yes |  |  |  | Yes |
| *Middleware* | Wildfly, JBOSS | WSO2 Carbon | Jetty, Apache HTTPD | any Java app server | any Java app server | Jetty, Tomcat | Apache HTTP, Nginx, etc |
| *Open source* | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| *Commercial support* | Yes | Yes | Yes | Third-party | Yes | Third-party | Third-party |
| *Add federation metadata* | **No** | Yes |  |  |  | Yes | Yes |
| *Add metadata from URL* | **No** | Yes |  |  |  | Yes | Yes |
| *Installation and configuration* | easy |  |  | difficult |  | difficult | moderate |

## Disadvantages/ Advantages

|  |  |  |
| --- | --- | --- |
|  | ***Disadvantages*** | ***Advantages*** |
| ***Scenario n°1*** | *The SAML grant type for OAuth doesn’t provide a refresh token. The lifetime of the access token issued by the OAuth authorization must match the lifetime of the SAML token used in the authorization grant.*  *If the SAML token expires, but the user still has a valid browser session. In such a scenario, the web application must redirect the user back to the SAML 2.0 identity provider, get a new SAML token, and exchange that token for a new access token. If the session at the SAML 2.0 identity provider is still live, then this redirection can be made transparent to the end user.* | *Less expensive.*  *All OAuth2 advantages.*  *Integration with SAML to resolve the gap related to authentication.*  *OAuth is more tailored towards access scoping than SAML. Access scoping is the practice of allowing only the bare minimum of access within the resource/app an identity requires once verified (what you can do).* |
| ***Scenario n°2*** | *Not supported by old version ADFS.*  *Using a version of AD FS that supports OpenId Connect, namely either (AD FS 4.0 on Windows Server 2016)* | *All OpenId Connect/OAuth2 advantages.*  *If ADFS 4.0 will be used, the compatibility problem will be solved. Otherwise integration with SAML to support older ADFS versions.* |
| ***Scenario n°3*** | *Does not support embedded (native) and mobile applications.*  *SAML does not do authorization explicitly. It simply provides the attributes in the SAML token and it's up to the application as to how these are handled.* | *All SAML advantages.*  *Only one SP and IdP library* |