Large-scale EXecution for Industry & Society

Deliverable D4.1

Analysis of mechanisms for securing federated infrastructure

Co-funded by the Horizon 2020 Framework Programme of the European Union
Grant Agreement Number 825532
ICT-11-2018-2019 (IA - Innovation Action)

DELIVERABLE ID | TITLE	D4.1 | Analysis of mechanisms for securing federated infrastructure
RESPONSIBLE AUTHOR	Frédéric Donnat (O24)
WORKPACKAGE ID | TITLE	WP4 | Orchestration and Secure Cloud/HPC Services Provisioning
WORKPACKAGE LEADER	LINKS
DATE OF DELIVERY (CONTRACTUAL)	30/09/2019 (M09)
DATE OF DELIVERY (SUBMITTED)	17/10/2019 (M10)
VERSION | STATUS	V1.3.2 | Final
TYPE OF DELIVERABLE	R (Report)
DISSEMINATION LEVEL	PU(Public)
AUTHORS (PARTNER)	O24; LINKS; IT4I; LRZ
INTERNAL REVIEW	Martin Golasowski (IT4I); Roberto Peveri (TESEO)

Project Coordinator: Dr. Jan Martinovič – IT4Innovations, VSB – Technical University of Ostrava
E-mail: jan.martinovic@vsb.cz, **Phone:** +420 597 329 598, **Web:** https://lexis-project.eu

DOCUMENT VERSION

VERSION	MODIFICATION(S)	DATE	AUTHOR(S)
0.1	Defined the Table of Content; first draft of the document	31/07/2019	Donnat F. (O24), Scionti A. (LINKS)
0.2	Improving text also in connection with D2.2	18/09/2019	Colucci A. (O24), Donnat F. (O24), Scionti A. (LINKS)
0.3	Final draft completed, ready for internal review	19/09/2019	Colucci A. (O24), Donnat F. (O24), Scionti A. (LINKS)
0.4	Last check and cleaning of the text before internal reviews	20/09/2019	Scionti A. (LINKS)
0.5	Internal review	26/09/2019	Golasowski M. (IT4I), Roberto Peveri (TESEO)
0.6	Answering comments	28/09/2019	Scionti A. (LINKS), Donnat F. (O24)
1.0	Final version	29/09/2019	Scionti A. (LINKS), Donnat F. (O24)
1.1	integrate review from Work Package leaders	30/09/2019	Martinovic J. (IT4I), Vojacek L. (IT4I), Hachinger S. (LRZ), Levrier M. (Bull/Atos)
1.2	Final version adding use-case in Introduction, creating appendix for technical definition and updating requirements section	30/09/2019	Scionti A. (LINKS), Colucci A. (O24), Donnat F. (O24)
1.2.2	Rewrite Exec Summary, Chapter 1 and Chapter 4	04/10/2019	Martinovic J. (IT4I), Golasowski M. (IT4I)
1.2.3	Update APPENDIX, Glossary, Table of Content, fix minor typo and merge second review from Work Package Leaders	06/10/2019	Scionti A. (LINKS), Levrier M. (Bull/Atos), Murphy S. (Cyclops), Donnat F. (O24)
1.3	Corrections, re-ordering some sections to facilitate reading.	09/10/2019	Hachinger S. (LRZ)
1.3.1	Formatting corrections, Table of Contents links corrections, re-merge changes removed due to previous correction	10/10/2019	Donnat F. (O24), Hachinger S. (LRZ), Colucci A. (O24)
1.3.2	Final formal check by coordinator	16/10/2019	Slaninova K. (IT4I)

GLOSSARY

ACRONYM	DESCRIPTION
AAI	Authentication and Authorization Infrastructure
ABAC	Attribute-based Access Control
ACL	Access Control List
API	Application Programming Interface
CADF	Cloud Auditing Data Federation
CBAC	Context-based Access Control
CPU	Central Processing Unit
DDI	Distributed Data Infrastructure
DMTF	Distributed Task Management Force
DNS	Domain Name System
FPGA	Field-Programmable Gate Array
GPU	Graphics Processing Unit
IETF	Internet Engineering Task Force
HMAC	Hash-Based Message Authentication Code (also expanded as Keyed-Hash Message Authentication Code)
HPC	Hyper-Performance Computing
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IAM	Identity and Access Management
IDP	Identity Provider
JSON	JavaScript Object Notation
JWS	JSON Web Signature
LDAP	Lightweight Directory Access Protocol
MFA	Multi-factor Authentication
NAS	Network Attached Storage
NIST	National Institute of Standard and Technology
OASIS	Organization for the Advancement of Structured Information Standards
OIDC	OpenID Connect
OTP/TOTP	(Time-based) One-Time Password
PAM	Pluggable Authentication Module

PIN	Personal Identification Number
POC	Proof Of Concept
RBAC	Role-based Access Control
RDBMS	Relational Database Management System
REST	Representational State Transfer
RSA	Rivest Shamir Adleman
SAML	Security Assertion Markup Language
SAN	Storage Area Network
SP	Service Provider
SQL	Structured Query Language
SSH	Secure SHell
SSO	Single Sign-On
TFA/2FA	Two-factor Authentication
UBAC	User-based Access Control
UDP	User Datagram Protocol
UI	User Interface
USB	Universal Serial Bus
WCDA	Weather and Climate Data API
WP	Work Package
XACML	eXtensible Access Control Markup Language
YORC	Ystia ORChestrator

TABLE OF PARTNERS

ACRONYM	PARTNER
Avio Aero	GE AVIO SRL
AWI	ALFRED WEGENER INSTITUT HELMHOLTZ ZENTRUM FUR POLAR UND MEERESFORSCHUNG
BLABS	BAYNCORE LABS LIMITED
Bull/Atos	BULL SAS
CEA	COMMISSARIAT A L ENERGIE ATOMIQUE ET AUX ENERGIES ALTERNATIVES
CIMA	Centro Internazionale in Monitoraggio Ambientale - Fondazione CIMA
CYC	CYCLOPS LABS GMBH
ECMWF	EUROPEAN CENTRE FOR MEDIUM-RANGE WEATHER FORECASTS
GFZ	HELMHOLTZ ZENTRUM POTSDAM DEUTSCHESGEOFORSCHUNGSZENTRUM GFZ
IT4I	VYSOKA SKOLA BANSKA - TECHNICKA UNIVERZITA OSTRAVA / IT4Innovations National Supercomputing Centre
ITHACA	ASSOCIAZIONE ITHACA
LINKS	FONDAZIONE LINKS / ISTITUTO SUPERIORE MARIO BOELLA ISMB
LRZ	BAYERISCHE AKADEMIE DER WISSENSCHAFTEN / Leibniz Rechenzentrum der BAdW
NUM	NUMTECH
O24	OUTPOST 24 FRANCE
TESEO	TESEO SPA TECNOLOGIE E SISTEMI ELETTRONICI ED OTTICI

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

The WP4 (Orchestration and Secure Cloud/HPC Services provisioning) is in charge of delivering the building blocks for the overall LEXIS platform architecture. As a matter of fact, in order to be able to orchestrate services or tasks, it is required to have knowledge of:

- Available resources of the platform (Cloud, HPC, Storage);
- Users of the platform and their roles;
- Workflows and tasks executed by the users in the platform.

Due to this, WP4 must provide the Authentication and Authorization Infrastructure (AAI) for the whole LEXIS Project infrastructure. The "Analysis of mechanisms for securing federated infrastructure" is the very first deliverable for WP4. This analysis is also driven by all preceding Co-Design activities and WP2 analysis on "Pilots needs / Infrastructure Evaluation Report" [1].

The LEXIS platform will have an own AAI solution, storing and maintaining LEXIS user accounts. LEXIS users will interact with the platform through these accounts. The LEXIS AAI will ensure proper authentication and secure lifecycle. The AAI will also provide a fine-grained user role and rights management in order to grant appropriate access rights for the main elements of the platform (portal, computational resources and storage).

Since the AAI will be operated by LEXIS itself, it has to conform to relatively strict requirements based in part on best practices in user management and service operation and in part on the characteristics of the pilots (WP5/6/7).

Position of the deliverable in the whole project context

The LEXIS AAI is one of the pillars for LEXIS Project. Any other work package will need to access the AAI system:

- WP3: The "LEXIS Data System (DDI)" will provide the LEXIS end-users access to private and public datasets according to rights based on their role provided by the LEXIS AAI. This includes access to public datasets as well as restricted access for users to their own private datasets. Thus, the DDI must be aware of the users and their rights.
- WP4: The "LEXIS Orchestration System" will provide the computational part of LEXIS infrastructure and as such it must link/relate all jobs/tasks to users with access rights.
- WP8: The "LEXIS Portal System" is the web user interface and then it must rely on an authentication and authorization system to properly allow users to execute tasks or give them access to specific datasets.
- WP5, WP6 and WP7: These "LEXIS Pilots" will use the LEXIS infrastructure.

The pilots will execute their workflows on particular datasets through interaction with the three core elements of the platform. The LEXIS AAI will ensure proper authentication of the users interacting with the platform according to their requirements.

Description of the deliverable

This document provides a basic overview of the current state-of-the art in the field of user management, authentication protocols, authorization and best practices for operating such service. Basic terminology is described as well as common standards and protocols. Main purpose of this document is to evaluate existing AAI solutions based on the requirements and select the one which will be the most suitable for administrating the LEXIS user accounts. Our benchmarking methodology is described and evaluation results of several widely used solutions are provided. A detailed description of the AAI layer and its planned integration in the LEXIS platform is provided. Requirements gathered from the co-design phase (WP2) and all other work packages, including technical and pilot WPs, are provided here together with their integration to the proposed AAI solution. The document ends with a description of the Proof-of-Concept (PoC) of the AAI as seen at M09 of the project. Finally, conclusions summarize the whole contribution of this report.

1 AAI IN THE CONTEXT OF THE LEXIS PROJECT

From co-design activities started at the beginning of the LEXIS project, we have characterized and described the "LEXIS AAI" not only from a design perspective, but also from a functional approach taking into consideration the security aspects of the overall LEXIS platform.

As described in deliverable D2.1 (WP2), Section 4.1 — Federated Identity, Access and Data management [1], the goal for the LEXIS AAI is to provide a secure authentication and authorization system distributed among different data centres which will hold the individual LEXIS user accounts. Moreover, the LEXIS AAI will need to ensure that all the building blocks of the LEXIS infrastructure will be able to check for authentication and authorization for any user or process accessing LEXIS infrastructural resources (i.e., network, compute or storage resources). From the original design and diagram presented in D2.1, we derived the LEXIS AAI design. Figure 1 depicts the integration of the AAI system within the overall LEXIS platform. Interaction among different components of the LEXIS platform is also shown.

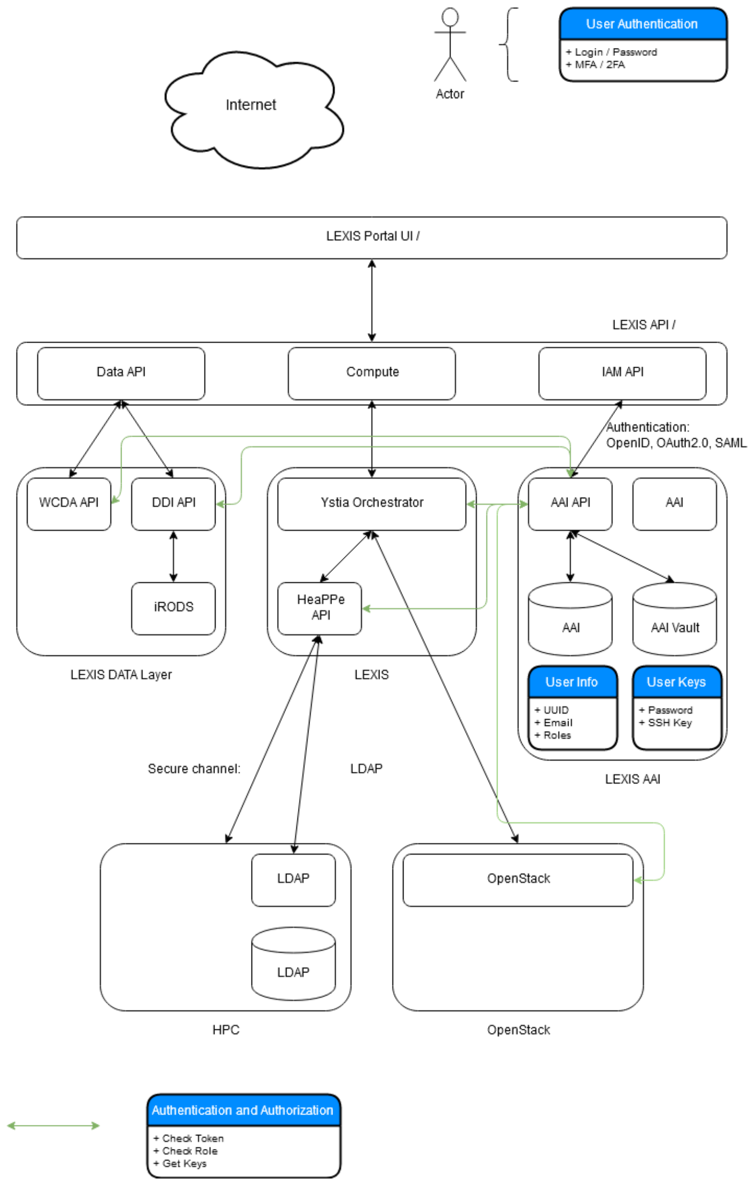


Figure 1 Authentication, authorization and identity system architecture

As shown in the diagram, the AAI service will be accessed from multiple integral parts of the platform. Therefore, it has to communicate through a proven protocol which will be easy to implement by the both data and orchestrator layers. The solution also needs to have high availability functionality to ensure resiliency, which can be achieved by operating an instance of such solution distributed in multiple data centres behind a load balancer.

Since LEXIS portal will rely on web-based technologies, protocols such as OpenID[1] or SAML[2] can be suitable to manage access tokens for the platform users.

2 DEFINITIONS AND OBJECTIVES

This section contains definitions of (security) concepts related to authentication, authorization and identity management from a security perspective. It also provides a definition of 'federation' in the context of information technology and how this applies to LEXIS infrastructure. The section ends with a short description of the design of AAI layer.

2.1 AUTHENTICATION, AUTHORIZATION AND IDENTITY (AAI)

This section describes basic concepts for authentication, authorization and identity management, with the aim of allowing readers to understand our AAI design process for LEXIS on the basis of a common background.

2.1.1 Authentication

Authentication is verifying the identity of a user, process or device which is often a prerequisite to allowing access to resources in an information system. This (cf. e.g. also NIST FIPS-200 [2]) is often a pre-requisite to give access to resource in an information system.

Usually, we differentiate 3 authentication types:

- **something you know**: this includes passwords, PINs, secret passphrases, etc. In few words, it refers to anything you can remember and (then) say, do or perform when requested;
- **something you have**: this includes physical objects, such as keys, smartphones, USB sticks, smart cards or token devices. In few words, it refers to anything you own that can be provided when requested.
- **something you are**: this includes any part of the human body, such as the fingerprint, face, iris, etc. In few words, it refers to any part of the human body that can support the verification process when requested.

Nowadays, most of the information systems are using one of these authentication types and the number of systems using 2-factor authentication (2FA) or multi-factor authentication (MFA) methods are growing. 2-factor authentication is the combination of 2 of these authentication types (similarly, multi-factor authentication systems combine more than 2 of the above-mentioned mechanisms). For instance, in such a system, most of the time you need to provide a password and then a secret number generated on your physical device or smartphone.

According to Cryptographic experts (cf. Handbook of Applied Cryptography [3]), we must pay attention in using fixed passwords and personal identification numbers (PINs), since their working scheme makes them fall under the category of 'symmetric key techniques providing unilateral authentication'. One-time password-based techniques are a step forward towards having a strong authentication mechanism in place.

On the other hand, strong authentication is provided using the 'Challenge-Response' scheme, where the idea is that the user, process or device (claimant entity) proves its identity to the system (verifier entity) by demonstrating knowledge of a secret without revealing or providing it. This is usually done by providing a response to a time-variant challenge (usually random and secret) where the response depends on both the secret and the challenge.

Strong authentication is often confused with 2FA or MFA; however, unless multiple authentication factors are used, such as *something you have* and *something you are*, it cannot be considered an MFA.

[1] OpenID: https://openid.net/connect/
[2] SAML: http://saml.xml.org/saml-specifications

2.1.2 Authorization

Authorization services are in charge of protecting application resources, managing user permissions and enforcing access policies.

Most of SSO systems support fine-grained authorization policies, usually offering different Access Control Mechanisms (ACM) such as:

- **User-based Access Control (UBAC)**: permissions are granted at the individual level. Despite allowing more granular control of the system, this scheme leads easily to management overload, as access rights have to be defined for each user separately;
- **Role-based Access Control (RBAC)**: users are assigned to a role (or a combination of roles), based on their job functions and privileges, which determine permissions they are granted to. This scheme allows a more structured way of granting access, organizing roles in a hierarchy where higher-level roles subsume permissions owned by sub-roles;
- **Attribute-based Access Control (ABAC)**: access rights are granted to users through the combination of different types of attributes (such as user attributes, resource attributes and environment conditions), determining permissions based on ‘who’ the users are instead of considering ‘what’ they do (like in RBAC);
- **Context-based Access Control (CBAC)**: with the support of firewall software features that intelligently filter TCP and UDP packets based on the application layer session information, access rights are granted to the user according to a specific contextual attributes (such as organization, application, network resources, etc.) that the user is attempting to gain access to.

2.1.3 Identity Federation

Identity federation is a way of linking a person's multiple digital identities, collecting all their attributes under the same entity, which can be shared among different domains to access several applications and services.

Identity federation is essential to Single Sign-On (SSO, cf. also NIST FIPS-800-63 [4]), where users' access is allowed after verifying their identity and issuing a single authentication `ticket' - called *token* - vouching for their identity and allowing access across multiple systems and organizations. “SSO is a subset of federated identity management, as it relates only to authentication, it is understood on the level of technical interoperability and it would not be possible without some sort of federation” [3].

2.2 ASPECTS OF IDENTITY FEDERATION

Aspects of identity federation, as mentioned as a fundamental component of AAI systems above, shall be discussed further in this section.

2.2.1 Single Sign-On (SSO)

Single Sign-On (SSO) is a security mechanism for enhancing user’s experience and security, as users’ authentication is requested only once for different applications federated within the same Identity Provider and keeping user’s credentials safe in the SSO server, preventing them from being cached by the actual service requiring them.

In general, this is achieved when the *SSO Identity Provider* authenticates the user and then issues a security token which is sent to the target applications (i.e., *SSO Service Provider*) asserting the successful user’s authentication and the reliability of their identity.

Often, an intermediary service is set up to connect multiple service providers with different identity providers. Such a kind of service is called *Identity Broker*.

[3] Federated identity: https://en.wikipedia.org/wiki/Federated_identity

2.2.2 Components of a typical federated identity / SSO system

A typical federated identity management system with SSO capabilities, thus consists of the following entities (including the party of the *Service Provider*):

- **Identity Provider (IDP)**:
  An Identity Provider is a system in charge of providing identity management and authentication within a federated or distributed infrastructure. Upon successful user authentication, the Identity Provider returns an authentication token that can be used as proof of the user identity.
  The Identity Provider usually authenticates user by validating a username/password (with or without any MFA) but it can also rely on another method or even another "trusted" Identity Provider for authenticating the user.
  As a matter of fact, the Identity Provider is in charge of managing the users' identities during their life-cycle (from creation to deletion). This system usually offers an API to facilitate integration with other applications such as web applications.
- **Service Provider (SP)**:
  In this context a Service Provider is a system providing "services" to an end user such as storage or processing. In modern software architecture (micro-services and not monolithic application), this system relies on another system called Identity Provider to provide identity management (authentication, authorization and management of the identity).
  In most cases the Service Provider completely relies on the Identity Provider to provide any user's attributes (not only authorization), but it may happen that the Service Provider also manages some very specific user's attributes that are only used locally.
  We need to notice that such an architecture usually provides enhanced usability from user perspective (usually coupled with SSO, user only needs to authenticate once) and better security (from identity management perspective, it avoids maintaining several identities in several places for the same user, thus reducing the attack surface).
- **Identity Broker**:
  The Identity Broker is an intermediary service in charge of creating a trusted connection among different Service providers with external Identity providers.
  When the user tries to access a resource, the Service Provider redirects him to the Identity Broker, which is in charge of providing a list of available Identity Providers to the user.
  Upon a successful authentication, the chosen Identity Provider issues a security token that will be used by the Service Provider to trust the authentication vouched by the Identity Provider and retrieves information about the user.

2.2.3 Standard Protocols

All the SSO systems mentioned above are based on standard protocols and provide support for SAML (2.0)[4] [6], OpenID Connect[5] [6] and/or OAuth (2.0)[6] [7, 8, 9].

Usually, standard protocols provide an access token, which is a token that can be provided as part of an HTTP request that grants access to the service being invoked on. This is part of the OpenID Connect and OAuth 2.0 specification.

Authentication and authorization standards important in our context are OpenID Connect, OAuth protocol (from IETF), LDAP [10] and the Kerberos [11] protocol.

[4] SAML specification: http://saml.xml.org/saml-specifications/
[5] OpenID Connect specification: https://openid.net/connect/
[6] OAuth 2.0: https://oauth.net/2/

2.3 FEDERATION OF WEB SERVICES AND "SOCIAL LOGIN"

Federation of web services, be it, e.g., social career platforms (e.g., LinkedIn, Xing) or code repository services (GitHub, etc.) is based on connection to trusted (usually federated) identity providers (see, e.g. [12]). Using social network services or big online stores and service providers (e.g., Facebook, Google, Amazon, etc.) as an identity provider, it is nowadays possible for an user to log into web services and portals with the credentials he uses to access the identity provider's main services (e.g. social network); we simply call this "social login" below.

Social login negates the need for the end user to remember login information for multiple web sites and services, while providing site owners with uniform demographic information as provided by the identity provider (e.g. basic data from the users' Facebook or Amazon profiles). Many consumer sites which use social network-based login mechanisms also keep a more traditional online registration and login mechanism for those users who either desire it or have not an account at a compatible social networking service.

The social login can be implemented strictly as an authentication system using standards such as OpenID or SAML. For consumer websites that offer social functionality to users, social login is often implemented using the OAuth standard. Sites using the social login in this manner typically offer social features such as commenting, sharing, reactions and gamification.

While social login can be extended to some corporate websites, in the context of strictly secure applications it is often impossible to trust a third-party identity provider. For this reason, social login is generally not used for highly secure applications such as those in banking or health. For applications such as forum, e-commerce or career network websites, etc., usage of social login is strongly increasing.

2.4 AAI SYSTEM FOR THE LEXIS PLATFORM: OBJECTIVES

The design of LEXIS AAI has been initiated as part of the co-design activities carried out in work package WP2; the used approach has been first described in deliverable D2.1 [1] and then refined in deliverable D2.2 (Section 3.1 — *Federated Identity & Access Management Layer*) [13].

Basically, the goal is to provide a federated Authentication & Authorization Infrastructure (AAI), which is able to manage the access to the LEXIS platform by users which can use resources in two different data centres (IT4I and LRZ).

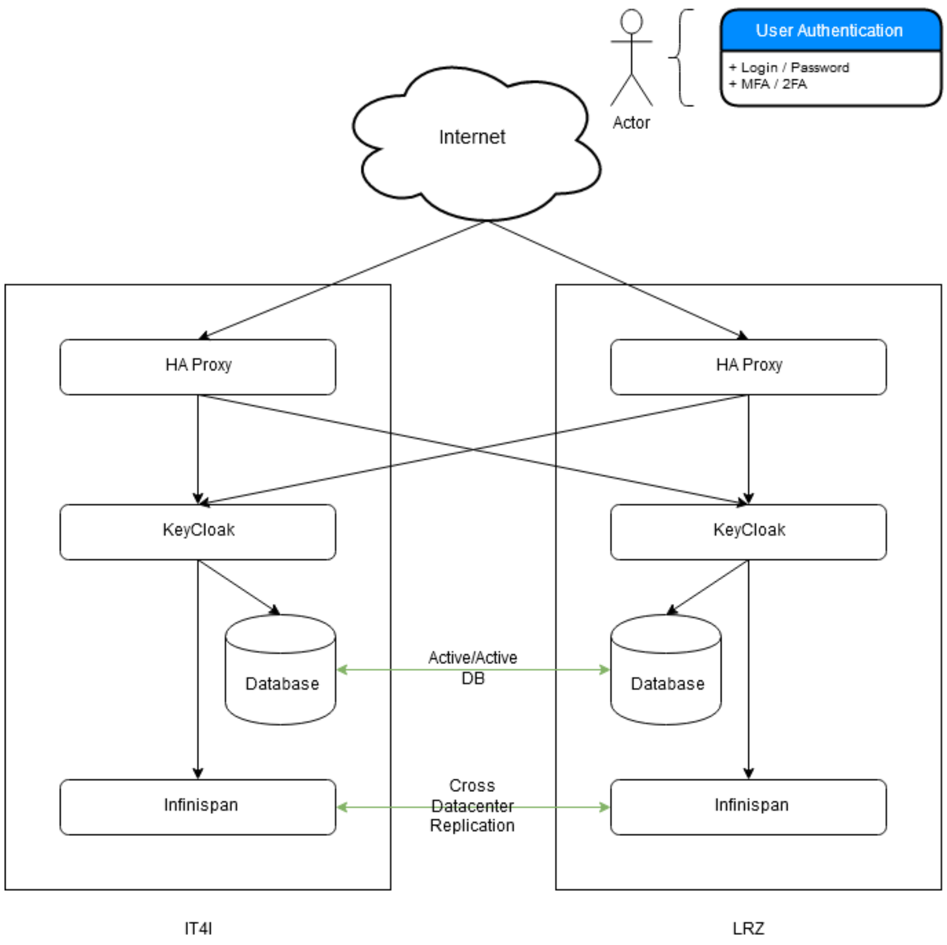


Figure 2 Cross-site AAI architectural configuration

As a key element in the co-designed LEXIS architecture, the AAI is linked to three different parts of the LEXIS infrastructure, which also pose the main requirements to be met by the AAI:

- **LEXIS Portal**: The portal with its front- and backend will expose LEXIS functionalities and features to the end-user. We can consider work packages WP5, WP6 and WP7 as our very first users´ beta-testing the LEXIS infrastructure (including the AAI system) via the portal.
- **LEXIS Computational Resources**: Computational layer of the LEXIS project which is hosted at two different supercomputing centres (IT4I and LRZ) and based on two different types of systems in each centre: HPC resources and OpenStack-based Cloud resources. The AAI system will support computing via the orchestration system, which is based on the software packages YORC (YSTIA orchestrator) & Alien4Cloud (provided by partner Bull/Atos) and the HEAppE middleware (provided by partner IT4I).
- **LEXIS Data Layer**: Data system of the LEXIS platform. This system actually spans three different sites (i.e., IT4I, LRZ and ECMWF) and is based on several storage solutions, such as iRODS and WCDA, where each centre can have its own storage backend (CEPH, SAN, NAS, etc.).

The reader can find additional details on the LEXIS infrastructure and technology selection in deliverables of WP2, specifically D2.1 [1] and D2.2 [13]. Within this ecosystem, the LEXIS AAI will be set up with the target of deploying a fully operational module, following best recommendation and practice for production deployment. Figure 2shows the cross-site architectural configuration for the devised AAI system, which allows to keep information synchronized across different infrastructures (i.e., data centres).

As reported in the following sections, the AAI will achieve its main objectives by integrating a set of key technologies which have been selected in such way they can cover all the requirements and specifications provided as input by WP2, WP3, WP4 and WP8. The overview given in this section shall be of help in understanding the in-depth analysis of all relevant technologies laid out in the following sections.

3 METHODOLOGY

This section is devoted to introducing the analysis and selection methodology (which comprises also the collection of requirements and specifications, as well as the analysis of state-of-the-art solutions) that we applied in LEXIS to define the set of all relevant technologies and tools for creating the AAI layer.

The applied methodology for the definition and design of the AAI system is as follows:

- Collection of requirements/specifications: First of all, we collected all the requirements/specifications with respect to the different building blocks of LEXIS infrastructure, i.e., the computational resource layer, the data layer, the network layer and the portal.
- Definition of evaluation criteria: Based on the expertise provided by all partners on the different technologies being used in HPC and Cloud environments, we have defined a list of evaluation criteria which includes (but is not limited to) high availability, clustering/scalability, authentication protocols, authorization services, APIs offered.
- Selection of open-source AA providers/systems: We have decided to use open source solutions and perform literature/internet research to identify the most robust, flexible and popular solutions that could fit our requirements/specifications.
- Analysis: We performed a first evaluation of features / usability of several possible solutions.
- Filtering phase: According to the evaluation criteria defined, a subset of the analysed solutions was selected: we extracted three solutions to be evaluated more in depth.
- Benchmarking selected solution(s): Benchmarking the selected possible solutions through the setup of a PoC (AA system) in the context of the LEXIS project is time- and resource-consuming as we have to integrate the solution with the different LEXIS building blocks (HPC and Cloud systems, Storage, …). We thus decided to fully benchmark only the most promising (cf. Table 1 and Section 6) system – Keycloak – further, while considering the other two solutions as back-up possibilities in case that Keycloak fails our evaluation criteria.
- Selecting a solution: Among various tested solutions and technologies, we found that Keycloak provides all the key features required to cover both requirements and specifications of the LEXIS project. This impression also pertained through the benchmarking phase.

The following subsections provide more details concerning the various phases of the used methodology.

3.1 COLLECTING REQUIREMENTS AND SPECIFICATIONS

The first step for selecting a proper solution to implement the AAI layer is the collection of all the requirements and specifications that the solution(s) will have to address. To this end, we collected requirements from all work packages, aiming at satisfying the requirements with respect to all building blocks. This was done with support of the co-design committee (WP2) to ensure success.

The first set of requirements are coming from the data centres (IT4I and LRZ) and their existing users. As a matter of fact, the partners will not change their own authentication and authorization system on their working infrastructures and the LEXIS AAI must be able to redirect authentication on the existing authentication systems.

The second set of requirements comes from the chosen software technologies for LEXIS platform, which mainly are YORC & Alien4Cloud (provided by Bull/Atos) for orchestration, HEAppE (provided by IT4I) as HPC middleware, OpenStack as cloud-management system and iRODS as data-management system.

The last set of requirements comes from the Portal layer based on new and flexible technologies, which can be, of course, somewhat adapted. Section 4 (Requirements and Specifications), will give more details on all the requirements that have been gathered.

3.2 DEFINITION OF EVALUATION CRITERIA

In order to be able to make an initial analysis of the Authentication and Authorization system, we have created a list of criteria covering the following areas:

- **Backend authentication system supported**: In order to integrate existing IT4I and LRZ users, the LEXIS AAI must support site-specific authentication backends.
- **Frontend authentication protocols/APIs supported**: In order to provide AAI for the LEXIS layers (Compute, Data, Portal), the LEXIS AAI must provide APIs/protocols such that the software that will be used in the LEXIS components can connect to the AAI.
- **Productivity/Deployment**: In order to constitute a solid pillar for the LEXIS project, the AAI system must have basic characteristics considered state-of-the-art in IT, such as High Availability, Scalability (with Clustering and Distributed System concepts), Resilience (with Backup and Disaster Recovery concepts), Manageability and Security.

3.3 ANALYSIS OF POSSIBLE SOLUTIONS

The Open-source community provides lots of solutions for Authentication, Authorization, Federation, Identity and Access management, accessible on the Internet. In addition to our current expertise and knowledge on existing Identity and Access Management (IAM) solutions and due to the fact technologies are emerging or evolving fast, a research of existing technologies and solutions/products has been done through various sources, literature, internet, and existing standards.

We have identified a set of 10 open source frameworks that can be used as IAM solutions, providing Authentication, Authorization and Federation with Single-Sign-On. All solutions can be found in a spreadsheet that is provided in Appendix A.

Among the solutions listed in Appendix A, we have further taken into account the three most promising ones in terms of the requirements defined above (Section 3.2):

- **Keycloak**: “Keycloak is an open source IAM solution also providing Single-Sign-On functionality and supporting most common identity provider (IdP). This project is supported by RedHat.” [7]
- **OpenStack Keystone**: “Keystone is an OpenStack service that provides API client authentication, service discovery, and distributed multi-tenant authorization by implementing OpenStack Identity API (the Identity service for OpenStack).”[8]
- **Unity**: “Unity is an open source solution IAM solution providing federation and also inter-federation. Unity supports a lot of endpoint and external Identity Providers (IdP) making the solution really flexible.”[9] For instance, Unity is used by B2ACCESS service from EUDAT[10].

3.4 PROOF OF CONCEPT AND SELECTION OF KEYCLOAK

After a detailed comparison among these three solutions, we finally decided to benchmark Keycloak, since it appears to cover all the requirements and specifications collected, while providing some additional benefits with respect to OpenStack Keystone and Unity.

The PoC with Keycloak had the purpose of excluding major obstacles to the usage of Keycloak in the LEXIS ecosystem. Due to the success of the PoC, Keycloak remained the selected solution.

[7] Keycloak: https://www.keycloak.org/about.html
[8] OpenStack Keystone web site: https://wiki.openstack.org/wiki/Keystone
[9] Unity IDM: https://www.unity-idm.eu
[10] EUDAT: B2ACCESS: https://eudat.eu/services/userdoc/b2access-service-integration

In the very unexpected case the problems with Keycloak hinder the progress of LEXIS, we have – with the remaining two frameworks mentioned above – back-up solutions which may be employed instead.

4 REQUIREMENTS AND SPECIFICATIONS

This section contains the synthesis of the requirements that have been gathered for the three main building blocks of the LEXIS platform which will interact with the LEXIS AAI:

- **Compute Layer**: This includes Cloud resources (with OpenStack) and HPC resources with all the relevant software tools;
- **Data Layer**: This includes the DDI (Distributed Data Infrastructure) and the selected software tools;
- **Portal Layer**: This refers to the LEXIS portal (or its backend) connecting to LEXIS AAI.

It is noteworthy that the requirements with respect to the first two layers (Compute and Data) contain the constraints coming from the existing infrastructure at the IT4I and LRZ data centres. Most importantly, those parts of existing cloud and HPC infrastructure which are production systems cannot be reconfigured simply to accept the LEXIS user identity provided by the AAI. Each centre operates its own AAI solution and manages its own set of user identities.

Affected resources are mainly HPC clusters at IT4I and LRZ and the OpenStack Cloud deployment at LRZ. The OpenStack instance at IT4I relevant to LEXIS will not be part of the operational IT4I infrastructure and thus can be used to test various experimental configurations.

Access to operational resources of the supercomputing centres is usually granted in the frame of computational projects proposed by users. Such projects, when approved, are assigned a project ID and often also a Unix group on a HPC system. User accounts of users taking part in a project activities endeavour are granted membership in the respective computational project for these activities. The project ID essentially is a unique identifier of an allocation of particular amount of computational resources in a particular centre. The project and respective ID can be obtained, depending on the computing centre, via supercomputing centre administration (Director’s discretion), by submitting a proposal to Open Call for computing resources, on commercial basis, etc. Each project ID is associated with its own project principal investigator (PI) who usually has the privilege to allow another supercomputing user identity to be associated with “his” project ID, thus allowing the user to use the project’s resources.

In the LEXIS AAI framework, each LEXIS user identity is, first, associated with a particular LEXIS user group (e.g. research institute or collaboration). Such a group can then have one or more LEXIS computational projects. These are an abstraction provided by the LEXIS platform which can encompass various project IDs (resource allocations) in individual supercomputing centres. Each LEXIS computing project is associated with particular set of workflows and datasets (based on roles provided by the AAI).

Mapping of the supercomputing centres project IDs (resource allocations) to particular LEXIS computing project will be done by a dedicated approval process, where the LEXIS user has to ask the project PI in the particular centre for permission. Once this association is approved, the LEXIS users associated with the given LEXIS computing project can use the resources provided by the particular supercomputing centre’s supercomputing project. However, this mapping does not create a direct link between the LEXIS user identity and user identity in the given HPC centre.

Mapping of the LEXIS user identity to the supercomputing user identity will be done by a specialized secured middleware (such as HEAppE for HPC [14]). The centre user to which the LEXIS user is mapped is represented by a special user account associated with a particular project ID as per approval of the project PI. Thus, the project PI does not need a LEXIS user account whatsoever.

An approval system which will help to automate the association between the LEXIS computing project and projects in the individual supercomputing centres will be created in the scope of the Pilots (WP5-7) with the help of the technical work packages.

4.1 CLOUD COMPUTATIONAL RESOURCES (OPENSTACK)

OpenStack is the main technology within the Cloud part of the LEXIS platform, and it is installed on LEXIS federated infrastructure (in both IT4I and LRZ centres). OpenStack provides its own service for authorization and authentication, namely OpenStack Keystone, but it can also rely on other existing authorization and authentication services such as Keycloak.

OpenStack actually supports several Identity backends, such as LDAP, but it also supports two models for federation identity, especially the one called *KeyStone as a Service Provider* which uses an external identity provider (such as Keycloak or Google) as identity source and authentication method [15]. Also, OpenStack KeyStone supports offloading authentication using SAML2.0 and OpenID Connect protocols. OpenStack has a web-based API which supports authentication through the OpenID protocol. In case that a particular OpenStack instance cannot be directly configured to accept LEXIS users authenticated by the LEXIS AAI (which is, e.g., the case for the LRZ Cloud), a mapping between the LEXIS user and a local supercomputing centre user has to be done according to the pattern described above. In this case, a specialized middleware such as HEAppE can directly provide tokens created for the local centre user accounts without the need to expose their credentials to the rest of the platform. Each component of the platform which would need to access the OpenStack API can obtain a valid token from HEAppE beforehand. The token will be issued for the mapped local user on behalf of the LEXIS platform user, but without its knowledge.

It is important to highlight that the HEAppE middleware currently implements its own authentication mechanism.

HEAppE provides a mapping between the LEXIS user and supercomputing centre accounts, which also includes accounts that are used to access the production OpenStack instance. Therefore, the new HEAppE extension has to be able to obtain an OpenStack Keystone token[11] for a particular supercomputing centre user. Parts of the LEXIS platform (mainly YORC) will use the HEAppE API on behalf of the LEXIS user identity which is authenticated against the LEXIS AAI. HEAppE will map the LEXIS identity to a particular supercomputing centre account and will obtain OpenStack API token for this user. The OpenStack API supports two formats of tokens - Fernet [16] and JWS [17], both of which can be provided by the OpenStack Keystone service. Once HEAppE obtains token for the mapped user, it will provide it to the LEXIS service which wants to use the OpenStack API (such as YORC). The tokens shall be configured in a way that it does not reveal the identity of the mapped user. In this instance, HEAppE will serve just as a proxy that provides the mapping between different user groups and authentication protocols.

4.2 HPC COMPUTATIONAL RESOURCES

Access to the HPC resources usually conforms to various security regulations and is almost exclusively realized through SSH with RSA key pair. Traditional HPC infrastructures usually do not support web based protocols such as OpenID or SAML and manage their own identities (as described above). Mapping between the LEXIS user identity and supercomputing centre identity will be realized by a specialized middleware called HEAppE, which provides an API for the interaction with an HPC system (job submission, control and monitoring).

The API will be accessed by components of the LEXIS platform which are authenticated against the LEXIS AAI. Then, the individual API calls on behalf of the LEXIS users are internally mapped to a supercomputing centre identity under which the actual interaction with a HPC cluster is realized. HEAppE stores the supercomputing centre identities (login and a private RSA key) in a Vault which is secured as a part of the LEXIS AAI. Thus, this solution conforms to

[11] OpenStack Keystone token: https://docs.openstack.org/keystone/latest/admin/tokens-overview.html

the AAI requirement to support the mapping between the LEXIS user and user accounts of the individual supercomputing centres.

It is important to highlight that the HEAppE middleware currently implements its own authentication mechanism. Support for OpenID or SAML protocols will be added by IT4I, based on the final AAI solution selected in this deliverable.

4.3 ORCHESTRATION LAYER (YORC)

YORC is the solution selected as the orchestration tool in LEXIS. It basically provides the orchestration service (YORC) and a frontend interface, namely Alien4Cloud. Here, we refer to YORC as the combination of the orchestration service and the frontend. YORC supports any SAML identity provider or LDAP server for authentication through its frontend interface (Alien4Cloud).

However, YORC does not support any external provider for authorization. In order to have a complete federated identity management system providing both authentication and authorization, we will need to find a way to map YORC roles to the authorization grants of the LEXIS AAI system. The creation of a YORC plugin/extension to rely on LEXIS AAI authorization or other solutions are currently considered in order to realize this.

To summarize, the technical requirements for the LEXIS AAI frontend authentication to be compatible with the orchestration layer are as follows:

- SAML2.0;
- or LDAP.

4.4 LEXIS DATA LAYER

The DDI (Distributed Data Infrastructure) as central part of the LEXIS data layer, will use the iRODS[12] system for distributed data management. iRODS uses by default a password system for user authentication, storing usernames and hashes in its so-called iCAT database (essentially a relational database that is used to store also meta-data of iRODS). Other authentication methods are supported via specific plugins [18]:

- **GSI**: Specific authentication plugin for Grid Security Infrastructure;
- **Kerberos**: Usage of a Key Distribution Centre and a Kerberos admin server;
- **PAM**: Linux Pluggable Authentication Modules is supported by iRODS and it can be configured to support LDAP server;
- **OpenID**: An experimental OpenID plugin exists for iRODS that allows support for OpenID protocol.

To summarize, the requirements for DDI system are as follows:

- Kerberos support for LEXIS AAI frontend authentication;
- or LDAP support for LEXIS AAI frontend authentication;
- or OpenID protocol support for LEXIS AAI frontend authentication protocol.

Although iRODS plugins can cover all the current needs of LEXIS project, potentially other ones will be developed in order to facilitate and automate integration of iRODS within the LEXIS platform.

As the second part of the LEXIS data system, the WCDA (Weather and Climate Data API) developed within LEXIS WP7 will be newly designed to use LEXIS AAI as its sole security reference for authentication and authorization. Thus, it does not pose any additional requirements to the AAI system.

[12] iRODS: https://irods.org

4.5 LEXIS PLATFORM PORTAL

The LEXIS portal (with its backend) will be flexible and adapt to the LEXIS AAI. As a matter of fact, the WP8 partners have expressed a preference for adopting commonly used software and protocols in IT such as SAML2.0 or OpenID Connect.

5 AAI SOLUTION: EVALUATION CRITERIA

The evaluation of potential solutions for implementing the LEXIS AAI requires the definition of a set of criteria that sets the basis for their comparison. The criteria used here to analyse and compare the different AAI solutions are the following:

- **License**: All the considered solutions being open source, this criterion is only used to document the license under which the corresponding solution is released;
- **Requirements:** It is the set of hardware and software requirements for the given solution; e.g., some solutions require Java 8 JDK, while for others just the installation of a RDBMS is necessary to deploy the system;
- **Clustering/Scalability**: High availability is one of the key features for a distributed system. A clustering set-up increments flexibility and reliability over failures and allows fail-over mechanisms;
- **Distributed/Multi-site**: Load balancing and multi-node deployment allow for a capillary coverage of the system network and generally better performances for the whole system;
- **Disaster Recovery/Backup**: This criterion is used to identify error recovery and backup functionalities, able to restore the state of the system after a fault occurred;
- **Authentication protocols**: As one of the most important criteria, we consider authentication and authorization protocols used in or supported by the solution. The analysed systems rely mostly on common standard protocols: SAML (generally 2.0), OpenID, OpenID Connect, OAuth (versions 1.0a [7, 8] and 2.0 [9]). Commonly used protocols are also Kerberos, WS-Federation (Passive), and others;
- **Authentication Integration**: This criterion comprises the functionalities offered with respect to integration with authentication mechanisms, including usually: default username & password login (against local database), X.509/SSL certificate-based authentication, LDAP authentication, OTP/TOTP, MFA (2FA), social networks login. In some cases, the systems evaluated make use of other, pre-defined mechanisms/ frameworks (e.g., Shibboleth IDP, Spring Security framework [18], FIDO, and others);
- **Authorization Services**: The authorization scheme adopted by the system can resemble e.g. RBAC, ABAC or UBAC models. Some solutions provide more advanced policy schemes, such as through XACML or Advanced Hybrid RBAC, offering a hybrid approach between RBAC and ABAC. Implementations based on ACL (Access Control List) or HBAC (Host and service-based Access Control model) are less common;
- **Functionalities**: Relevant additional functionalities that may be offered by some implementations, such as caching; users grouping and management; lifecycle management for users, groups, identities and roles;
- **Auditing**: Logging of all the security-relevant events and actions performed on the system by both users and administrators is a fundamental requirement from security standpoint;
- **Storage**: Storage type and features used by the system;
- **Admin API**: API functions and UI available for administration purposes. It usually involves user management, roles assignment, system administration;
- **User API**: API functions and UI available for the end-user, to manage their account and system resources they have access to;
- **Self-registration/User validation**: Option that enables the end-user to create and activate autonomously their new account. This functionality can lighten the administration overhead created by hand-checking every new user, but may – depending on the implementation of the process – have a somewhat negative impact on security;
- **Comments**: Any other relevant functionality or characteristics not covered by the previous points.

In the following subsections, we detail somewhat on the most relevant aspects of the above criteria, covering first AAI-related criteria and then productivity criteria.

5.1 CRITERIA FOR AUTHENTICATION, AUTHORIZATION AND IDENTITY

The authentication scheme should rely on standard protocols, as it will ensure compatibility with a larger set of Identity and Service Providers. Compliance to open standards is crucial in defining several authentication interfaces able to connect in different contexts.

The authorization scheme usually adopts the RBAC model, which provides flexibility over policy definition and enforcement thanks to an easy role management; rules are static, direct and easy to visualize, allowing security admins to directly see the users and resources that they will affect when creating or modifying a policy. However, in some contexts like huge companies, this scheme could lead to role explosion, due to the high number of different roles which must differentiate with each other depending on the characteristics of the user. Thus, a mitigated solution is preferable, like the RBAC/ABAC hybrid solution proposed by IDM365, in which some of the policy mechanics are determined with some additional logic operating on the singular attributes of each user (e.g., office location, production area, etc.).

5.2 CRITERIA FOR PRODUCTIVITY

The LEXIS AAI system will be one of the pillars of LEXIS infrastructure and as such it *must* provide strong capabilities for supporting productivity and deployment. These are considered basic and fundamental capabilities for such IT system nowadays. Thus, the LEXIS AAI must be:

- Available or highly available, in order to ensure that any user, process, device can authenticate and get authorized to run their tasks/jobs at any time;
- Scalable, in order to avoid any impact on the production system according to the load on the system;
- Resilient, in order to avoid data loss or not being able to recover from error, dysfunction or disaster;
- Manageable and secure, in order to keep the system properly running in production and protected from threats.

It is worth mentioning that the last two criteria (i.e., resiliency and easy management / security) usually have an impact on the IT administration team (usually including the security team), while the first two (availability and scalability) affect the architecture designers.

6 SELECTED SOLUTIONS

Following the approach explained in the sections above, we selected three solutions as a possible basis for implementing the LEXIS AAI. These provide the most interesting features, addressing all the requirements and specifications posed by the LEXIS platform design. All three solutions are discussed in this section, while we selected and benchmarked only the most promising one through the implementation of PoCs (Section 7).

Sections 6.1-6.3 detail somewhat on the three evaluated systems, and Section 6.4 concludes the comparison and selection of our preferred solution.

6.1 KEYCLOAK

Developed by JBoss (a Red Hat division), Keycloak[13] is an open source Single Sign-On solution with Identity and Access Management. It also provides Single Sign-On functionalities, managing user's logout in place of the applications belonging to the same realm. Keycloak also has built-in support to connect to existing LDAP or Active Directory servers or custom providers in other stores, such as a relational database (i.e., a RDBMS such as SQL Oracle or PostgreSQL).

Besides the Standalone Mode, the program provides 3 different operating modes supporting clustering:

- **Standalone Clustered Mode**: It requires a copy of the Keycloak distribution on each node in the cluster;
- **Domain Clustered Mode**: It provides a central place to store and publish configurations, common to all the nodes in the cluster;
- **Cross-Datacentre Replication Mode**: It allows to run Keycloak in a cluster across multiple data centres, typically located in different geographical regions; each data centre has its own cluster in this mode.

Keycloak provides a replication mechanism, achieved by the usage of distributed caches among the nodes in the cluster. A number of nodes (configurable as an attribute in the cluster's settings) are chosen as owners of that data, which are indeed not replicated to every single node in the cluster; thus any node has to query the cluster to obtain a specific cache entry that it doesn't own. For this reason, the availability of a specific piece of data is related to the nodes that are hosting it: if all those nodes go down, the data is lost permanently. In these cases, the users will be logged out automatically and asked to login again.

Keycloak provides client adapters for several platforms and programming languages, but it is built on standard protocols like OpenID Connect, OAuth 2.0 and SAML 2.0, thus allowing integration with any application and service.

The authentication process is built upon different mechanisms: local user with password policy, OTP policy, Kerberos, X509 certificate + LDAP + Kerberos. From the viewpoint of authorization policies, Keycloak supports the following ones: Attribute-based access control (ABAC), Role-based access control (RBAC), User-based access control (UBAC), Context-based access control (CBAC), Rule-based access control, Time-based access control + Support for custom access control mechanisms (ACMs) through a Policy Provider Service Provider Interface (SPI).

Keycloak also provides Identity Brokering features, as it provides social login and authentication with existing OpenID Connect or SAML 2.0 Identity Providers, configurable through the administrative console, which provides functionalities for user management too. Also, Keycloak provides a rich set of auditing capabilities, recording every user login action or even admin actions (e.g., configuration changes), which can be stored in the database and reviewed in the Admin Console. Furthermore, plugins can listen for these events through an additional listener SPI, allowing to interact with it and perform appropriate actions. Built-in listeners offer some basic auditing features, including a simple logger and the ability to send an email when specific events occur.

[13] Keycloak: https://www.keycloak.org/

It is also possible to enable User self-registration. Some further comments can be found in StackHPC's article on Federation and identity brokering using Keycloak [19].

6.2 OPENSTACK KEYSTONE

Among the OpenStack services, Keystone is the identity service offering authentication and authorization mechanisms, such as API client authentication, service discovery, and distributed multi-tenant authorization by implementing OpenStack's Identity API [14].

Keystone maintains a central directory which keeps a mapping of the users with the services they can access, but it can also integrate existing backend directories (e.g. LDAP). It supports multiple forms of authentication including standard username&password credentials, token-based systems, multi-factor authentication (MFA), time-based one-time passwords (TOTP), HTTPD authentication for mod_mellon and mod_shibboleth, X.509 Tokenless authorization. It supports all standard protocols such as LDAP, OAuth, OpenID Connect, SAML and SQL. Additionally, users and third-party applications can determine accessible resources through a queryable list of all of the services deployed in the corresponding OpenStack Cloud.

Like most of OpenStack projects, Keystone defines Role-based Access Control (RBAC) policy rules. However, group-based role assignments are needed to facilitate federation of users by the Identity Service: groups objects will be defined, mapping all the belonging users to their local role assignments.

Keystone provides enhanced auditing capabilities through the implementation of the PyCADF library, capable of uttering notifications according to the DMTF CADF specification [20]: this standard provides "compliance with security, operational, and business processes and supports normalized and categorized event data for federation and aggregation" [15].

There are two supported clients: *python-keystoneclient* project providing python bindings and *python-openstackclient* providing a command line interface.

6.3 UNITY

Unity[16] is an authentication service with Single Sign-On (SSO), providing identity management capabilities and offering federation and inter-federation management features. It can be configured to integrate a storage backend, which can be:

- A typical relational database backend (RDBMS), such as SQL, MySQL, H2;
- Hazelcast distributed in-memory data grid — overlay over RDBMS, offering in-memory operations computing, optimal for clustering and managing large traffic of data.

Unity does not provide auditing functionalities yet. Nevertheless, it provides management of identities and entities, groups and attributes. Authorization is indeed based on Role-based Access Control (RBAC), also supporting authorization on group level. Unity provides an Authentication system based on OAuth 2.0, OpenID Connect, SAML endpoints (Web and SOAP) and External LDAP, allowing users to login using password or X509 certificate [21].

The Web Admin UI facilitates the operations of server management. The most important features of the Web Admin UI are:

[14] OpenStack Keystone: https://www.openstack.org/software/releases/ocata/components/keystone
[15] OpenStack Auditing: https://docs.openstack.org/mitaka/config-reference/identity/auditing.html
[16] Unity IDM: https://www.unity-idm.eu

- Management of attribute types, attribute classes, credential types and credential requirements (Schema management tab);
- Possibility to manage groups, their attribute classes and attribute statements (Contents management tab);
- Control over entities and identities and their group membership (Contents management tab);
- Full attribute control (Contents management tab);
- Management of registration forms, possibility to list them instantly from the Web Admin UI and to manage the received requests (Registrations management tab);
- Possibility to create and load database dumps and to browse and trigger reconfiguration of endpoints, authenticators and translation profiles (Server management tab).

On the other hand, ordinary users can manage their profiles through the Web User Home UI, a simple interface to update their credentials and information about them.

User registration is allowed through customizable registration forms, that can be used to collect enrollement information about the user (typical use case) or retrieve it from remote IDP (in case the user has been authenticated in such way), simply defining automated actions to be performed on newly created accounts.

6.4 COMPARISON OF SOLUTIONS AND SELECTION FOR POC

Chyba! Nenalezen zdroj odkazů. summarizes the pros and cons we found in the three solutions when evaluating them in appropriate depth.

Table 1 Comparison of the Keycloak, OpenStack Keystone and Unity AA solutions

	PROS	CONS
KEYCLOAK	• Clustered and Distributed deployment • Authorization Capabilities (ABAC, RBAC, UBAC, CBAC) • Authentication frontend with SAML 2.0 • Auditing capabilities	• Lack of LDAP or Kerberos frontend protocols
OPENSTACK KEYSTONE	• OpenStack ready • Auditing capabilities (CADF) • Authentication frontend with SAML 2.0	• Lack of LDAP or Kerberos frontend protocols • Limited authorization capabilities RBAC
UNITY	• Authentication frontend with SAML 2.0	• Lack of LDAP or Kerberos frontend protocols

It is evident that Keycloak shows some advantage with respect to the other solutions. We would like to mention that Keycloak has been selected for being the Authentication and Authorization solution in several similar projects such as "StackHPC Ltd" [19, 22]. Due to these reasons, we clearly decided for Keycloak to be deployed for a PoC test.

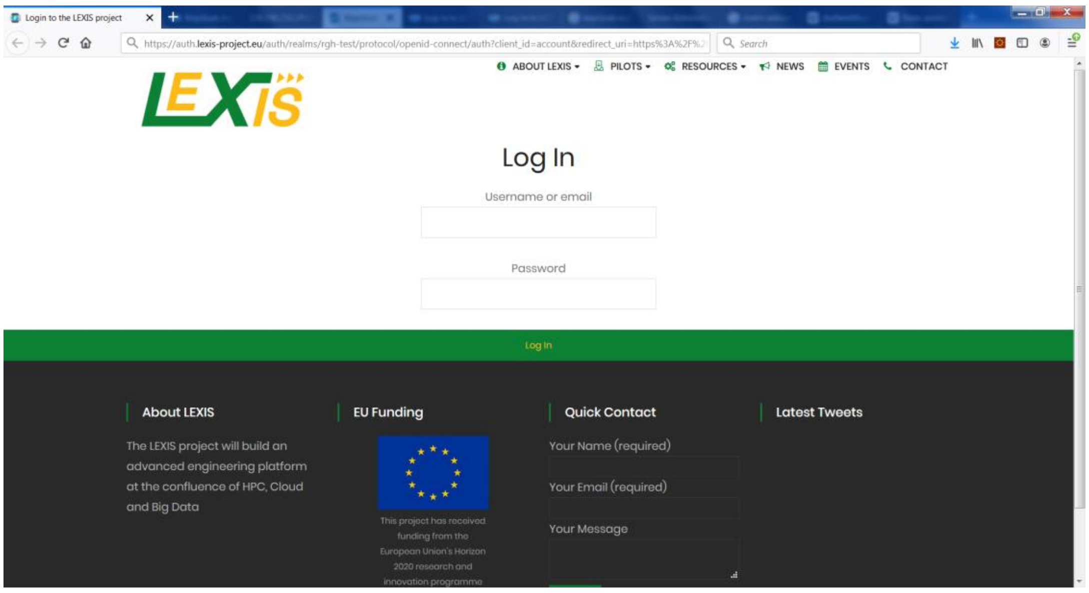
7 FINAL PROOF-OF-CONCEPT TEST AND LEXIS AAI SOLUTION

We have deployed two Keycloak installations in *Standalone* mode (i.e., each data centre —IT4I and LRZ— has its own installation of Keycloak) in order to perform integration tests with all the components of the LEXIS infrastructure. Those tests are part of the validation of the proof-of-concept (PoC) with Keycloak. Work package members have an access and can configure a specific "realm" for their own testing purpose.

In the following text we illustrate the configuration of Keycloak to fit example LEXIS systems and lay out the plans for the final LEXIS AAI solution.

7.1 CONFIGURATION WITH RESPECT TO LEXIS PORTAL

Figure 3 shows a screenshot where LEXIS portal authentication is done via Keycloak (in a way practically invisible to the user).


Login to the LEXIS project
https://auth.lexis-project.eu/auth/realms/rgh-test/protocol/openid-connect/auth?client_id=account&redirect_uri=https%3A%2F%
Search
ABOUT LEXIS
PILOTS
RESOURCES
NEWS
EVENTS
CONTACT

Log In

Username or email

Password

Log In

About LEXIS

The LEXIS project will build an advanced engineering platform at the confluence of HPC, Cloud and Big Data

EU Funding

This project has received funding from the European Union's Horizon 2020 research and innovation programme

Quick Contact

Your Name (required)

Your Email (required)

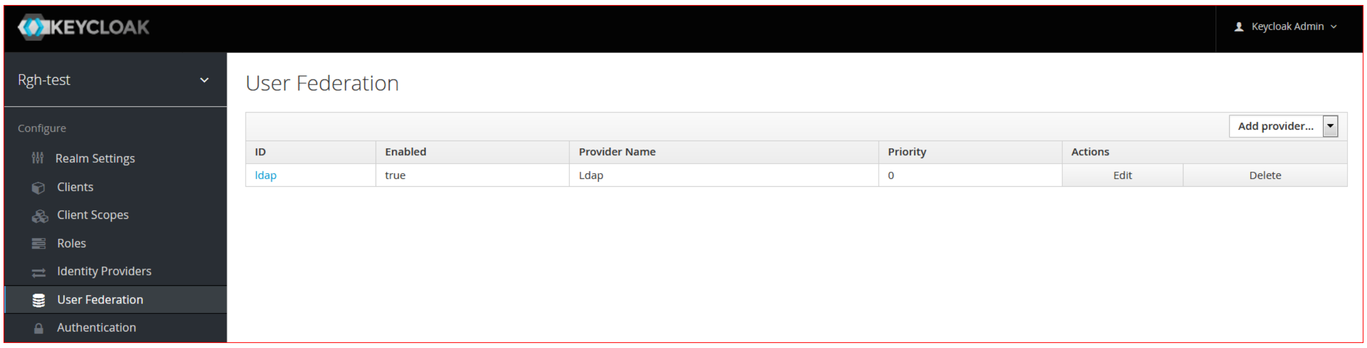
Your Message

Latest Tweets

Figure 3 LEXIS portal login (web) page

7.2 CONNECTION OF KEYCLOAK TO LRZ LDAP BACKEND

As a proof-of-concept component with respect to connecting to the LDAP AAI system of an HPC centre, Figure 4 shows a Keycloak configuration for using the LRZ LDAP server as authentication backend.



User Federation

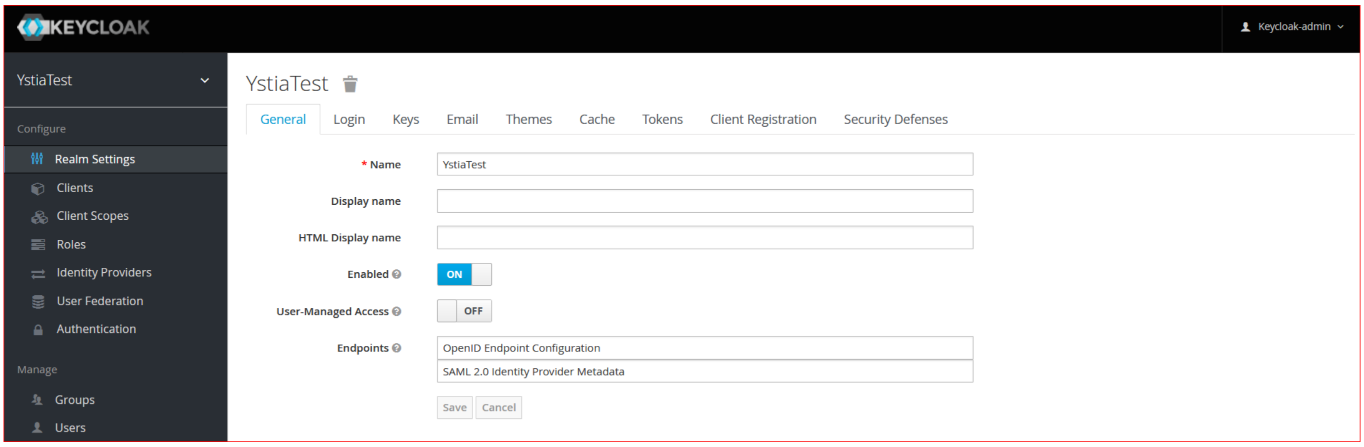
ID	Enabled	Provider Name	Priority	Actions	
ldap	true	Ldap	0	Edit	Delete

Figure 4 Redirection to the Keycloak-based AAI — LDAP backend

We used the "User Federation" feature of Keycloak adding the LDAP backend server on read-only mode for LEXIS users.

7.3 CONNECTION OF YORK TO KEYCLOAK

Figure 5, in turn, shows the Keycloak configuration for YORC, which needs the SAML2.0 protocol. We thus have configured both the SAML2.0 and OpenID connect protocols in.



* Name	YstiaTest
Display name	
HTML Display name	
Enabled	ON
User-Managed Access	OFF
Endpoints	OpenID Endpoint Configuration SAML 2.0 Identity Provider Metadata

Figure 5 YORC orchestration layer — Keycloak configuration

7.4 PLANS FOR FINAL LEXIS AAI SYSTEM

The PoC showed no major obstacles, leaving Keycloak as the preferred solution to implement the LEXIS AAI.

In addition, open design choices for the LEXIS AAI system as for role/permission scheme and for the high-availability setup have been clarified within the Proof-of-Concept phase and shall be reported here.

Firstly, based on the integration tests made in both data centres with all different software tools and components, we started working on a Role and Permission scheme for authorization, with the basic concept of three permissions:

- **List**: Users, processes or devices are able to list a resource and get its details; e.g., name, creation date, etc. We can refer to such details as the meta-data of the resource;
- **Read**: Users, processes or devices can access the resource in read-only mode;
- **Execute**: Users, processes or devices can execute actions on the resource such as creation, update, deletion.

For instance, if we take a simple file as sample, the list permission allows accessing the file information/properties such as name, author, type of file, size, dates (created, modified, accessed), read permission allows reading the content of the file, and the execute permission allows creating, updating, deleting, changing permission of the file.

Secondly, a redundant setup scheme for Keycloak at IT4I and LRZ for high availability was chosen. Keycloak documentation provides a nice architecture for a distributed architecture on two different data centres that we would like to put in place with minor modification (see Figure 6). Load balancing functionality shall be ensured as much as possible in the context of LEXIS and this setup, using e.g. HAProxy or other frameworks.

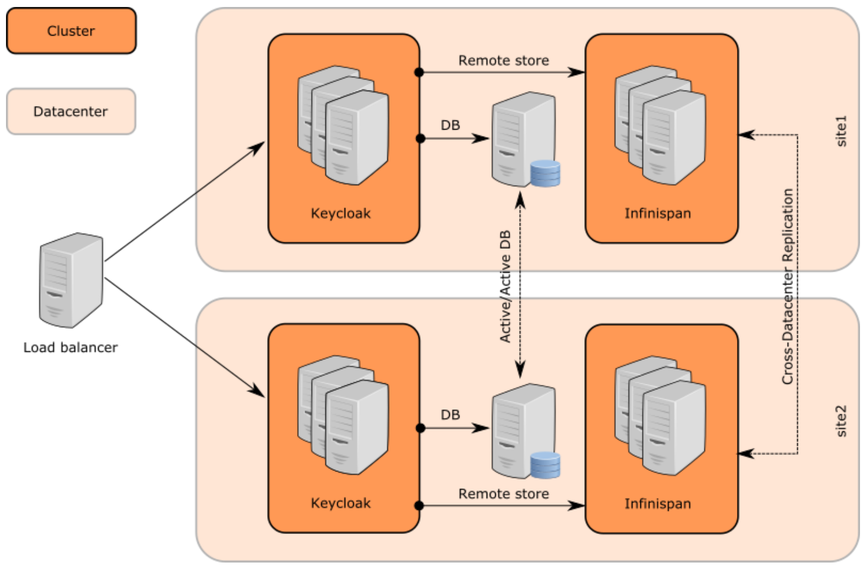


Figure 6 Two sites Keycloak configuration also supporting high availability [source: Keycloak web site]

8 CONCLUSIONS

Authentication and Authorization Infrastructure (AAI) is one of the key components of the LEXIS platform, which in turns aims at federating both HPC and Cloud-based computing resources, along with a distributed data storage system. Ensuring the access to computing, networking and storage resources, as well as data available on the platform with fine-grained rights management for users is mandatory. HPC centres introduce additional requirements with regards to access to HPC clusters, where the LEXIS AAI system has to interact with lower-level permission systems implementing compute-time grants for users´ projects. All these points and other requirements coming from pilot applications (WP5, WP6 and WP7), the portal design team (WP8), the orchestration layer (WP4), the distributed data infrastructure (WP3), and overall co-design (WP2) are taken into account to design and implement an AAI solution that will integrate well with the LEXIS platform.

This document has focused on AAI technology evaluation for LEXIS, while later deliverables will describe the final AAI solution in more detail. In the work presented here, three most promising AAI solutions have been selected for a detailed comparison. Sections of the document have been devoted to describing the set of criteria for evaluation and comparison with details on our methodology. One most promising solution (Keycloak) of three promising solutions has finally been selected for setting up a Proof-of-Concept (PoC). The PoC has been used to benchmark the solution. Keycloak has been found to cover all the requirements and specifications gathered. While Keycloak is extensible to include other technologies as well as other features, the alternative two solutions can be kept in mind as an emergency backup. As part of the PoC, some promising initial results have been achieved testing the Keycloak integration with other layers of the LEXIS platform (e.g., YORC orchestration, Portal).

REFERENCES

[1] LEXIS Deliverable, *D2.1 Pilots needs / Infrastructure Evaluation Report.*

[2] National Institute of Standards and Technology, “Minimum Security Requirements for Federal Information and Information Systems,” March 2006. [Online]. Available: https://csrc.nist.gov/publications/detail/fips/200/final. [Accessed September 2019].

[3] J. Katz and et al., Handbook of applied cryptography, CRC press, 1996.

[4] P. A. Grassi, M. E. Garcia and . J. L. Fenton, “Digital Identity Guidelines,” June 2017. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-3.pdf. [Accessed September 2019].

[5] A. Anderson and H. Lockhart, SAML 2.0 profile of XACML, OASIS, 2004.

[6] N. Sakimura et al., “OpenID Connect Core 1.0 incorporating errata set 1,” 2014. [Online]. Available: https://oauth.net/2/. [Accessed September 2019].

[7] “Session Fixation Attack against OAuth 1.0 Request Token approval flow,” April 2009. [Online]. Available: https://oauth.net/advisories/2009-1/. [Accessed September 2019].

[8] E. E. Hammer-Lahav, The OAuth 1.0 Protocol. RFC-5849, April, 2010.

[9] E. D. Hardt, The OAuth 2.0 Authorization Framework. RFC-6749, 2012.

[10] E. J. Sermersheim, Lightweight Directory Access Protocol (LDAP): The Protocol. RFC4511, 2006.

[11] C. Neuman, T. Yu, S. Hartman and K. Raeburn, The Kerberos Network Authentication Service (V5). RFC-4120, 2005.

[12] A. Singhal, T. Winograd and K. Scarfone, “Guide to Secure Web Services,” August 2007. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-95.pdf. [Accessed September 2019].

[13] L. Deliverable, *D2.2 Key parts LEXIS Technology Deployed on Existing Infrastructure and Key Technologies Specification.*

[14] V. Svaton, J. Martinovic, J. Krenek, T. Esch and P. Tomancak, “HPC-as-a-Service via HEAppE Platform,” in *onference on Complex, Intelligent, and Software Intensive Systems*, 2019.

[15] “Introduction to Keystone Federation,” August 2019. [Online]. Available: https://docs.openstack.org/keystone/latest/admin/federation/introduction.html. [Accessed September 2019].

[16] “Fernet specification,” 2014. [Online]. Available: https://github.com/fernet/spec/. [Accessed September 2019].

[17] “JSON Web Signature,” May 2015. [Online]. Available: https://tools.ietf.org/rfc/rfc7515.txt. [Accessed September 2019].

[18] “Spring Security framework,” [Online]. Available: https://spring.io/projects/spring-security. [Accessed September 2019].

[19] N. Jones, “Federation and identity brokering using Keycloak,” November 2018. [Online]. Available: https://www.stackhpc.com/federation-and-identity-brokering-using-keycloak.html. [Accessed September 2019].

[20] DMTF, “Cloud Auditing Data Federation standard,” 2015. [Online]. Available: https://www.dmtf.org/standards/cadf. [Accessed September 2019].

[21] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley and R. Housley, “Internet X.509 Public Key Infrastructure Certificate - RFC,” May 2008. [Online]. Available: https://tools.ietf.org/html/rfc5280. [Accessed September 2019].

[22] “Open Ondemand,” April 2019. [Online]. Available: https://gitlab.chpc.utah.edu/HPC/ondemand-info. [Accessed September 2019].

A. APPENDIX – LIST OF IAM SOLUTIONS

This section of the document reports a list of all the IAM solutions that have been evaluated.

KEYCLOAK:

WEBSITE	**https://www.keycloak.org**
LICENSE	Apache License Version 2.0 https://github.com/keycloak/keycloak/blob/master/License.htm
REQUIREMENTS	RDMS (Relational Database like SQL, Oracle, PostreSQL)
CLUSTERING/SCALABILITY	Standalone Clustered Mode, Domain Clustered Mode, Cross-Data-Centre Replication Mode
DISTRIBUTED/MULTISITE	Cross-Data-Centre Replication Mode
DISASTER RECOVERY/BACKUP	https://www.keycloak.org/docs/4.8/server_admin/#_export_import
AUTHENTICATION PROTOCOLS	OpenID Connect, SAML 2.0
AUTHENTICATION INTEGRATION	Local user with password policy, OTP policy, kerberos, X509 certificate + LDAP + Kerberos
AUTHORIZATION SERVICES	Authorization policies with: Attribute-based access control (ABAC), Role-based access control (RBAC), User-based access control (UBAC), Context-based access control (CBAC), Rule-based access control, Time-based access control + Support for custom access control mechanisms (ACMs) through a Policy Provider Service Provider Interface (SPI)
FUNCTIONALITIES	Caching, Policy management, Identity Broker
AUDITING	https://www.keycloak.org/docs/latest/server_admin/index.html#auditing-and-events
STORAGE	RDBMS (MySQL, PostgreSQL, Oracle)
ADMIN API	https://www.keycloak.org/docs/latest/server_admin/index.html#admin-console
USER API	https://www.keycloak.org/docs/latest/server_admin/index.html#user-management
SELF-REGISTRATION/USER VALIDATION	https://www.keycloak.org/docs/latest/server_admin/index.html#_user-registration
COMMENTS	https://www.stackhpc.com/federation-and-identity-brokering-using-keycloak.html

UNITY:

WEBSITE	**https://www.unity-idm.eu/**
LICENSE	https://github.com/unity-idm/unity/blob/dev/LICENCE.txt
REQUIREMENTS	Distributed RDMS (Relational Database like SQL, MySQL, H2) or Hazelcast storage (NoSQL)

CLUSTERING/SCALABILITY	Possible using Hazelcast: overlay over RDBMS, where all operations are done in memory
DISTRIBUTED/MULTISITE	https://www.unity-idm.eu/documentation/unity-2.8.2/manual.html#_notes_on_redundant_installations
DISASTER RECOVERY/BACKUP	https://www.unity-idm.eu/documentation/unity-2.8.2/manual.html#configuration
AUTHENTICATION PROTOCOLS	OAuth 2 Authorization Server and OpenID Connect endpoints; LDAP, OpenID, SAML
AUTHENTICATION INTEGRATION	(2014) Local user with password, X509 certificate; SAML endpoints (Web and SOAP) available; External LDAP and SAML
AUTHORIZATION SERVICES	https://www.unity-idm.eu/documentation/unity-1.4.0/manual.html#_authorization
FUNCTIONALITIES	https://www.unity-idm.eu/documentation/unity-1.4.0/manual.html#_core_engine_features
AUDITING	Not yet available
STORAGE	RDBMS (Embedded H2 Database)
ADMIN API	https://www.unity-idm.eu/documentation/unity-1.4.0/manual.html#contents-management
USER API	https://www.unity-idm.eu/documentation/unity-1.4.0/manual.html#_user_home_endpoint
SELF-REGISTRATION/USER VALIDATION	https://www.unity-idm.eu/documentation/unity-1.4.0/manual.html#_registration_forms
COMMENTS	

APACHE SYNCOPE:

WEBSITE	**https://syncope.apache.org/**
LICENSE	https://www.apache.org/licenses/
REQUIREMENTS	HW requirements, Java JDK and JRE, Java EE Container, RDBMS (PostgreSQL, MySQL, MS SQL, Oracle)
CLUSTERING/SCALABILITY	http://syncope.apache.org/docs/2.1/reference-guide.html#high-availability
DISTRIBUTED/MULTISITE	http://syncope.apache.org/docs/2.1/reference-guide.html#system-administration
DISASTER RECOVERY/BACKUP	http://syncope.apache.org/docs/2.1/reference-guide.html#database-connection-pool
AUTHENTICATION PROTOCOLS	SAML 2.0, OpenID Connect
AUTHENTICATION INTEGRATION	Based on Spring Security
AUTHORIZATION SERVICES	Based on Spring Security
FUNCTIONALITIES	Users, Groups and Any Objects; Roles, Policies, Resources; Realms

AUDITING	https://syncope.apache.org/docs/2.1/reference-guide.html#audit
STORAGE	Flat files (XML, CSV, etc.), LDAP, RDBMS (MySQL, Oracle, etc.), platform-specific (Microsoft Active Directory, FreeIPA, PowerShell, etc.), Web services (REST, SOAP, etc.), Cloud providers and more
ADMIN API	https://syncope.apache.org/docs/2.1/reference-guide.html#admin-console
USER API	https://syncope.apache.org/docs/2.1/reference-guide.html#enduser-application
SELF-REGISTRATION/USER VALIDATION	Self-registration, self-service and password reset through End-User UI
COMMENTS	

OPENIAM:

WEBSITE	**https://www.openiam.com/**
LICENSE	https://www.openhub.net/p/openiam-idm-ce
REQUIREMENTS	http://docs40.openiam.com/#installation/about.htm#1.2_System_requirements
CLUSTERING/SCALABILITY	RDBMS cluster
DISTRIBUTED/MULTISITE	http://docs40.openiam.com/#installation/about.htm#1.1_Choosing_the_installation_type%3FTocPath%3DInstallation%2520Guide%7C1.%2520About%2520installing%2520OpenIAM%7C_____1
DISASTER RECOVERY/BACKUP	Some Docker installation + scripting seems the best way
AUTHENTICATION PROTOCOLS	SAML, Oauth 2.0, OpenID Connect
AUTHENTICATION INTEGRATION	(Enterprise version) password, Directory (AD/LDAP), SSO protocols (SAML, OAuth2, OpenID Connect), OTP policy (SMS/email/mobile with push notification), social authentication, Kerberos, Certificate-based authentication, Custom login module, Adaptive (Contextual) Authentication
AUTHORIZATION SERVICES	RBAC Access control policies, Attribute Based Access Control - XACML (add-on)
FUNCTIONALITIES	User lifecycle management, Identity lifecycle management, Role lifecycle management (more on OpenIAM Features)
AUDITING	NoSQL audit repository (optional), http://docs40.openiam.com/#administration/sysadm/configuration.htm#24.8_Configuring_system_audit_log%3FTocPath%3DAdministration%2520Guide%7CPart%2520II%253A%2520System%2520administration%7C24.%2520System%2520configuration%7C_____8
STORAGE	RDBMS
ADMIN API	http://docs40.openiam.com/#administration/index.htm%3FTocPath%3DAdministration%2520Guide%7C_____0
USER API	http://docs40.openiam.com/#self-service/index.htm%3FTocPath%3DSelf-Service%2520User%2520Guide%7C_____0

SELF-REGISTRATION/USER VALIDATION	http://docs40.openiam.com/#self-service/overview.htm%3FTocPath%3DSelf-Service%2520User%2520Guide%7C1.%2520About%2520Self-Service%7C1.3%2520Self-registration%7C_____0#1.3_Self-registration
COMMENTS	https://www.openhub.net/p/openiam-idm-ce, http://docs40.openiam.com/whats-new/wn-architecture.htm?TocPath=What%27s%20New%7C2.%20Architecture%7C_____6

WSO2:

WEBSITE	**https://wso2.com/identity-and-access-management/**
LICENSE	https://wso2.com/licenses
REQUIREMENTS	https://docs.wso2.com/display/IS580/Installation+Prerequisites
CLUSTERING/SCALABILITY	https://docs.wso2.com/display/IS580/Deployment+Patterns
DISTRIBUTED/MULTISITE	https://docs.wso2.com/display/IS580/Setting+Up+Separate+Databases+for+Clustering, https://docs.wso2.com/display/IS580/Deployment+Patterns
DISASTER RECOVERY/BACKUP	https://docs.wso2.com/display/IS580/Deployment+Guidelines+in+Production
AUTHENTICATION PROTOCOLS	SAML2, OpenID Connect and WS-Federation Passive
AUTHENTICATION INTEGRATION	X.509 certificate, IWA with Kerberos, Fast IDentity Online (FIDO), Time-based One-time Password (TOTP); LDAP (ApacheDS, an external LDAP, Microsoft Active Directory, or any JDBC database), MFA, Adaptive Authentication
AUTHORIZATION SERVICES	Role-based access control (RBAC), eXtensible Access Control Markup Language (XACML) 2.0/3.0,
FUNCTIONALITIES	User, group management
AUDITING	https://docs.wso2.com/display/IS580/Monitoring+the+Identity+Server
STORAGE	RDMS
ADMIN API	https://docs.wso2.com/display/IS580/Calling+Admin+Services
USER API	https://docs.wso2.com/display/IS580/Using+APIs
SELF-REGISTRATION/USER VALIDATION	https://wso2.com/whitepapers/customer-identity-and-access-management-a-wso2-reference-architecture/#2111
COMMENTS	

GLUU:

WEBSITE	**https://gluu.org/docs**
LICENSE	https://gluu.org/docs/ce/4.0/#license
REQUIREMENTS	Nginx for Load Balancing/Proxying, Redis server (NB: Gluu server embeds a local LDAP server with multi-master replication)
CLUSTERING/SCALABILITY	GLUU Cluster Manager, https://gluu.org/docs/ce/4.0/installation-guide/cluster/

DISTRIBUTED/MULTISITE	Using Clustering installation
DISASTER RECOVERY/BACKUP	https://gluu.org/docs/ce/4.0/operation/backup/
AUTHENTICATION PROTOCOLS	Shibboleth SAML IDP, OAuth 2.0 federation standards like OpenID Connect & UMA
AUTHENTICATION INTEGRATION	SCIM, U2F, FIDO 2.0/WebAuthn, LDAP
AUTHORIZATION SERVICES	Authorization policies with: User-based access control (UBAC), Attribute-based access control (ABAC), with Group-based and Role-based policies as a natural subset
FUNCTIONALITIES	SSO, 2FA, Customer Identity and Access Management
AUDITING	https://wiki.shibboleth.net/confluence/display/IDP30/AuditLoggingConfiguration
STORAGE	Redis + Embedded LDAP
ADMIN API	https://gluu.org/docs/ce/4.0/admin-guide/oxtrust-ui/#accessing-the-ui
USER API	https://gluu.org/docs/ce/4.0/api-guide/api/
SELF-REGISTRATION/USER VALIDATION	https://gluu.org/docs/ce/user-management/user-registration/
COMMENTS	https://www.gluu.org/shibboleth-idp/

EVOLEUM MIDPOINT:

WEBSITE	**https://evolveum.com/midpoint/**
LICENSE	https://wiki.evolveum.com/display/midPoint/Licensing
REQUIREMENTS	External RDBMS (SQL), Load Balancer and, High Availability proxy Tomcat (according to deployment)
CLUSTERING/SCALABILITY	https://wiki.evolveum.com/display/midPoint/High+Availability+and+Load+Balancing
DISTRIBUTED/MULTISITE	https://wiki.evolveum.com/display/midPoint/System+Requirements#SystemRequirements-HighAvailability
DISASTER RECOVERY/BACKUP	No Automated procedure, must be handle according to each component
AUTHENTICATION PROTOCOLS	Based on Spring Security: Kerberos, Oauth 1(a) and 2.0, SAML 2.0
AUTHENTICATION INTEGRATION	Based on Spring Security
AUTHORIZATION SERVICES	Advanced Hybrid RBAC
FUNCTIONALITIES	Live synchro for managing Identity life-cycle, https://wiki.evolveum.com/pages/viewpage.action?pageId=14286949
AUDITING	SQL repository + auditing to PostgreSQL, audit also to logs
STORAGE	RDBMS (SQL)

ADMIN API	https://wiki.evolveum.com/display/midPoint/User+Interface
USER API	https://wiki.evolveum.com/display/midPoint/REST+API
SELF-REGISTRATION/USER VALIDATION	https://wiki.evolveum.com/display/midPoint/Self+Registration+Configuration
COMMENTS	https://idm365.com/idm365-the-rbac-abac-hybrid-solution/

SOFFID:

WEBSITE	**https://www.soffid.com/**
LICENSE	http://www.soffid.com/doc/console/iam-core/license.html
REQUIREMENTS	RDBMS
CLUSTERING/SCALABILITY	http://confluence.soffid.org/display/SOF/Creating+a+multi-master+MariaDB+cluster
DISTRIBUTED/MULTISITE	Through DB clustering + Load balancer / HA Proxies
DISASTER RECOVERY/BACKUP	http://www.soffid.com/doc/console/iam-core/license.html, http://confluence.soffid.org/display/SOF/System+backup
AUTHENTICATION PROTOCOLS	SAML and OpenID bridge
AUTHENTICATION INTEGRATION	LDAP directories, MS Active Directory, RDBMS and most common Operating Systems; Two factor authentication (2FA)
AUTHORIZATION SERVICES	RBAC; also, XACML optional module available to define attribute-based control policy (ABAC)
FUNCTIONALITIES	Fine tuning permissions based on organization role, organization unit or granted roles
AUDITING	http://www.soffid.com/our-solutions/#identity-governance-audit
STORAGE	it supports certain number of RDBMS including MariaDB, MySQL, Oracle and Microsoft SQL Server
ADMIN API	Admin scripting, http://confluence.soffid.org/pages/viewpage.action?pageId=173703171
USER API	A single, simple and intuitive web interface for the end user to manage their own profile, request passwords, access directly to their applications and manage their own business processes.
SELF-REGISTRATION/USER VALIDATION	http://confluence.soffid.org/display/SOF/Web+services+reference
COMMENTS	

SHIBBOLETH:

WEBSITE	**https://www.shibboleth.net/**
LICENSE	http://apache.org/licenses/LICENSE-2.0.html
REQUIREMENTS	https://wiki.shibboleth.net/confluence/display/IDP30/SystemRequirements

CLUSTERING/SCALABILITY	Use of additional software such as memecache or JPA/Hibernate, https://wiki.shibboleth.net/confluence/display/IDP30/Clustering
DISTRIBUTED/MULTISITE	Uses Clustering techniques with heavy configuration
DISASTER RECOVERY/BACKUP	Manual process that need to be automated
AUTHENTICATION PROTOCOLS	SAML 1.1 and 2.0, CAS 2
AUTHENTICATION INTEGRATION	LDAP, Kerberos, JAAS, X.509, SPNEGO, Duo Security, and container-based authentication systems
AUTHORIZATION SERVICES	Authorization policies with Access Control (type not specified)
FUNCTIONALITIES	https://wiki.shibboleth.net/confluence/display/IDP30/InterestingFeatures
AUDITING	https://wiki.shibboleth.net/confluence/display/IDP30/AuditLoggingConfiguration
STORAGE	Internal IDP storage services + RDMS such as Oracle, PostgreSQL, MySQL, H2 (https://wiki.shibboleth.net/confluence/display/IDP30/StorageConfiguration)
ADMIN API	https://wiki.shibboleth.net/confluence/display/IDP30/AdministrativeConfiguration
USER API	Old API no more supported (https://wiki.shibboleth.net/confluence/display/SHIB2/IdPAPI), and work in progress (https://wiki.shibboleth.net/confluence/display/DEV/API+Key+Access+Control)
SELF-REGISTRATION/USER VALIDATION	No user registration system
COMMENTS	https://www.gluu.org/shibboleth-idp/

OPENSTACK KEYSTONE:

WEBSITE	**https://docs.openstack.org/keystone/latest/**, **https://docs.openstack.org/keystone/stein/**
LICENSE	https://github.com/openstack/keystone/blob/master/LICENSE
REQUIREMENTS	Python, Default SQL Backend
CLUSTERING/SCALABILITY	https://severalnines.com/blog/clustering-mysql-backend-openstack, https://docs.openstack.org/ha-guide/
DISTRIBUTED/MULTISITE	https://wiki.openstack.org/wiki/Keystone_edge_architectures#Architecture_options
DISASTER RECOVERY/BACKUP	**https://docs.openstack.org/keystone/rocky/admin/identity-upgrading.html**
AUTHENTICATION PROTOCOLS	SAML2.0, OpenID Connect, OAuth 1.0a
AUTHENTICATION INTEGRATION	Multi-Factor Authentication (MFA), Time-based One-time Password (TOTP), HTTPD authentication for mod_mellon and mod_shibboleth, X.509 Tokenless authorization, OAuth1 1.0a; integration with existing LDAP directories (see Docs: Authentication Mechanisms and Federated Identity)

AUTHORIZATION SERVICES	Role-based access control (RBAC), https://wiki.openstack.org/wiki/KeystoneUseCases#User_Story:_Basic_RBAC_.28role_based_access_controls.29
FUNCTIONALITIES	https://wiki.openstack.org/wiki/KeystoneUseCases
AUDITING	https://docs.openstack.org/keystone/pike/advanced-topics/event_notifications.html#auditing-with-cadf
STORAGE	RDBMS, Galera for instance, LDAP
ADMIN API	There are two supported clients, *python-keystoneclient* project provides python bindings and *python-openstackclient* provides a command line interface.
USER API	https://docs.openstack.org/keystone/stein/contributor/http-api.html
SELF-REGISTRATION/USER VALIDATION	https://wiki.openstack.org/wiki/OpenStack:Horizon_User_Registration_Blueprint
COMMENTS	https://www.redhat.com/en/topics/openstack

EDUGAIN:

WEBSITE	**https://edugain.org/**
LICENSE	From custom repository: https://github.com/biancini/edugain-connectivity-check/blob/master/LICENSE
REQUIREMENTS	RDBMS (MariaDB)
CLUSTERING/SCALABILITY	https://wiki.geant.org/display/eduGAIN/Federation+Architectures
DISTRIBUTED/MULTISITE	rely on the Architecture
DISASTER RECOVERY/BACKUP	Manual backup of your own Identity Provider
AUTHENTICATION PROTOCOLS	SAML Shibboleth IdP, midPoint, SAML 2.0
AUTHENTICATION INTEGRATION	https://wiki.geant.org/display/aai/AAI+Integration+Services
AUTHORIZATION SERVICES	From basic documentation page (https://technical.edugain.org/documents)
FUNCTIONALITIES	From a presentation done in 2018: Group Management (Grouper+UI), Collaborative Organizations Management (COmanage+UI), SAML-based Identity Provider and related Service Provider (Shibboleth+UI), InCommon Federation Manager (FM+UI), Entity Registry Storage + Provisioning and Deprovisioning (midPoint), AMQP Compliant messaging middleware (RabbitMQ), Relational Database Solution to support deployment (MariaDB); Solution Packaging (Docker Containers)
AUDITING	https://technical.edugain.org/monitoring
STORAGE	E.g., Shibboleth, Jetty, LDAP, MySQL
ADMIN API	https://technical.edugain.org/api
USER API	https://technical.edugain.org/api

SELF-REGISTRATION/USER VALIDATION	https://wiki.geant.org/display/eduGAIN/eduGAIN+Access+Check
COMMENTS	https://wiki.geant.org/display/eduGAIN/Tools+and+Services, https://wiki.geant.org/display/eduGAIN/eduGAIN+GDPR+Impact+Assessment

APEREO CAS:

WEBSITE	**https://www.apereo.org/projects/cas**
LICENSE	https://apereo.github.io/cas/4.2.x/protocol/CAS-Protocol-Specification.html#appendix-e-cas-license
REQUIREMENTS	https://apereo.github.io/cas/4.2.x/planning/Installation-Requirements.html
CLUSTERING/SCALABILITY	https://apereo.github.io/cas/4.2.x/planning/High-Availability-Guide.html#high-availability-guide-haclustering, https://apereo.github.io/cas/6.0.x/high_availability/High-Availability-Guide.html
DISTRIBUTED/MULTISITE	https://apereo.github.io/cas/4.2.x/planning/High-Availability-Guide.html#multiple-cas-server-nodes
DISASTER RECOVERY/BACKUP	Default configuration include 1 backup node for Hazelcast (https://apereo.github.io/cas/6.0.x/ticketing/Hazelcast-Ticket-Registry.html#hazelcast-ticket-registry), https://apereo.github.io/cas/6.0.x/services/Configuring-Service-Replication.html
AUTHENTICATION PROTOCOLS	CAS Protocol version 1m2 and 3 (exclusive for CAS), SAML 1.1, OAuth 1.0 and 2.0, OpenID Connect, SCIM and WS-Fed; possible to integrate with Shibboleth IdP (https://apereo.github.io/cas/6.0.x/protocol/Protocol-Overview.html)
AUTHENTICATION INTEGRATION	Database, JAAS, LDAP, OAuth 1.0/2.0, OpenID, RADIUS, SPNEGO (Windows), Trusted (REMOTE_USER), X.509 client SSL certificate, Remote Address, YubiKey, Apache Shiro, pac4j; Multifactor Authentication (MFA)
AUTHORIZATION SERVICES	CAS ABAC, Custom ABAC, LDAP support
FUNCTIONALITIES	Java (Spring Webflow/MVC servlet) server component; Pluggable authentication support (LDAP, database, X.509, 2-factor); Support for multiple protocols (CAS, SAML, OAuth, OpenID); Cross-platform client support (Java, .Net, PHP, Perl, Apache, etc.); Integrates with uPortal, Liferay, BlueSocket, Moodle, and Google Apps to name a few
AUDITING	https://apereo.github.io/cas/4.2.x/installation/Audits.html, https://apereo.github.io/cas/6.0.x/logging/Logging.html
STORAGE	LDAP system
ADMIN API	https://apereo.github.io/cas/6.0.x/configuration/Configuration-Properties.html#rest-api
USER API	https://apereo.github.io/cas/6.0.x/protocol/REST-Protocol.html
SELF-REGISTRATION/USER VALIDATION	We did not found clear documentation on it, but we suspect that there is a self-registration feature
COMMENTS	

FREEIPA:

WEBSITE	**https://www.freeipa.org/page/Main_Page**
LICENSE	https://www.freeipa.org/page/License
REQUIREMENTS	https://www.freeipa.org/page/FreeIPAv2:PRD
CLUSTERING/SCALABILITY	Assuming that clients do intelligent caching, we do not need that many FreeIPA servers to handle the load from all clients.
DISTRIBUTED/MULTISITE	We did not find any proper documentation for properly setting up such installation
DISASTER RECOVERY/BACKUP	https://www.freeipa.org/page/Backup_and_Restore, https://www.freeipa.org/page/FreeIPAv2:V2/RollingUpgrade
AUTHENTICATION PROTOCOLS	SAML; LDAP
AUTHENTICATION INTEGRATION	System Security Services Daemon (SSSD); Kerberos (with mod_auth_gssapi or mod_auth_kerb); Pure Application Level, Kerberos SSO (ticket), SAML-based, Certificate-based; Login form-based
AUTHORIZATION SERVICES	Host and service based access control (HBAC), with access control check to the Kerberos authentication method on the Apache level in order to prevent access to unauthorized users that are able to get the Kerberos ticket (Note: default rule *allow_all* grants access from anywhere to anywhere to any user and service. It needs to be disabled)
FUNCTIONALITIES	The FreeIPA Directory Service is built on the 389 DS LDAP server, acting as data backend for all identity, authentication (Kerberos) and authorization services and other policies
AUDITING	https://www.freeipa.org/page/Session_Recording#Audit_recording_details
STORAGE	Using Directory servers such as LDAP or Kerberos server
ADMIN API	Web UI or CLI
USER API	https://www.freeipa.org/page/API_Examples
SELF-REGISTRATION/USER VALIDATION	https://www.freeipa.org/page/Self-Service_Password_Reset
COMMENTS	

JOSSO:

WEBSITE	**http://www.josso.org/**
LICENSE	https://github.com/atricore/josso2/blob/2.4.3/LICENSE
REQUIREMENTS	JRE 8 or newer
CLUSTERING/SCALABILITY	http://docs.atricore.com/josso2/2.4.0/josso-reference-guide/html/en-US/JOSSO_Reference.html#About_High_Availability
DISTRIBUTED/MULTISITE	http://docs.atricore.com/josso2/2.4.0/josso-reference-guide/html/en-US/JOSSO_Reference.html#High_Availability_and_Scalability

DISASTER RECOVERY/BACKUP	**http://docs.atricore.com/josso2/2.4.0/josso-reference-guide/html/en-US/JOSSO_Reference.html#Identity_Appliance_Lifecycle_Management**
AUTHENTICATION PROTOCOLS	SAML 2.0, OAuth 2.0, OpenID 2.0, SSL for certificate-based, ID confirmation with OAuth2 Access Token Issuance (?)
AUTHENTICATION INTEGRATION	(Following protocols order) LDAP for Directory-based auth., Kerberos for Integrated Windows auth., WiKID Two Factor Authentication (2FA) with OTP, SSO Domino Auth. with Lightweight Third-Party Authentication (LTPA from IBM), Certificate-based auth. via SSL, JBoss Enterprise Portal Platform (or JBoss EPP)
AUTHORIZATION SERVICES	Role-Based Access Control (RBAC) with both accounts and groups provisioning; Identity Appliance Life Cycle Management, Account and Entitlement Management
FUNCTIONALITIES	No extra functionality mentioned apart from the IAM ones.
AUDITING	http://docs.atricore.com/josso2/2.4/tutorials/josso-auditing-tutorial/html/en-US/JOSSO_Tutorial_Auditing.html
STORAGE	Vault system, RDBMS, LDAP (several choice are available)
ADMIN API	http://docs.atricore.com/josso2/2.4/tutorials/josso-jaxrs-tutorial/html/en-US/JOSSO_Tutorial_JAXRS.html
USER API	http://docs.atricore.com/josso2/2.4/tutorials/josso-jaxrs-tutorial/html/en-US/JOSSO_Tutorial_JAXRS.html#_sample_client_code
SELF-REGISTRATION/USER VALIDATION	The system seems to support this feature whereas it is not clearly mentioned.
COMMENTS	Based on Atricore IAM Platform

OPENAM:

WEBSITE	**https://www.openidentityplatform.org/**
LICENSE	https://github.com/OpenIdentityPlatform/OpenAM/blob/master/LICENSE.md
REQUIREMENTS	fully qualified domain name (FQDN); Java JRE; Docker and Apache HTTP Server
CLUSTERING/SCALABILITY	To enable high availability for large-scale and mission-critical deployments, OpenAM provides both system failover and session failover. These two key features help to ensure that no single point of failure exists in the deployment, and that the OpenAM service is always available to end-users. Redundant OpenAM servers, policy agents, and load balancers prevent a single point of failure. Session failover ensures the user's session continues uninterrupted, and no user data is lost. (from Wikipedia) (http://www.janua.fr/clustering-independant-openam-servers/, https://backstage.forgerock.com/knowledge/kb/article/a61399600)
DISTRIBUTED/MULTISITE	Using Clustering
DISASTER RECOVERY/BACKUP	No real documentation available, the process must be defined according to use case
AUTHENTICATION PROTOCOLS	SAML, Oauth 2.0, OpenID Connect 1; possible setup of WebAuthn standard by W3C and FIDO
AUTHENTICATION INTEGRATION	https://github.com/OpenIdentityPlatform/OpenAM/wiki/Authentication-modules

AUTHORIZATION SERVICES	Authorization policy from basic, simple, coarse-grained rules to highly advanced, fine-grained entitlements based on XACML (eXtensible Access Control Mark-Up Language). Authorization policies are abstracted from the application, allowing developers to quickly add or change policy as needed without modification to the underlying application. (from Wikipedia)
FUNCTIONALITIES	No extra functionality mentioned apart from the IAM ones.
AUDITING	https://backstage.forgerock.com/docs/openam/13.5/reference/#chap-audit-log-messages
STORAGE	noSQL embedded database (https://backstage.forgerock.com/docs/idm/6.5/install-guide/#chap-install)
ADMIN API	OpenAM provides client application programming interfaces with Java and C APIs and a RESTful API that can return JSON or XML over HTTP, allowing users to access authentication, authorization, and identity services from web applications using REST clients in their language of choice. OAuth2 also provides a REST Interface for the modern, lightweight federation and authorization protocol.
USER API	https://backstage.forgerock.com/docs/idm/6.5/self-service-reference/
SELF-REGISTRATION/USER VALIDATION	https://backstage.forgerock.com/docs/openam/13.5/reference/#legacy-user-self-service
COMMENTS	Announced by Sun Microsystems in 2005, formerly supported by ForgeRock, from 2010 to 2016 (then forked in ForgeRock Access Management, under paid commercial license). Free and open-source fork now supported by Open Identity Platform Community (see Wikipedia)

UNIVENTION CORPORATE SERVER (UCS):

WEBSITE	**https://github.com/univention/univention-corporate-server**
LICENSE	https://github.com/univention/univention-corporate-server/blob/4.4-1/LICENSE
REQUIREMENTS	1GB memory and 8GB hard drive space
CLUSTERING/SCALABILITY	No real clustering capabilities out of the box (https://docs.software-univention.de/performance-guide-4.4.html#slapd)
DISTRIBUTED/MULTISITE	https://docs.software-univention.de/manual-4.4.html#domain:fault-tolerant
DISASTER RECOVERY/BACKUP	https://docs.software-univention.de/manual-4.4.html#domain-ldap:Domain_controller_backup
AUTHENTICATION PROTOCOLS	SAML, Kerberos, SSL certificate, LDAP, OpenID Connect; RADIUS; UMCP 2.0 (based on JSON)
AUTHENTICATION INTEGRATION	LDAP, Kerberos, Two Factor Authentication (2FA) (for example a TAN generated randomly each time), Certificate-based auth. via SSL
AUTHORIZATION SERVICES	Access to the information contained in the LDAP directory is controlled by Access Control Lists (ACLs) on the server side; RADIUS
FUNCTIONALITIES	Group and computer management, IP and network management, file share management

AUDITING	Possible, on a share-by-share basis even. You'll have to enable the audit VFS module for the share (from UCS Forum)
STORAGE	LDAP server
ADMIN API	https://docs.software-univention.de/ucs-python-api/
USER API	Web interface of UCS system. Microsoft Windows clients and Max OS X systems are integrated via a Samba-based, AD-compatible Windows domain; most Linux distros (Ubuntu, Debian, SUSE or RedHat) can also be integrated into the domain. Web browser with Dojo/UMC JS API, communication to UMC HTTP server via AJAX and JSON
SELF-REGISTRATION/USER VALIDATION	https://www.univention.com/blog-en/2019/04/ucs-4-4-self-services-new-features/
COMMENTS	https://www.univention.com/blog-en/2015/11/release-of-ucs-4-1-with-docker-single-sign-on-mechanism-and-two-factor-authentication/

AEROBASE IAM:

WEBSITE	**https://aerobase.io/iam**
LICENSE	https://github.com/aerobase/unifiedpush-server/blob/master/LICENSE.txt
REQUIREMENTS	Java 8 JDK, HW requirements (512M RAM + 1GB storage), a shared external database (PostgreSQL, MySQL, Oracle, etc.) for Cluster mode
CLUSTERING/SCALABILITY	https://aerobase.io/docs/installation/index.html#_standalone-ha-mode, https://aerobase.io/docs/installation/index.html#_clustering
DISTRIBUTED/MULTISITE	No real multi-site recommendation, so a clustered or HA mode should be envisioned
DISASTER RECOVERY/BACKUP	No out of the box backup
AUTHENTICATION PROTOCOLS	OpenID Connect, OAuth 2.0, SAML 2.0; LDAP, Kerberos
AUTHENTICATION INTEGRATION	LDAP, Identity brokering, Social login (Google, GitHub, Facebook, etc.), Kerberos bridging, Two-factor Authentication (2FA)
AUTHORIZATION SERVICES	Reamls, Groups, Users with user role mapping
FUNCTIONALITIES	Central management of users, roles, role mappings, clients and configuration through Admin console
AUDITING	No out of the bow mechanisms
STORAGE	RDBMS with embedded PostgreSQL or MySQL, PostgreSQL, MS SQL or Oracle (https://aerobase.io/docs/installation/index.html#_database)
ADMIN API	https://aerobase.io/docs/server_development/index.html#admin-rest-api
USER API	web apps and RESTful web services (https://aerobase.io/docs/server_development/index.html#identity-brokering-apis)
SELF-REGISTRATION/USER VALIDATION	Not available as far as we could see

COMMENTS	https://aerobase.io/docs/server_admin/index.html#features