Integrating CLARIN SSO Authentication into INCEpTION: A Technical Report

author: Michele Mallia[1]

co-authors: Dimitris Galanis[2], Dimitris Gkoumas[3], Stelios Piperidis[4]

revisor : Richard Eckart De Castilho[5]

mail: michele.mallia@ilc.cnr.it[1] - michele.mallia@cnr.it[1] - galanisd@athenarc.gr[2] - dgkoumas@athenarc.gr[3] - spip@athenarc.gr[4] - richard.eckart_de_castilho@tu-darmstadt.de[5]

Pisa : 05/2025

Summary

Introduction

This technical report provides comprehensive documentation for the integration of a Single Sign-On (SSO) authentication mechanism—specifically designed to interoperate with the CLARIN authentication federation—within the INCEpTION platform. INCEpTION, a semantic annotation and knowledge management environment, is developed by the Technical University of Darmstadt and is hosted at the CLARIN-EL research infrastructure. This integration represents a significant step forward in improving the usability, accessibility, and interoperability of the platform within the context of distributed digital research infrastructures.

The overarching objective of this document is twofold. First, it aims to deliver an in-depth explanation of the technical implementation of SSO within the INCEpTION application. This includes configuration details, integration strategies, and protocol-level considerations required to ensure secure and seamless user authentication across federated identity providers. Second, it seeks to contextualize this implementation within the broader scope of ongoing research infrastructure projects, articulating the motivations that necessitated this enhancement and the strategic value it contributes to the scientific and scholarly community engaged with CLARIN services.

Beyond the technical layer, this documentation also serves to inform stakeholders—developers, administrators, and project coordinators—about the broader project rationale and the expected long-term benefits. These include improved user experience, centralized access management, enhanced security compliance, and the promotion of FAIR data principles through better alignment with established digital infrastructure standards.

The document is structured into five main sections for clarity and ease of navigation. Section 1 focuses on the institutional actors involved, the overall project context, and the motivations that led to the implementation of the SSO integration. Section 2 focuses on the technological environment, detailing the relevant tools, frameworks, and authentication standards that underpin the integration. Section 3 elaborates on the architectural design of the implemented solution, offering diagrams and component-level explanations. Section 4 delivers a step-by-step procedural guide to implementing the SSO mechanism within an INCEpTION deployment, along with explanations of key operational workflows and authentication logic. Section 5 is dedicated to validating the functioning of the federated authentication infrastructure. It provides a step-by-step overview of how to verify that the entire identity flow—from an external Identity Provider, through the SimpleSAMLphp proxy and Keycloak broker, to the target application (INCEpTION)—is correctly established. Lastly, Section 6 concludes the report with a forward-looking analysis of the broader implications of this work, emphasizing its reusability, scalability, and applicability in related research ecosystems such as ILC4CLARIN and H2IOSC.

1 – Actors, Projectual Contexts and motivations

In this section, we discuss the key institutional actors, collaborative frameworks, and strategic motivations that led to the implementation of a federated **Single Sign-On (SSO)** system within the **INCEpTION** platform. We begin by outlining the roles of the organizations and individuals involved, including their respective contributions to the project from both a technical and infrastructural perspective. We then explore the broader project contexts—such as **CLARIN:EL**, **ILC4CLARIN**, and **H2IOSC**—that provided the motivation and operational framework for the development. Finally, we highlight how the adoption of SSO aligns with long-term objectives for sustainability, interoperability, and reuse within national and European research infrastructures.

1.1 – Institutional Actors and Collaborative Framework

The implementation of the Single Sign-On (SSO) authentication system within the INCEpTION platform[1] emerged as the result of a structured and multi-faceted collaboration involving several key institutional actors operating at the intersection of research infrastructure, language technologies, and digital identity management. This initiative was deeply rooted in the collaborative ethos of the CLARIN community, where interoperability and accessibility are core principles.

A central role in this collaboration was played by members of the **CLARIN:EL**[2] infrastructure (De Castilho, Klie, Kumar, Boullosa, & Gurevych, 2018), which represents the Greek national hub for language resources and technologies and is an integral component of the European CLARIN ERIC infrastructure. The technical team from CLARIN:EL, including experts such as **Dimitris Gkoumas**, **Stelios Piperidis**, and **Dimitrios Galanis**, brought to the table extensive experience in implementing federated identity solutions, having successfully deployed SSO within their own national infrastructure portal. Their contribution served not only as a reference model but also as a source of technical consultation throughout the process.

In parallel, critical development support was provided by **Richard Eckart de Castilho** of the **UKP Lab at the Technical University of Darmstadt**, the institution responsible for the development and maintenance of the INCEpTION platform. As one of the original architects of the system, de Castilho's input was vital in identifying the appropriate integration points within the application's architecture and ensuring compatibility with SAML-based authentication protocols.

This transnational and interdisciplinary collaboration bridged software development, infrastructure configuration, and authentication federation policies, ultimately facilitating the seamless integration of federated identity management capabilities into a robust, open-source annotation environment. The synergy among these actors not only enabled technical progress but also embodied a shared commitment to sustainable and interoperable research tools for the linguistic and digital humanities communities (De Castilho, Klie, Kumar, Boullosa, & Gurevych, 2018).

[1] https://inception-project.github.io/
[2] https://www.clarin.gr/en

1.2 – Role of ILC4CLARIN and Technical Implementation

This technical development was conducted at the **Institute for Computational Linguistics "A. Zampolli" (CNR-ILC)**[3], located in Pisa, Italy. CNR-ILC is a prominent research institute within the Italian National Research Council (CNR) and has long played a central role in supporting digital language resource infrastructures at both the national and European levels. The institute also hosts **ILC4CLARIN**[4], the first certified **CLARIN B-Centre**[5] in Italy, which provides a suite of stable, persistent services and repositories for the storage, distribution, and long-term preservation of linguistic data and tools. As a key actor in the Italian CLARIN node (**CLARIN-IT**[6]), CNR-ILC ensures alignment with the broader CLARIN ERIC standards, including those related to access, authentication, and data sharing.

The implementation effort was led by the technical team at CNR-ILC and included the **customization and deployment of a Service Provider component using SimpleSAMLphp**[7], a widely adopted open-source SAML toolkit. This component was tailored to interoperate with the CLARIN Service Provider Federation and adapted to comply with the identity and attribute exchange requirements of the infrastructure. At the same time, **Keycloak**[8], an open-source identity and access management system, was configured to act as the **federated Identity Broker**[9], bridging authentication requests between the local service and the CLARIN Identity Federation.

To enable interaction with this authentication proxy, the application must be capable of communicating via the **SAML protocol**[10]. This requires appropriate configuration and, if necessary, specific adaptations to ensure compatibility with the identity broker—in this case, Keycloak. These adjustments allow the application to delegate user authentication to external Identity Providers, enabling users to access INCEpTION through their institutional credentials without the need for separate account creation or manual user provisioning.

All testing and preliminary deployments were conducted on a **virtualized infrastructure provided by GARR**[11], the national network provider for education and research in Italy. GARR's virtual environment ensured not only high availability and secure connectivity but also compliance with academic standards for testing federated authentication mechanisms. This environment proved essential for iterative testing, troubleshooting, and validation of the SSO implementation across different identity federation scenarios, prior to the deployment in a production context.

---

[3] https://www.ilc.cnr.it/
[4] https://ilc4clarin.ilc.cnr.it/
[5] https://www.clarin.eu/content/certified-b-centres
[6] https://www.clarin-it.it/en
[7] https://simplesamlphp.org/
[8] https://www.keycloak.org/
[9] The term "federated identity broker" refers to an **intermediary system that manages user authentication between multiple identity providers (IdPs) and one or more service providers (SPs).**
A federated identity broker is a "bridge" that allows a user to access a web service using their credentials from another system (e.g., Google, Facebook, or an institutional identity provider), without having to create a new account for each service.
[10] **SAML (Security Assertion Markup Language)** is an open standard for exchanging authentication and authorization data between parties—specifically, between an identity provider (IdP) and a service provider (SP). It enables Single Sign-On (SSO) across different domains by allowing users to authenticate once and gain access to multiple systems.
[11] https://www.garr.it/it/

1.3 – Motivation and Strategic Objectives

The primary motivation for enabling SSO in INCEpTION stems from the platform's growing importance as a tool for semantic annotation and knowledge management across a wide range of academic disciplines. Researchers in linguistics, philology, corpus studies and digital humanities increasingly rely on INCEpTION for the annotation of complex semantic phenomena, the construction of domain-specific corpora, and the development of machine-readable lexical resources.

The application is natively enabled to communicate via the SAML protocol. By integrating it with Keycloak, it is possible to manage user identities in a dual fashion—both through the local database and via Keycloak—thus leveraging all the benefits offered by Keycloak as an identity broker. However, the application had never been tested within the CLARIN environment, and reaching full compatibility required a collaborative effort. To this end, ILC4CLARIN acted as a service provider and actively contributed to the integration process, with the shared goal of aligning the application with the CLARIN federated authentication model.

From a technical perspective, this involved integrating INCEpTION with a federated identity broker that could validate users based on their home institution credentials and automatically provision access rights as needed. From a strategic standpoint, it also aligned the platform with the **FAIR principles**, especially in terms of reusability and accessibility of digital research tools.

Moreover, implementing SSO also supports broader goals related to the **sustainability and scalability** of research infrastructure services. A federated authentication system reduces administrative overhead, simplifies maintenance, and enhances security by centralizing credential management. It also facilitates the adoption of INCEpTION in diverse contexts, including externally hosted deployments, cross-national collaborations, and project-based infrastructures. As a result, this SSO integration not only improves the immediate user experience but also lays the foundation for more robust and extensible deployments in the future, especially within initiatives such as **H2IOSC** and **STARWARS**, where INCEpTION plays a key functional role.

1.4 – INCEpTION and the STARWARS Project

INCEpTION was adopted in the context of the **STARWARS project**[12] (*STormwAteR and WastewAteR networkS heterogeneous data AI-driven management*), a multidisciplinary European initiative aiming to tackle the growing challenge of managing heterogeneous and complex data related to urban water infrastructures. STARWARS brings together researchers from the fields of **Artificial Intelligence (AI)**, **Water Sciences**, **Geoinformatics**, and **Digital Engineering**, with the goal of developing innovative methodologies for representing, merging, reasoning, and querying across diverse data sources—including factual databases, geospatial information, technical reports, analogue maps, and time-variant sensor data.

Within this ambitious research framework, **INCEpTION** was adopted as a key component for the **semantic annotation of technical documentation and domain-specific corpora**, which are critical

[12] https://cordis.europa.eu/project/id/101086252

for enabling downstream knowledge extraction and AI-driven reasoning. For example, annotation workflows within STARWARS involved labeling terminologies, linking entities, and capturing context-specific metadata across multilingual, heterogeneous textual datasets. Such operations required a flexible, yet secure environment for multiple project stakeholders—including academic researchers, public utility representatives, and domain experts—to access and contribute collaboratively.

While Single Sign-On (SSO) was not implemented in the INCEpTION instance used by researchers to generate project data, integrating SAML-based SSO into this environment could offer significant application-level benefits—such as improved user experience, better access control, and streamlined collaboration. Nonetheless, the main relevance of this section lies in the application's adoption within the STARWARS project as a practical use case for collaborative semantic annotation in a distributed research setting.

1.5 – Interactions with H2IOSC project

The impact of the SSO implementation within INCEpTION extends significantly beyond the immediate needs of the platform or its deployment in the STARWARS project. It also plays a crucial strategic role within the framework of **H2IOSC**[13] (*Humanities and Cultural Heritage Italian Open Science Cloud*), a pioneering national initiative that aims to accelerate the **digital transformation** of research infrastructures operating in the domains of **humanities** and **cultural heritage**. Funded through the **European Union's Next Generation EU programme**[14] and Italy's **National Recovery and Resilience Plan (NRRP)**[15], H2IOSC represents a collaborative effort involving 12 research institutes from the National Research Council of Italy (CNR) and 18 operational units from **the Department of Social Sciences and Humanities, Cultural Heritage**[16] (CNR DSU).

H2IOSC's mission is to federate and enhance four major Italian nodes of European research infrastructures—**CLARIN-IT**, **DARIAH-IT**[17], **E-RIHS.it**[18], and **OPERAS-IT**[19]—by establishing a unified and FAIR-compliant digital research ecosystem. Within this large-scale endeavor, **INCEpTION's SSO integration was tested and validated in the context of the "Archivio ViVo" pilot**, one of the thematic demonstrators associated with **Work Package 7.2**. This pilot explored innovative models of access and data governance, and the successful incorporation of federated SSO served as a practical demonstration of how a single, standards-based authentication mechanism can streamline access to multiple research tools and services, especially those involving sensitive or restricted datasets.

The relevance of this work is further amplified by its alignment with the **core principles of Open Science and FAIR data governance**[20], both of which are pillars of the H2IOSC strategy. By introducing a **federated identity and access management system**, researchers, curators, and digital humanists

[13] https://www.h2iosc.cnr.it/
[14] https://next-generation-eu.europa.eu/index_en
[15] https://commission.europa.eu/strategy-and-policy/recovery-plan-europe_en
[16] https://www.dsu.cnr.it/
[17] http://dariah.cnr.it/
[18] https://www.e-rihs.it/
[19] https://osf.io/w2bp8/
[20] https://www.go-fair.org/fair-principles/

can access a range of interoperable services through a single identity, thereby reducing technical barriers and enabling more fluid collaboration across institutions and domains.

Importantly, the same SSO infrastructure developed and refined through the INCEpTION use case is now being considered for wider adoption within other **H2IOSC pilot services**, notably the **Linguistic Linked Open Data (LLOD) pilot**, which will provide **RDF-based linguistic resources** accessible via SPARQL endpoints and other Linked Data technologies. These resources will play a central role in semantic interoperability initiatives and the broader reuse of curated linguistic datasets. Implementing federated access control mechanisms such as SSO is critical to balancing openness with security, ensuring that users can seamlessly authenticate while maintaining appropriate levels of data protection, especially when dealing with licensed or sensitive content.

2 – Technological Framework and Authentication Infrastructure

This section provides a detailed examination of the technological environment underpinning the integration of **federated Single Sign-On (SSO)** within the INCEpTION platform. The implementation of SSO in research infrastructure contexts requires the careful orchestration of multiple components, including standardized authentication protocols, identity federation frameworks, and institutional service configurations.

We begin by outlining the **Security Assertion Markup Language (SAML)** protocol, which constitutes the foundational standard for secure identity exchange across organizations. The principles of SAML-based authentication are critical to understanding how identity is asserted, consumed, and trusted across federated environments.

Following this, we explore the **CLARIN Service Provider Federation (SPF)**, the identity federation framework adopted within the European CLARIN infrastructure. CLARIN SPF provides a structured trust network and a common policy framework that enables seamless authentication for researchers across national and institutional boundaries. Its alignment with **eduGAIN**, the global academic identity federation, ensures compatibility and scalability across the European research space. As part of this discussion, we will also consider the role of the Italian national academic and research network, **GARR**, and its contribution to federated identity management within the national context.

We then delve into the configuration of the **ILC4CLARIN Service Provider**, hosted at the CNR Institute for Computational Linguistics "A. Zampolli". This SP instance was built using **SimpleSAMLphp** and connected to **Keycloak**, both of which will be briefly introduced in the following section to clarify their respective roles and functionalities within the authentication architecture.

2.1 – The SAML Protocol: Principles and Adoption in Research Infrastructures

This section provides an introduction to the principles behind the SAML protocol and explores its strategic importance to the academic and scholarly communities. After outlining the basic building blocks and functionality of the protocol, the main roles involved in the SAML architecture will be described, specifically the Identity Provider (IdP), the Service Provider (SP), and any Attribute Authorities. The goal is to provide a clear and operational understanding of how SAML enables researchers and institutional users to access distributed resources securely and transparently, using credentials provided by their home organization.

2.1.1 – Overview of the Security Assertion Markup Language (SAML)

The **Security Assertion Markup Language (SAML)**[21] is an XML-based open standard developed by the OASIS consortium that enables the secure exchange of authentication and authorization data between identity providers (IdPs) and service providers (SPs). Originally designed to support **Single Sign-On (SSO)** in web-based environments, SAML has become a cornerstone protocol in federated identity management across academic, governmental, and enterprise domains.

At its core, SAML enables the **decoupling of user identity management from the services that consume identity assertions**, allowing users to authenticate once with their home institution and then access multiple services within the federation without needing to re-enter credentials. This makes it particularly suitable for large-scale, distributed research environments where interoperability, decentralization, and user convenience are critical.

SAML operates through a system of structured messages called **assertions**, which convey statements about a user's identity, authentication status, and optionally, their entitlements or attributes. These assertions are generated by a trusted **Identity Provider** and consumed by a **Service Provider**, which uses the information to make access control decisions.

The standard defines three core roles:

- **Identity Provider (IdP):** Authenticates the user and issues signed assertions.
- **Service Provider (SP):** Receives assertions and grants access based on them.
- **User (Principal):** The subject whose identity is being asserted.

A typical SAML-based authentication flow includes:

1. **User initiates access** to a protected resource on a Service Provider.
2. The SP redirects the user to a **Discovery Service** or directly to their **Identity Provider**.
3. The user authenticates at their IdP (via institutional credentials).
4. The IdP **generates and signs a SAML assertion** and redirects the user back to the SP.
5. The SP **validates the assertion**, extracts identity attributes, and grants access accordingly.

To ensure security and trust, SAML relies heavily on **public-key cryptography**, **metadata exchange**, and **digital signatures**. Federations—such as **eduGAIN**[22] or **IDEM GARR**[23]—facilitate the establishment of these trust relationships by maintaining a curated list of registered IdPs and SPs, along with their associated metadata.

In the context of research infrastructures, SAML has been widely adopted due to its compatibility with **federated identity frameworks** and its ability to enforce **privacy-preserving attribute release policies**. It enables researchers to use their home institution credentials to access services hosted

[21] https://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html
[22] https://edugain.org/
[23] https://www.idem.garr.it/

across national or international nodes, reducing friction and ensuring that authentication processes remain under institutional control.

By implementing SAML-based SSO in tools like INCEpTION, research communities benefit from **simplified access**, **scalability**, and **alignment with FAIR and Open Science principles**, ultimately fostering broader collaboration and data reuse.

2.1.2 – Roles of Identity Provider (IdP), Service Provider (SP), and Attribute Authorities

In a SAML-based federated authentication ecosystem, the interaction between different entities is governed by well-defined roles that ensure secure, scalable, and interoperable access control. The three primary actors involved are the **Identity Provider (IdP)**, the **Service Provider (SP)**, and optionally, the **Attribute Authority (AA)**. Each plays a crucial part in enabling trust and information flow across institutional and infrastructural boundaries.

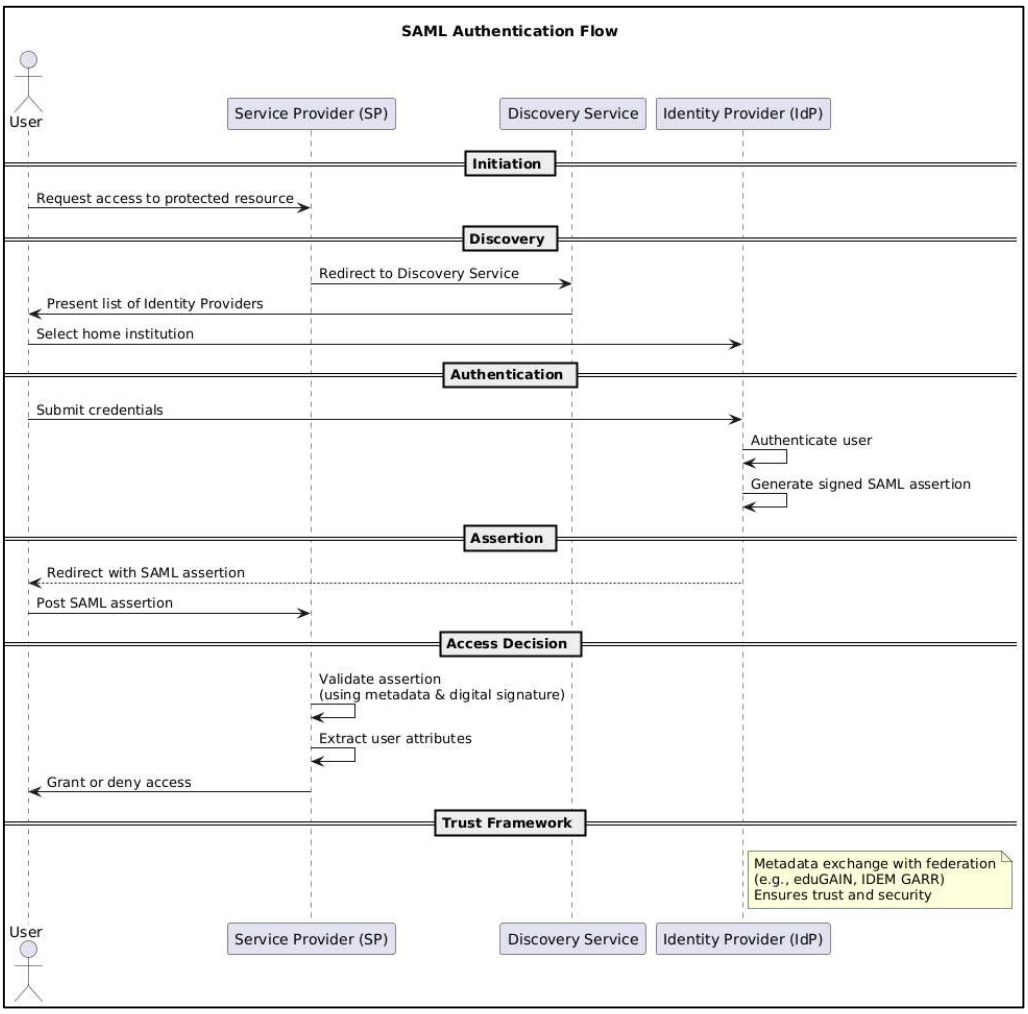


Figure 1: SAML workflow diagram

2.1.2.1 – Identity Provider (IdP)

The Identity Provider is the entity responsible for **authenticating the user** and issuing a digitally signed SAML assertion that contains statements about the authenticated user's identity. Typically operated by a university or research institution, the IdP is the point where the user logs in using their institutional credentials (e.g., username and password).

Key responsibilities of the IdP include:

- Verifying the identity of the user using local authentication methods.
- Generating SAML assertions that assert the user's identity and other attributes (e.g., name, affiliation, email).
- Digitally signing assertions to ensure their authenticity and integrity.
- Optionally, enforcing attribute release policies based on the requesting Service Provider.

In a federated context such as **eduGAIN** or **CLARIN SPF**[24], IdPs serve as trust anchors that enable users to seamlessly authenticate across multiple SPs using a single institutional identity.

2.1.2.2 – Service Provider (SP)

The Service Provider is the **consumer of SAML assertions** and the provider of the actual digital service or resource the user wants to access. SPs rely on the IdP to handle authentication and trust the IdP's assertion to make access control decisions.

Responsibilities of the SP include:

- Redirecting unauthenticated users to the appropriate IdP or Discovery Service.
- Accepting and validating incoming SAML assertions using federation metadata and public keys.
- Parsing user attributes from the assertion to determine access permissions or personalize the user experience.
- Enforcing local policies on session management, role assignment, or logging.

In research infrastructures, SPs include a wide range of services such as repository platforms, annotation tools (e.g., INCEpTION), data portals, and virtual research environments. By delegating authentication to IdPs, SPs avoid managing passwords and improve security compliance.

[24] https://www.clarin.eu/content/service-provider-federation

2.1.2.3 – Attribute Authority (AA)

An Attribute Authority is an optional but important SAML role that **issues additional user attributes** beyond what the IdP provides. While many IdPs act as both authenticators and attribute providers, federations that require complex attribute management may separate these concerns.

Responsibilities of an AA include:

- Hosting authoritative information about users (e.g., group memberships, roles, project affiliations).
- Responding to attribute queries from SPs or IdPs.
- Providing attributes in a signed and trusted manner.

In contexts such as **CLARIN**, an AA can be useful when service access depends on project-specific entitlements or when managing delegated access policies. Although not always implemented, the AA role increases flexibility and modularity in identity federation.

2.2 – Federated Authentication: The CLARIN Service Provider Federation (SPF)

This section explores the architecture and operation of the SPF federation adopted by CLARIN, illustrating its organizational principles, trust mechanisms among federated entities, and integration with national and international federations such as eduGAIN. Section 2.2.1 provides an introduction to the CLARIN SPF operating model, highlighting the technical requirements and guidelines for service providers wishing to join the federation. Section 2.2.2, on the other hand, focuses on how search services can be concretely integrated with CLARIN's Authentication and Authorization Infrastructure (AAI), illustrating the resulting functional, organizational, and accessibility benefits.

The overall objective is to clarify how the federated approach adopted by CLARIN enables effective management of users' digital identities, promoting collaboration among entities and facilitating access to European digital infrastructures in a uniform way that complies with international standards.

2.2.1 – Introduction to the CLARIN SPF

The **CLARIN Service Provider Federation (SPF)** is a dedicated **authentication and authorization infrastructure (AAI)** developed to facilitate **secure and seamless access to digital language resources and tools** offered within the CLARIN infrastructure. As a European research infrastructure committed to the principles of interoperability, sustainability, and openness, CLARIN has prioritized

the creation of a federated model that allows researchers to access services across national boundaries using their institutional credentials.

The CLARIN SPF is built upon the **SAML 2.0 protocol** and encompasses a set of national academic identity federations that are part of the broader eduGAIN inter-federation framework. Rather than being directly integrated into eduGAIN, the CLARIN SPF brings together these participating federations—such as IDEM GARR (Italy) and DFN-AAI (Germany)—under a common policy and trust framework. Through this structure, a researcher affiliated with any member institution of a supported national federation can authenticate once and access CLARIN services across nodes and countries, without the need for separate accounts or manual registration. CLARIN's federated authentication approach is characterized by several design principles:

- **Trust-based Federation Model**
  Participating IdPs and SPs are registered with their respective national federations and are trusted via metadata exchange and adherence to common policy frameworks. CLARIN provides an additional coordination layer to ensure consistent behavior across services.
- **Central Discovery Service**
  CLARIN operates a centralized **Discovery Service**, which helps users select their home institution during the login process. This improves usability and ensures the correct redirection of authentication requests to the appropriate IdP.
- **Service Provider Registry and Metadata Aggregation**
  CLARIN maintains its own metadata registry of approved service providers. These SPs, such as DSPACE or the Virtual Language Observatory, must meet technical and policy requirements in order to be included.
- **Attribute Release and Authorization Policies**
  The CLARIN SPF relies on a minimal, well-defined set of user attributes (e.g., eduPersonPrincipalName, eduPersonScopedAffiliation, mail) to enable user identification and authorization. Attribute release is governed by privacy-aware policies to ensure compliance with GDPR and national regulations.

The CLARIN SPF represents a mature and field-tested solution that addresses the specific needs of the linguistic and digital humanities research communities. It significantly reduces the complexity associated with identity and access management while promoting **user autonomy, cross-border collaboration, and adherence to Open Science principles**.

2.2.2 – Integration of Research Services with the CLARIN AAI

Integrating a research service with the **CLARIN Authentication and Authorization Infrastructure (AAI)** entails a combination of **technical configuration**, **policy compliance**, and **service alignment** with the federation's trust and interoperability framework. The process is designed to ensure that research tools and platforms can leverage the federated identity ecosystem while providing a consistent and secure user experience across the CLARIN landscape.

A typical integration workflow begins with the **registration of the service provider (SP)** within the metadata ecosystem maintained by the CLARIN SPF. This involves exposing a metadata file that describes the SP's entity ID, endpoints (e.g., assertion consumer service), supported bindings, certificates, and attribute requirements. Once verified, this metadata is aggregated by CLARIN's central registration service or by the national federation (e.g., IDEM GARR), and made available to participating identity providers via **signed and published metadata feeds**.

In parallel, the service must be technically capable of consuming SAML assertions and interpreting attributes defined by the federation's schema. While CLARIN officially supports and recommends well-established middleware such as Shibboleth SP, other SAML-compliant solutions can also be integrated, provided they adhere to the required standards. For instance, tools like Keycloak—although not explicitly referenced in CLARIN documentation—can act as a SAML proxy, enabling applications that do not natively support SAML to still benefit from federated authentication. This flexibility allows for the inclusion of a broader range of services, including legacy or lightweight applications, without requiring a complete redesign of their internal authentication mechanisms.

Once a service is technically ready and metadata has been exchanged, it must pass compliance checks covering secure communication (e.g., HTTPS, digital signatures), session management, and correct handling of user attributes. It must support attribute-based access control (ABAC), using standard attributes like eduPersonPrincipalName or eduPersonEntitlement to assign roles or permissions. For collaborative or multi-tenant services, CLARIN AAI also allows for role and group delegation beyond what is provided by the identity provider.

CLARIN offers guidance on key aspects such as attribute release, IdP discovery, GDPR compliance, incident handling, and sustainability planning. Full integration into the CLARIN AAI enhances a service's visibility, simplifies user access, promotes adoption, and ensures alignment with international standards for security and data protection.

2.3 – The GARR Infrastructure: National Research Identity Federation

This section analyzes GARR's strategic and operational contribution to the construction of the federated identity for the Italian academic and research system. In particular, Section 2.3.1 describes GARR's role as the national authority for the management of the IDEM federation, illustrating its governance activities, technical support and integration with the European eduGAIN network. In contrast, Section 2.3.2 focuses on the operational instance hosted in Palermo, one of the key test and development environments for federated services, highlighting its functionality, technical configuration and practical impact on the integration of new service providers.

2.3.1 – Role of GARR in Supporting Research Identity Services in Italy

The **Consortium GARR** plays a foundational role in the development and maintenance of the **digital identity infrastructure for the Italian academic and research community**. As Italy's **National**

Research and Education Network[25] (NREN), GARR provides not only high-capacity network connectivity but also a suite of trust, identity, and federation services that are essential to enabling secure and scalable authentication across institutions and research infrastructures.

GARR's contribution to digital identity services is anchored in the **IDEM Federation[26] (IDEntity Management)**, which it coordinates. IDEM serves as the Italian identity federation for research and education, and it is one of the national constituents of the **eduGAIN interfederation**. Through IDEM, GARR facilitates **federated access to digital resources**, allowing users from Italian institutions to authenticate using their home credentials across a variety of services, both national and international.

GARR supports the ecosystem through several strategic and operational functions:

- **Federation Governance and Coordination**
  GARR acts as the policy and operational authority for IDEM, managing participant onboarding, compliance verification, and the development of documentation and technical guidelines. It defines the **technical and legal framework** that binds Identity Providers and Service Providers in a trust-based relationship, ensuring compliance with national laws and international standards.

- **Metadata Aggregation and Publication**
  One of GARR's core responsibilities is to **aggregate and validate metadata** from all participating entities. This metadata, which contains the technical descriptors and trust anchors of Identity Providers and Service Providers, is digitally signed and published through trusted endpoints. This allows for **secure and dynamic trust establishment** among federation members.

- **Support for SAML Implementation and Troubleshooting**
  GARR provides **technical support and consultancy** to institutions implementing SAML-based authentication systems, offering best practices, diagnostic tools, and training resources. This support is particularly valuable for smaller institutions or specialized research centers integrating complex services, such as linguistic tools or data portals.

- **Infrastructure for Testing and Development**
  GARR maintains **testbed environments**—including the operational instance located in **Palermo**—which allow developers and administrators to test metadata configuration, attribute release, and federation interoperability before full production deployment. This feature has been essential in the iterative testing of services such as INCEpTION, reducing integration errors and improving federation compliance.

- **Promoting FAIR and Open Science Integration**
  Beyond technical roles, GARR actively contributes to the alignment of Italian identity infrastructure with **European Open Science initiatives**. Its participation in eduGAIN and collaboration with projects like EOSC, H2IOSC, and CLARIN-IT ensures that Italian researchers can **access and contribute to a federated, interoperable digital research space**.

[25] https://about.geant.org/nrens/
[26] https://www.idem.garr.it/

2.3.2 – Overview of the Palermo Instance: Functionalities and Operational Characteristics

The **Palermo instance**, provided through GARR's distributed infrastructure, is not a federation component in itself (e.g., it is not an IdP or SP), but rather a **virtualized computing environment** designed to support the **deployment and hosting of software services** relevant to the academic and research community. It functions as a **flexible and reliable cloud resource**, offering virtual machines (VMs) and hosting capabilities for applications that require secure, stable, and high-bandwidth infrastructure.

In the context of federated authentication and service integration, the Palermo instance has proven especially valuable by **providing an execution environment for components related to authentication, authorization, and identity management workflows**. For example, during the implementation of the INCEpTION SSO integration, a virtual machine hosted in Palermo was used to deploy:

- A **SimpleSAMLphp-based Service Provider**, customized and tested to interoperate with national and international federations;
- A **Keycloak identity broker**, acting as a bridge between the application and upstream Identity Providers;
- Auxiliary web services (e.g., application frontends, metadata viewers, diagnostic tools) needed to simulate real-world deployment scenarios.

The operational characteristics of the Palermo instance make it particularly well-suited for these use cases:

- **On-demand provisioning of VMs**, allowing rapid deployment and testing of authentication workflows in isolation or in conjunction with live federation components;
- **High-performance networking**, leveraging GARR's backbone to ensure low-latency communication with institutional IdPs or federation metadata registries;
- **Secure access policies**, including support for SSH key management, firewall rules, and TLS configurations appropriate for pre-production environments;
- **Persistence and monitoring**, allowing services to remain active over extended testing periods with visibility into performance and availability.

While it does not constitute a SAML entity itself, the Palermo instance plays a **critical infrastructural role** in enabling research institutions to host, test, and harden authentication-related services prior to integration into national or international federations. This model supports innovation, reduces deployment friction, and promotes best practices in secure, federation-aware software configuration.

In essence, Palermo provides **infrastructure-as-a-service (IaaS)** to the research and education community, empowering service owners to implement and refine identity solutions in a production-grade environment without sacrificing control or flexibility. Its contribution is thus infrastructural rather

than functional in the SAML protocol stack—but nonetheless essential for projects that require scalable and trusted hosting within the Italian academic network.

2.4 – The ILC4CLARIN Service Provider Instance

This section describes in detail the role and technical configuration of the ILC4CLARIN Service Provider, based on SimpleSAMLphp and managed by the Institute of Computational Linguistics "A. Zampolli." Section 2.4.1 introduces the architecture and main functionalities of the system, highlighting the technological and infrastructural choices adopted. Section 2.4.2 delves into the SAML metadata generation and publication processes, as well as compliance requirements for inclusion in federations such as CLARIN SPF. It continues, in Section 2.4.3, with the integration of the Service Provider within a proxy architecture via Keycloak, which allows for dynamic management of the interaction between external Identity Providers and applications that are not natively SAML-compliant. Finally, Section 2.4.4 analyzes the specific contribution of the ILC4CLARIN node within the CLARIN-IT national network, highlighting how this SP instance supports interoperability, scalability, and reusability of federated access within a broader digital services infrastructure.

2.4.1 – Overview of the SimpleSAMLphp-Based Service Provider at CNR-ILC

As part of the integration of INCEpTION into the CLARIN federation ecosystem, a dedicated Service Provider (SP) instance was deployed and configured by the CNR Institute for Computational Linguistics "A. Zampolli" (CNR-ILC) and hosted by the GARR network. The core component enabling this integration is **SimpleSAMLphp**, a widely adopted, open-source PHP-based[27] implementation of the SAML 2.0 protocol. Known for its flexibility and ease of configuration, SimpleSAMLphp serves as a robust framework for enabling SAML-based authentication flows between federated Identity Providers and web-based applications.

In this context, SimpleSAMLphp acts as the authentication gateway, handling the reception of SAML assertions, validating digital signatures, managing user sessions, and extracting identity attributes. Its modular architecture allows administrators to define a wide range of operational parameters through structured configuration files, making it particularly suitable for research environments with strict privacy, interoperability, and metadata conformance requirements.

The "proxy" (IdP+SP) instance was installed on a dedicated Linux-based virtual machine provisioned within the ILC4CLARIN infrastructure, designed specifically to host authentication-related services. The deployment included a standalone Apache server running PHP-FPM, with SimpleSAMLphp at its core. The configuration involved setting up the SP's entity ID and an Identity Provider, defining assertion consumer service (ACS) endpoints, selecting HTTP-POST bindings, and associating valid

[27] https://www.php.net/

X.509 certificates for signing and encryption—all handled via SimpleSAMLphp's intuitive directory structure and control files.

To ensure secure communications over the web, the virtual host was configured with HTTPS using TLS certificates issued and renewed via Let's Encrypt. However, for SAML-specific operations—such as signing and encrypting assertions, and publishing metadata—the system relies on self-signed X.509 certificates with a multi-year validity (typically three years). These certificates, distinct from transport-level encryption, are referenced within the SAML metadata and are used to establish cryptographic trust between the Service Provider and Identity Providers, in accordance with federation-level requirements.

The metadata representing the SP was generated using SimpleSAMLphp's built-in tools and customized to reflect CLARIN SPF's expected attributes and service categories. After schema validation, the metadata was submitted to CLARIN's registration process for eventual propagation into eduGAIN via the federation's trusted metadata feed. This step was essential to enable automated trust establishment with remote Identity Providers participating in IDEM GARR and other national federations.

Beyond federation registration, the SP instance was configured to process a minimal set of user attributes—such as eduPersonPrincipalName, eduPersonScopedAffiliation, and mail—mapping them to environment variables or internal roles via SimpleSAMLphp's attribute processing modules. This ensured that user identity information was handled in a privacy-conscious manner, respecting data minimization principles and aligning with CLARIN's attribute release policies.

The session management subsystem of SimpleSAMLphp was tuned for short-lived, secure interactions: cookies were scoped and flagged appropriately, session timeouts were enforced, and hooks were made available for future integration with role-based access control layers.

2.4.2 – Metadata Generation, Registration, and Federation Compliance

The integration of the ILC4CLARIN Service Provider into the federated authentication ecosystem required the generation of a SAML metadata file that accurately describes the SP's technical and organizational identity. This metadata includes essential information such as the entity identifier, endpoints for the assertion consumer service, supported bindings, cryptographic certificates, and administrative contact details. Its correctness and conformance are prerequisites for trust establishment across identity federations.

At CNR-ILC, the metadata for the SimpleSAMLphp-based SP was manually curated and validated to align with the specifications required by CLARIN's AAI coordination team. Rather than submitting this metadata directly to IDEM GARR, the standard procedure involves **registering the SP metadata with CLARIN**, which acts as the intermediary aggregator and validator for federation-specific configurations. CLARIN reviews the metadata to ensure compatibility with its federation policies and technical standards, and then **propagates it to the interfederation layer**, making it discoverable by national federations such as IDEM GARR through eduGAIN.

This architecture reflects CLARIN's role as a thematic service hub operating across national boundaries, capable of managing its own registry of Service Providers and contributing them to the

wider identity federation fabric. The metadata submitted through CLARIN is signed, included in CLARIN's metadata feed, and made available for consumption by all compliant Identity Providers participating in eduGAIN, including those within the IDEM GARR federation.

Maintaining federation compliance also required that the metadata adhere to expected security and trust practices, such as the use of valid, non-expired TLS certificates, correctly scoped attribute requirements, and the inclusion of descriptive elements (e.g., display names, privacy policies, and technical contacts). Where applicable, entity category tags were declared—such as Research and Scholarship or Code of Conduct—to facilitate attribute release by Identity Providers based on pre-negotiated trust frameworks.

This indirect, CLARIN-mediated registration process ensures that the ILC4CLARIN Proxy is aligned with both the thematic federation's requirements and the technical standards of the broader interfederation ecosystem. It also reinforces a model of **distributed trust coordination**, where domain-specific infrastructures like CLARIN act as integration facilitators between national identity federations and domain-level service providers.

2.4.3 – Integration with Keycloak and Proxy Configuration

The integration of the ILC4CLARIN Proxy with the INCEpTION platform made use of **Keycloak**, an open-source identity and access management system developed by Red Hat. Designed to support modern authentication and authorization scenarios, Keycloak provides a flexible and extensible platform capable of handling multiple identity protocols—including SAML 2.0, OpenID Connect, and OAuth 2.0—making it well suited for acting as a bridge between federated infrastructures and research applications.

At its core, **Keycloak functions as an identity broker and identity provider**, offering features such as user authentication, identity federation, session management, and role-based access control. One of its key strengths is its support for federated login: it can consume assertions from external Identity Providers and present them to applications in a format that they can readily process. It also includes a built-in user directory, support for multi-tenancy through the concept of "realms," and fine-grained control over identity attributes and access policies.

In the context of the INCEpTION integration, Keycloak was configured to act as an **intermediary SAML broker**. Rather than relying on INCEpTION to handle the complexities of interacting directly with federated Identity Providers, Keycloak took on the responsibility of processing SAML assertions issued by a SimpleSAMLphp-based proxy. It validated these assertions, extracted relevant identity attributes (such as user identifiers, affiliations, and email addresses), and provided a consistent SAML endpoint for INCEpTION to consume.

This approach offered multiple benefits. First, it **simplified the integration process**, insulating the application from federation-specific configurations. Second, Keycloak provided enhanced **attribute transformation and mapping capabilities**, enabling administrators to control how identity information was presented to services. Finally, its built-in **session management** allowed for coordinated login and logout operations, helping maintain a coherent user experience across the authentication chain.

Beyond the INCEpTION use case, the deployment of Keycloak opened up possibilities for broader reuse. Thanks to its modular architecture, other applications within the ILC4CLARIN infrastructure can be integrated with the same identity backend, reducing duplication of effort and enabling centralized identity policy enforcement. This makes Keycloak a strategic component in enabling federated access to web-based research tools, especially in infrastructures that prioritize interoperability, flexibility, and sustainability.

2.4.4 – Specific Role of ILC4CLARIN within the CLARIN-IT Node

Within the Italian CLARIN consortium (CLARIN-IT), the ILC4CLARIN centre, hosted by the Istituto di Linguistica Computazionale "A. Zampolli" of the CNR, serves both as a provider of linguistic resources and as an enabler of technical interoperability. As a certified CLARIN B-Centre, it ensures long-term access to language data and tools in line with CLARIN ERIC standards.

ILC4CLARIN distinguishes itself within CLARIN-IT through its role in federated access and service deployment. It supports the integration of services requiring authentication—such as INCEpTION—by maintaining SAML-compliant Service Providers and facilitating identity federation. The centre also contributes to the coordination of authentication flows at the national level, aligning configurations with IDEM GARR and CLARIN SPF policies.

Thanks to this infrastructure and expertise, ILC4CLARIN is involved in pilot implementations and supports the federation of services developed within national initiatives like H2IOSC and other collaborative projects.

3 – Architectural Design and Component Interactions

This section presents the architectural design of the federated authentication system deployed to integrate INCEpTION into the CLARIN Service Provider Federation. The architecture is built upon a containerized infrastructure, orchestrated through Docker[28] and managed via Portainer[29], with components logically and functionally separated across well-defined layers to ensure security, interoperability, and scalability.

The system is organized into distinct services: the application layer hosting INCEpTION, an intermediary identity brokering layer handled by Keycloak, and a federation interface layer based on a customized SimpleSAMLphp stack acting both as a Service Provider and an Identity Provider. All components are routed internally through an Apache reverse proxy and exposed via HTTPS, ensuring secure communication across the entire workflow. There is also a NGINX[30] component inside the SimpleSAMLphp acting like a reverse proxy for internal connections but this is not really necessary for the correct function of the stack.

The section begins with a high-level overview of the architecture, followed by a description of the *vocabs* infrastructure. It then focuses on the interactions between the individual components: how INCEpTION delegates authentication to a Keycloak instance; how Keycloak, in turn, processes SAML assertions received from a SimpleSAMLphp-based proxy; and how this proxy interacts with the CLARIN Discovery Service and external Identity Providers. A single metadata configuration is maintained for the Keycloak client, while federated IdPs are dynamically registered through metadata ingestion and transformation scripts.

To support a clearer understanding of data flows and trust relationships, this section includes architectural diagrams that illustrate both the static configuration and the dynamic behavior of the authentication sequence. From discovery to session propagation, the analysis provides insights into how each component contributes to a seamless, federation-compliant login experience.

3.1 – High-Level Architecture of the Authentication System

The architecture underpinning the authentication system deployed for the integration of INCEpTION within the federated environment is based on a **modular, layered design** and is entirely orchestrated through **Docker containers**. This containerized approach enables flexible deployment, simplified configuration, and environment isolation for each component, making the architecture both scalable and portable across different infrastructure setups.

At a high level, the system is structured into three principal tiers: the **federation interface layer**, the **brokering and transformation layer**, and the **application access layer**. Each tier is encapsulated in one or more Docker containers, ensuring consistency in deployment and easing the management of inter-component dependencies.

---

[28] https://www.docker.com/
[29] https://www.portainer.io/
[30] https://nginx.org/

The **first layer** consists of the components that directly interface with the federated identity infrastructure—namely, the CLARIN SPF and its connected national federations such as IDEM GARR. Within this tier, the **SimpleSAMLphp-based IdP + SP** runs in a dedicated Docker container and functions as the public-facing gateway. It handles incoming SAML authentication requests and responses, validates digital signatures, extracts standard identity assertions, and enforces federation compliance. This SP instance is registered in the CLARIN metadata and is trusted by the participating Identity Providers.

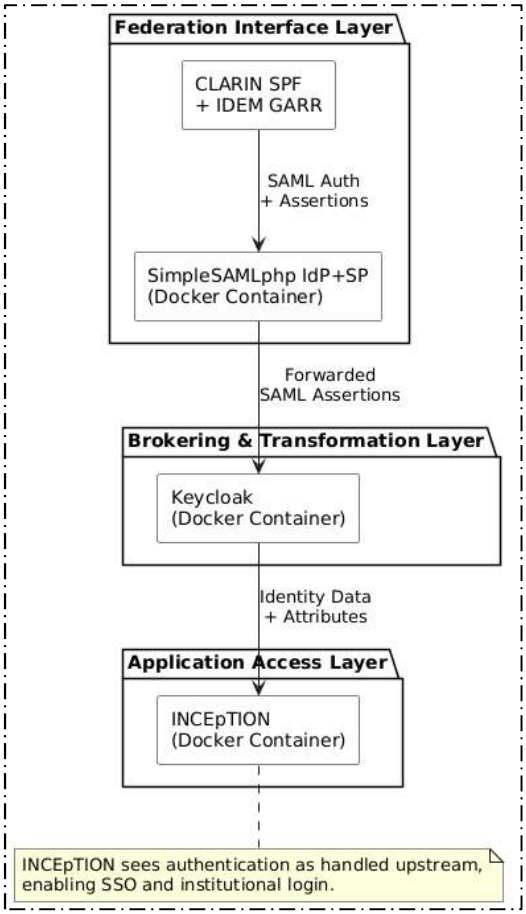


Figure 2: High level architecture

The **second layer** acts as the identity brokering module and is implemented using a **Keycloak container**, configured to process and relay SAML assertions. This layer separates the federated authentication logic from the application itself, enabling flexible user attribute mapping, normalization, and transformation. While INCEpTION is SAML-aware, this intermediary layer allows centralized identity control and supports future integrations with other services that may require protocol adaptation or advanced access policy enforcement.

The **third layer** comprises the **INCEpTION platform**, also containerized, which consumes the authentication assertions and initiates user sessions based on the identity data provided. From the application's point of view, authentication is handled by an upstream trusted identity service, abstracting away the underlying federation complexities. This loose coupling improves maintainability and ensures that INCEpTION can fully benefit from features like single sign-on (SSO) and institutional login.

Each component, though isolated in its own container, communicates securely with the others using clearly defined protocols and interfaces. HTTPS is enforced across all communications to guarantee data confidentiality and integrity. Trust within the federation is maintained through metadata validation, certificate management, and strict adherence to SAML protocol specifications. Session continuity is preserved across containers, with synchronized login/logout workflows coordinated by the Service Provider and Keycloak components.

This container-based architecture provides a reproducible, modular, and federation-compliant solution that is well-suited for complex research infrastructures such as ILC4CLARIN and adaptable to broader contexts like H2IOSC.

3.2 – Vocabs Architecture

The infrastructure hosted at **vocabs.ilc4clarin.ilc.cnr.it** provides a modular and containerized environment designed to support federated authentication services and web-based research applications within the ILC4CLARIN ecosystem. The entire system is orchestrated using Docker and managed through Portainer, ensuring high configurability, scalability, and operational isolation of its components.

Within this environment, each service runs inside a dedicated Docker container, organized into logical stacks. Network routing between containers is managed by an internal **Apache HTTP server operating as a reverse proxy**, which handles hostname- and path-based routing (e.g., /inception, /auth), enforces HTTPS communication, and serves as the main ingress point for all external requests. TLS protection is ensured through the use of self-signed certificates, providing encrypted communication within a trusted internal context.

A core element of the infrastructure is the **SimpleSAMLphp-based proxy stack**[31], deployed using the official Docker image. This stack is composed of two main components: a **Service Provider (SP)** and an **Identity Provider (IdP)**, both configured to support their respective roles in a SAML 2.0 federation context. The configuration has been customized to include federation-specific logic and metadata management using files such as `authsources.php`, `config.php`, and `saml20-sp-remote.php`. Metadata for federated Identity Providers is dynamically generated by processing the XML feed published by the CLARIN SPF, ensuring compatibility and continuous alignment with the broader CLARIN and IDEM GARR federations. These external federation services—including IdP discovery and metadata publication—are accessible via the CLARIN Discovery Service[32], which is referenced in the system configuration.

[31] In addition to the SimpleSAMLphp proxy—which internally includes both the SP and IdP components—a dedicated NGINX container is deployed to act as a reverse proxy. This component is responsible for routing HTTP/S traffic to the appropriate SimpleSAMLphp endpoints, decoupling transport-level concerns from SAML-specific logic and simplifying external access management.
https://github.com/simplesamlphp/docker-simplesamlphp

[32] https://discovery.clarin.eu/

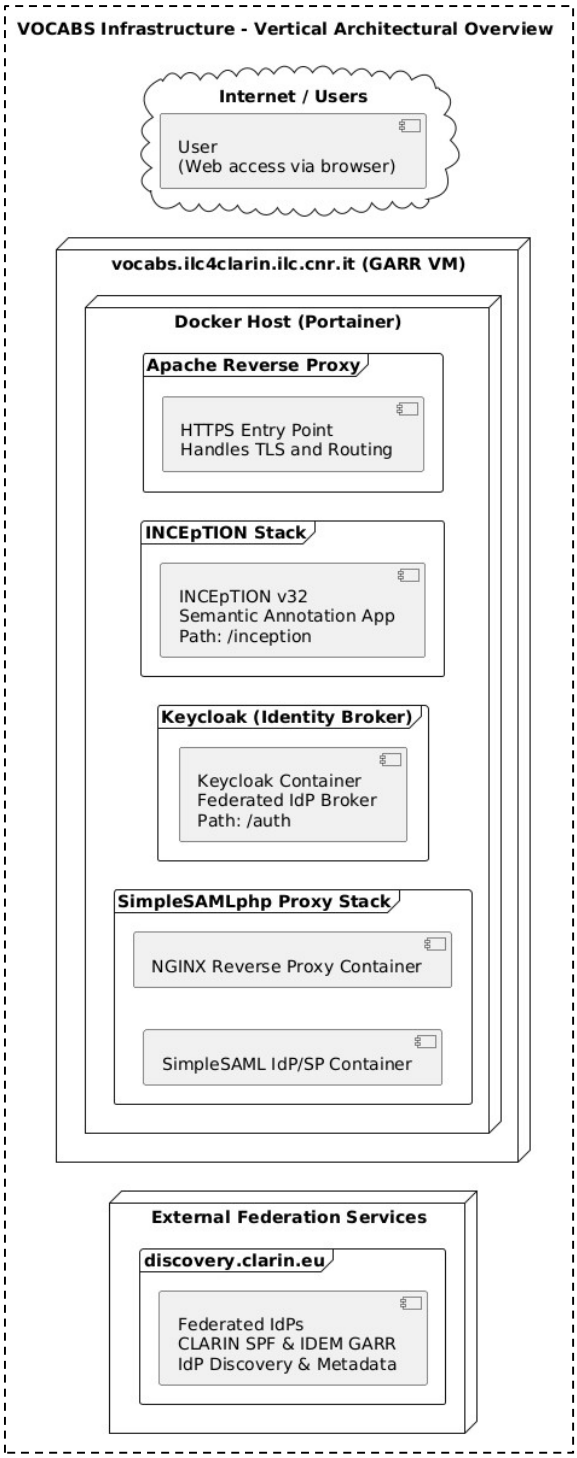


Figure 3: Vocabs architecture

On the application side, **INCEpTION** is deployed as a self-contained stack and exposed at the `/inception` path. The application is up to date with version 32 and configured via the `settings.properties` file to support SAML-based authentication. This configuration points to a SAML endpoint exposed by **Keycloak**, which serves as an intermediary identity broker.

Keycloak, although external to the primary Docker stack, is launched via a standalone docker run command and exposed at the `/auth` path. It functions as an identity bridge between the federated IdP infrastructure and INCEpTION, handling user sessions, attribute mapping, and authentication handoff.

This modular setup allows for clean separation of concerns and simplifies identity integration, supporting a scalable and maintainable authentication model across services within the ILC4CLARIN framework.

3.3 – Interactions between INCEpTION, Keycloak, the SP Proxy, and the Discovery Service

The authentication workflow is orchestrated through a layered system where **Keycloak**, **SimpleSAMLphp acting as a combined IdP-SP proxy**, and the **CLARIN Discovery Service** collaborate to enable federated login for INCEpTION users.

When a user attempts to log in to INCEpTION, the application redirects them to **Keycloak**, which operates as its internal identity provider. Keycloak, however, does not authenticate users directly in this scenario; instead, it delegates authentication to an external identity source—in this case, a **SimpleSAMLphp-based IdP** configured as a **proxy**.

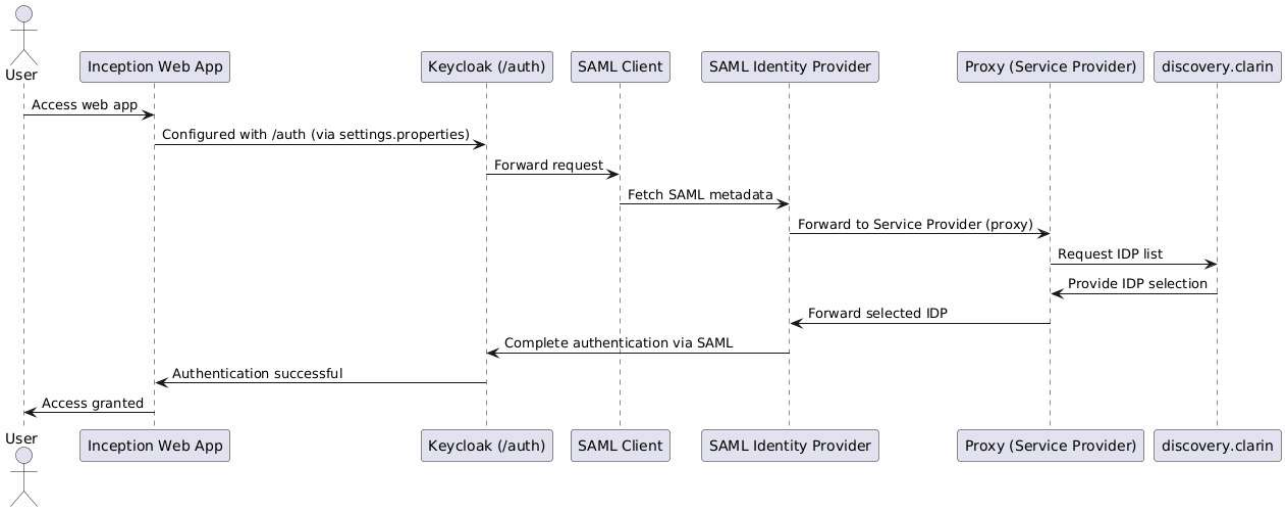


Figure 4: SAML workflow from INCEpTION app

This proxy IdP is tightly coupled with a **SimpleSAMLphp SP**, forming a self-contained identity brokering layer. The SP component of the proxy initiates the SAML authentication process by interacting with the broader CLARIN federation. Upon receiving the delegation from Keycloak, the proxy IdP transfers control to its internal SP, which then redirects the user to the **CLARIN Discovery Service**.

The Discovery Service allows the user to select their home institution. Once the selection is made, the SP within the proxy forwards the authentication request to the selected institutional Identity Provider. After successful authentication, the IdP issues a SAML assertion and returns it to the proxy's SP, which passes it back to the proxy IdP component. The IdP validates the assertion and extracts the identity attributes.

Once the user has been successfully authenticated by their institutional Identity Provider, and the SAML assertion has been validated by the Service Provider component of the proxy, the assertion is passed internally to the proxy's Identity Provider, which in turn communicates with **Keycloak**, configured as a **SAML Identity Provider** for INCEpTION. In this setup, Keycloak receives the SAML authentication request initiated by the application and delegates user authentication to the upstream federation through the proxy.

After the proxy IdP has completed the authentication with the user's institutional IdP and issued a valid assertion, Keycloak consumes this assertion as an external identity provider. It maps the federated attributes into its internal user model, applies any configured attribute transformations or role assignments, and establishes a session for the authenticated user. The entire flow is based on **SAML 2.0**, and there is **no protocol bridging** or conversion to OpenID Connect at any stage.

This configuration allows INCEpTION—which supports SAML natively—to interface directly with Keycloak for authentication, while keeping the federation logic and metadata processing encapsulated within the proxy infrastructure. This modular approach simplifies maintenance and supports broader integration within federated research environments.

4 – Implementation Workflow and Operational Logic for SSO Integration

This section provides a practical guide to implementing federated Single Sign-On (SSO) within an INCEpTION deployment, focusing on the configuration steps, operational logic, and architectural roles of each component involved in the authentication flow. Rather than covering full installation procedures, we assume that the core components—such as INCEpTION, Keycloak, and the proxy infrastructure—are already deployed within the *vocabs* environment. The emphasis will therefore be on configuring and fine-tuning these elements to enable a functional SAML-based integration, such as with the CLARIN Service Provider Federation.

The section begins by outlining the necessary technical prerequisites and available containerized services, then proceeds through the configuration phases—from enabling INCEpTION to communicate with the authentication infrastructure, to setting up Keycloak as a SAML-aware identity intermediary, and finally to adjusting the SimpleSAMLphp proxy stack.

In addition to step-by-step configuration, the section provides an overview of the operational mechanisms, including authentication request routing, assertion validation, and session management. More detailed installation and deployment instructions are addressed in dedicated sections. This approach is intended to guide system administrators and developers in implementing a robust and standards-compliant SSO environment suitable for federated research infrastructures.

4.1 – Prerequisites and Environment Setup

This section provides a comprehensive overview of the technical and infrastructure prerequisites required to deploy the system. Section 4.1.1 lists the necessary software components and key dependencies, including tools for identity management, encryption, and Docker support. Section 4.1.2 outlines the specifications of the virtual machine used, with particular attention to network configuration and traffic protection through firewall rules and TLS. Section 4.1.3 focuses on the Docker environment setup, covering the management of persistent volumes and the modular structuring of containers to facilitate reuse and maintainability. This initial setup phase lays the groundwork for the successful integration of federated Single Sign-On (SSO) mechanisms.

The subsequent sections describe the configuration of the main components involved in the authentication architecture. Section 4.2 discusses how to configure INCEpTION to support SAML-based authentication and properly manage user sessions. Section 4.3 addresses the setup of the SimpleSAMLphp proxy, which serves as the interface with the federated identity infrastructure. Section 4.4 explains the configuration of Keycloak as an identity broker, handling the reception and transformation of SAML assertions and enabling flexible user attribute management. Together, these sections offer a detailed and structured path toward the implementation of a robust and interoperable SSO system.

4.1.1 – Required software and dependencies

To implement the SSO integration described in this document, a set of open-source tools and system components must be installed and properly configured. The architecture relies on containerization to ensure modularity and ease of deployment. All key services run within isolated Docker containers, with routing and orchestration handled by a reverse proxy and managed via a container control panel.

Here is the list of components needed:

- **Docker Engine**
  Docker is required to containerize and isolate each component of the authentication infrastructure. All core services—SimpleSAMLphp proxy, INCEpTION, and Keycloak—are deployed as individual Docker containers.
- **Docker Compose**
  While individual containers can be started manually, Docker Compose simplifies the orchestration of multi-container stacks, particularly for the SimpleSAMLphp proxy setup.
- **Portainer (Web-based Docker Manager)**
  Portainer is used as the graphical interface for managing containers, monitoring logs, restarting services, and handling persistent volumes. It is optional but useful for system administrators unfamiliar with command-line Docker operations.
- **Apache HTTP Server[33] (Reverse Proxy)**
  A reverse proxy is required to route incoming HTTPS requests to the appropriate Docker containers based on domain name or URL path (e.g., `/inception`, `/auth`). Apache is the preferred option in this setup, though Nginx could be used as an alternative.
- **SimpleSAMLphp**
  The central component of the authentication proxy. SimpleSAMLphp must be deployed using its official Docker image and configured to act both as a Service Provider (SP) and Identity Provider (IdP), enabling federation compatibility and internal routing of SAML assertions.
- **Keycloak**
  Keycloak is used as a SAML-compliant Identity Provider for INCEpTION. It consumes SAML assertions issued by the proxy and provides session management and user mapping. The application is launched separately via docker run, outside of the main proxy stack.
- **INCEpTION**
  The target web application to be protected with federated SSO. Version 32 (or later) of INCEpTION is required, with SAML authentication enabled in its `settings.properties` configuration file.

Instead, here is a list of additional but no less important items:

[33] https://httpd.apache.org/

- **OpenSSL**[34]
  Used for generating and managing cryptographic keys and certificates required for signing and encrypting SAML assertions within the SimpleSAMLphp configuration.
- **Let's Encrypt** [35]**/ Certbot**
  For securing the Apache reverse proxy and enabling HTTPS for all exposed services. While self-signed certificates can be used in internal or testing environments, Let's Encrypt is recommended for production scenarios to ensure trust and compliance.
- **Git**[36]
  Used to clone configuration repositories, including the SimpleSAMLphp Docker image, sample configurations, and the INCEpTION deployment stack if managed via version control.

4.1.2 – Virtual Machine specifications and Network configuration

The deployment environment for the federated authentication system is hosted on a virtual machine provisioned through the GARR cloud infrastructure, specifically designed for research and academic purposes. The instance, named `h2iosc_lod_test`, is part of the broader H2IOSC test infrastructure and has been tailored to support lightweight virtualization and container orchestration for identity federation services and semantic web applications.

This virtual machine is based on the **Ubuntu 22.04 LTS** operating system and uses a standard image provided and optimized by GARR. The instance has been allocated a flavor of type m1.xlarge, providing a balanced hardware profile suitable for containerized deployments: 16 GB of RAM, 8 virtual CPUs, and 80 GB of root disk storage. This configuration ensures sufficient resources for running the Docker engine, managing containerized services such as SimpleSAMLphp, Keycloak, and INCEpTION, and maintaining acceptable performance during typical usage peaks.

The virtual machine is connected to the `clarin-it-VM-net` internal network and is externally reachable through a public interface. All critical ports required for HTTP/S traffic (e.g., ports 80 and 443), Docker-based administration via Portainer (port 9000), and SSH access are properly exposed through the associated security group rules. These rules have been configured to allow only essential inbound traffic, while maintaining open access on commonly used TCP ports for federated authentication and container management.

From a network standpoint, the configuration prioritizes both accessibility and containment: while public access is enabled for the services exposed via Apache (e.g., `/inception`, `/auth`), internal routing and isolation of containers is achieved through Docker networking and the reverse proxy. TLS encryption is enforced for all web-facing services using self-signed certificates, suitable for controlled research environments.

---

[34] https://www.openssl.org/
[35] https://letsencrypt.org/
[36] https://git-scm.com/

The instance is linked to a persistent storage volume of over 2 TB, which is mounted directly on the VM and used to store application data, federation metadata, logs, and container volumes. This guarantees data durability across reboots and facilitates system recovery and backup procedures.

The environment is designed to be stable, reusable, and replicable across different deployment contexts within the H2IOSC and ILC4CLARIN infrastructure, offering a flexible foundation for implementing and testing federated access scenarios in line with FAIR and Open Science principles.

Here's a **summary list** that captures the key characteristics of the virtual machine and its network environment:

- **Hosting Infrastructure**: GARR Cloud
- **Operating System**: Ubuntu 22.04 LTS (GARR-optimized image)
- **CPU**: 8 virtual CPUs
- **RAM**: 16 GB
- **Disk**: 80 GB root disk + 2 TB attached volume (persistent storage)
- **Deployment Purpose**: SSO integration testing within H2IOSC/CLARIN infrastructure
- **Containerization**: Docker-based with Portainer for stack management
- **Web Services Exposure**:
    - Apache reverse proxy with HTTPS enforcement
    - Port mapping includes: HTTP (80), HTTPS (443), Portainer (9000)
- **TLS Security**: Self-signed certificates for secure communication
- **Inbound Access Rules**:
    - SSH access restricted to selected IP ranges
    - HTTP/S and application ports open for public and federated use
    - ICMP allowed for basic network diagnostics
- **Network Integration**:
    - Internal network: `clarin-it-VM-net`
    - Configured for both internal Docker routing and public access
- **Use Case Focus**:
    - Hosting of SimpleSAMLphp (SP and IdP proxy)

    - Keycloak identity broker
    - INCEpTION semantic annotation platform

- **Storage Role**:
    - Mounted persistent volume used for federation metadata, logs, and container volumes
    - Ensures durability and backup across sessions

4.1.3 – Docker Environment and Volume Setup

The authentication system is designed around a containerized architecture, and as such, careful planning of the Docker environment is essential to ensure modularity, data persistence, and smooth service orchestration. The virtual machine serves as the Docker host and runs multiple stacks simultaneously, each associated with a specific service (e.g., SimpleSAMLphp, INCEpTION, Keycloak).

The Docker daemon is configured to run as a system service, with administrative access handled through Portainer. Portainer provides a user-friendly web interface that facilitates the deployment, monitoring, and management of containers, especially useful when working with multiple, isolated stacks.

Each application stack—whether it's the authentication proxy or the web application—is encapsulated within its own set of containers. These containers share a common network namespace defined within Docker, allowing for internal communication between services while remaining decoupled from the host system.

To ensure configuration and user data are not lost across restarts or updates, **named Docker volumes** are employed extensively throughout the infrastructure. Volumes are mounted into the containers to store critical assets such as:

- SimpleSAMLphp metadata and custom configuration files
- TLS certificate directories used by the reverse proxy
- Logs and session state for monitoring and diagnostics
- Keycloak realm configuration and local user database (if applicable)
- INCEpTION data repositories and system configuration

These volumes are stored on the mounted 2 TB persistent disk attached to the VM, ensuring durability across container lifecycle operations and reboot scenarios. The volume mount paths are managed explicitly in each container’s deployment specification, either via Docker Compose or manually through Portainer’s interface.

To further isolate service contexts and simplify troubleshooting, each stack is logically grouped and named according to its function (e.g., inception-stack, docker-simplesamlphp, keycloak-standalone). This naming convention helps maintain clarity in multi-container environments and supports reproducibility of the setup in other virtualized deployments within the infrastructure.

4.1.3.1 INCEpTION Docker Compose Configuration

The INCEpTION platform is deployed as a containerized application using a minimal yet production-ready Docker Compose configuration. The stack is composed of two primary services—**db** and **app**—connected via a dedicated Docker network called inception-net. The configuration ensures isolation, persistent data storage, and health-dependent startup sequencing.

The container of MariaDB 10.7 is responsible for handling all relational data storage for the INCEpTION platform. Its key characteristics are:

- **Environment variables** are used to set database credentials and initialize the inception schema.
- The **root password** is generated randomly for security using `MYSQL_RANDOM_ROOT_PASSWORD=yes`.
- A **named volume** (db-data) ensures that database content is persisted outside of the container's lifecycle, allowing safe restarts and upgrades.
- The command section enforces UTF-8 compatibility using `utf8mb4`, which is critical for storing multilingual data and annotations.
- A **healthcheck** is defined to monitor the readiness of the database. This ensures the application container (app) waits until the database is healthy before starting.

The container of INCEpTION web application use the image hosted on GitHub Container Registry (ghcr.io). It is configured with the following key features:

- The **application port** inside the container (8080) is mapped to a configurable host port, typically `${INCEPTION_PORT:-8085}`.
- Database connectivity is defined using standard JDBC variables:
    - `INCEPTION_DB_URL` references the internal service name db.
    - SSL is disabled (`useSSL=false`) for internal communication.
- A volume (`app-data`) is used to persist project data and user uploads (e.g., annotations, models) outside of the container.
- The container waits for the database to become healthy via `depends_on` with condition: `service_healthy`, avoiding premature startup.
- The **restart policy** is set to unless-stopped, which ensures resilience across reboots or daemon restarts.

By network side**,** A dedicated **Docker network** (`inception-net`) is defined to allow secure communication between containers without exposing database ports externally.

Two named **volumes** (`app-data` and `db-data`) are created for data persistence, ensuring that both application data and database content survive container restarts or rebuilds.

This Docker-based setup offers a straightforward, reproducible deployment method for INCEpTION that supports efficient data management, flexible configuration, and modular extension (e.g., SSO integration with Keycloak). It is well-suited for both testing environments and production deployments within the broader ILC4CLARIN infrastructure. You can find a full version of the docker-compose code in Appendix A1.

4.1.3.2 Keycloak Docker Compose Configuration

Keycloak is deployed as a standalone containerized service, accompanied by a dedicated PostgreSQL database instance. This configuration enables a flexible and self-contained identity management environment tailored for federated authentication scenarios. The system uses Docker Compose to orchestrate both services with clear separation of responsibilities and persistent data management.

The postgres container provides the database backend required by Keycloak. Its configuration includes:

- **Image**: Uses version 11 of the official PostgreSQL image, ensuring compatibility with Keycloak 14.
- **Container name**: `keycloak_db` for clarity and internal service resolution.
- **Volumes**: A bind mount (`./keycloak/data/`) maps host storage to the container's data directory to persist user and realm data across container restarts.
- **Environment variables**:
    - `POSTGRES_DB`: Creates a database named keycloak.
    - `POSTGRES_USER` and `POSTGRES_PASSWORD`: Define the credentials used by Keycloak to connect.
- **Ports**: Maps port 5432 (PostgreSQL) to 18056 on the host—useful for direct administrative access or debugging.
- **Restart policy**: Set to always, ensuring that the container restarts automatically after a failure or host reboot.

The keycloak container runs the actual Keycloak identity server based on version 14.0.0. Its configuration includes:

- **Image**: `quay.io/keycloak/keycloak:14.0.0`, an official build from the upstream distribution.
- **Environment variables**:

    - `DB_VENDOR, DB_ADDR, DB_DATABASE, DB_USER, DB_PASSWORD`: These parameters configure the database connection using the PostgreSQL service.
    - `KEYCLOAK_USER` and `KEYCLOAK_PASSWORD`: Set up an initial administrative user (admin) for accessing the Keycloak web console.
    - `PROXY_ADDRESS_FORWARDING`: Enables proper header handling when Keycloak is deployed behind a reverse proxy.
- **Volumes**:
    - Two customized configuration files (`standalone.xml` and `standalone-ha.xml`) are mounted directly into Keycloak's configuration directory, allowing fine-grained control over the JBoss runtime settings.
- **Ports**: Maps port 8080 inside the container to 8180 on the host, making the Keycloak interface accessible via `http://localhost:8180` (or the appropriate public hostname).
- **Restart policy**: Also set to always.
- **Dependencies**: The container waits for the PostgreSQL database to be up via depends_on, ensuring proper startup sequencing.

You can find a complete version of `docker-compose.yml` in Appendix A2.

4.1.3.3 SimpleSAMLphp Docker Compose Configuration

To deploy the authentication proxy based on SimpleSAMLphp in a reproducible, modular, and portable manner, a dedicated Docker Compose environment has been created. This environment is composed of two primary services defined in the `docker-compose.yml` file: a custom-built SimpleSAMLphp instance acting as a Service Provider (SP), and an Nginx container responsible for managing incoming HTTP(S) requests and forwarding them appropriately within the internal Docker network.

The core component of this setup is the SimpleSAMLphp service, defined under the name `sp.ilc4clarin.ilc.cnr.it`. This container is **built from a custom Dockerfile** located in the `build/` directory and is configured to serve the SAML authentication proxy on the hostname **sp.ilc4clarin.ilc.cnr.it**. It leverages a PHP 8.1 base image with Apache, and is extensively tailored for secure SAML-based identity federation use cases.

The Dockerfile used to build the SimpleSAMLphp container ensures a robust and production-ready environment by including a number of essential enhancements:

- **Base System Setup**: The image starts from `php:8.1-apache`, ensuring compatibility with modern PHP applications. Apache is configured to serve content from `/code/public`, which is aligned with the working directory used in the container.

- **System Dependencies**: Core utilities and libraries—such as `curl, git, gnupg, libpng-dev, zlib1g-dev,` and `libssl-dev`—are installed to support PHP extensions and runtime operations needed by SimpleSAMLphp and the underlying PHP engine.
- **PHP Extensions**: The container includes a full stack of PHP extensions relevant to web security and performance, including `gd, mbstring, mysqli, pdo,` and `pdo_mysql`. These are compiled and enabled during the image build.
- **PECL Extensions**: Both `memcached` and `redis` extensions are installed via PECL to support future session management or caching strategies.
- **Xdebug Integration**: To aid in debugging and development, the container builds and installs **Xdebug 3.1.6** manually from source, and configures it via an `.ini` file placed into the PHP configuration directory.
- **Composer Installation**: Composer is installed globally, allowing dependency management or future integrations with PHP packages during runtime or further customization.
- **Apache Configuration**: Apache is explicitly configured to use `mpm_prefork` (instead of the default `mpm_event`) to better support PHP module operation. Custom versions of apache2.conf and `mpm_prefork.conf` are copied into the image, ensuring alignment with the SAML proxy's performance and threading requirements.
- **Startup Script**: A custom startup.sh script is provided and marked as executable. This script is designated as the entry point to ensure Apache starts with the desired runtime configuration.

At runtime, several binding points are mounted into the container to ensure flexibility and persistence:

- `./simplesamlphp:/code`: This binding point contains the core SimpleSAMLphp application, including all relevant libraries, web assets, and internal modules.
- `./proxy:/conf`: This directory includes all key configuration files, such as:
    - `authsources.php`: Defines authentication sources and their corresponding settings.
    - `config.php`: The main configuration file governing core logic, paths, and logging.
    - Instead of a single `saml20-sp-remote.php` and `saml20-idp-remote.php` files, the system uses a `metadata` directory that contains all metadata definitions for both remote Identity Providers and remote Service Providers. The latter correspond to applications for which the proxy IdP provides federated authentication services.
- /etc/letsencrypt:/etc/letsencrypt: Although TLS termination is typically handled by Nginx, the certificates issued via Let's Encrypt are also made available within the SimpleSAMLphp container.

The environment variable SIMPLESAMLPHP_CONFIG_DIR is set to `/conf/`, pointing SimpleSAMLphp to the custom-mounted configuration directory. This design allows administrators to modify service behavior, adjust federation parameters, and reload metadata without rebuilding the container image, fostering rapid iteration during deployment and testing.

Complementing the SAML proxy is a dedicated nginx service, defined in the same `docker-compose.yml`. This container plays a pivotal role in securely exposing the SimpleSAMLphp interface to the outside world. Its responsibilities include:

- **Request Routing**: The Nginx container links to the SimpleSAMLphp container internally and routes inbound requests based on predefined path or hostname rules.
- **TLS Termination**: Port 2443 is exposed to the host and intended for HTTPS traffic. Let's Encrypt certificates can be mounted into the container to provide encryption for external requests. Although the corresponding volume is currently commented out in the example, it can be enabled if needed (`/etc/letsencrypt`).
- **HTTP Exposure**: Port 84 is mapped for development or local access over plain HTTP, simplifying testing before enabling strict HTTPS.
- **Configuration Management**: The directory `./nginx` is mounted read-only into the container to provide custom configuration files. These files may define upstream services, server blocks, and TLS behavior, ensuring that Nginx operates in alignment with the overall authentication infrastructure.

4.2 – Configuration of INCEpTION

This section focuses on the configuration steps required to enable federated authentication within the INCEpTION platform. Given that INCEpTION does not natively support SAML, the integration relies on a pre-configured identity broker—Keycloak—to handle SAML assertions and session management. The goal of this setup is to allow INCEpTION to authenticate users based on the SAML 2.0 assertions issued by remote institutional Identity Providers, without altering its core architecture.

The section outlines the necessary changes to INCEpTION's `settings.properties` file, highlighting the parameters that govern its interaction with the Keycloak instance. These include SAML endpoint references, realm and client configuration details, and any additional flags required for secure and compliant integration.

It is important to note that several elements of this configuration—such as the registration of the Keycloak Identity Provider, the configuration of client secrets, and the handling of SAML certificates—are based on guidance provided in the official INCEpTION documentation, specifically the Administrator Guide available at: https://inception-project.github.io/releases/33.0/docs/admin-guide.html#sect_security_authentication_saml2. Readers are encouraged to refer to that source for complete and up-to-date instructions, especially for aspects that may vary based on version or deployment context.

In this technical report, we focus on the configuration elements that had the most significant impact on the successful implementation of SSO integration, providing practical insights into their role and application within the broader infrastructure.

4.2.1 – Enabling SAML authentication in settings.properties

To enable SAML authentication in INCEpTION, the key component to configure is the `settings.properties` file located in the root directory of the INCEpTION application container. This file serves as the primary configuration hub for customizing runtime behavior, including the activation and management of external identity providers.

For a successful SAML integration, INCEpTION must be configured to delegate authentication to an external identity broker—in this case, **Keycloak**—which itself has been previously configured to accept and validate SAML assertions issued by the upstream proxy service. The integration does not require any modification to the core INCEpTION source code; it is entirely managed through properties exposed via the application's configuration file.

Below is an example of the minimal set of parameters to be added or modified in `settings.properties` to enable SAML authentication:

```
spring.security.saml2.relyingparty.registration.inception-client.assertingparty.entity-id=https://vocabs.ilc4clarin.ilc.cnr.it/auth/realms/inception-demo
spring.security.saml2.relyingparty.registration.inception-client.assertingparty.verification.credentials[0].certificate-location=file:/export/keycloak-saml-idp.crt
spring.security.saml2.relyingparty.registration.inception-client.assertingparty.singlesignon.url=https://vocabs.ilc4clarin.ilc.cnr.it/auth/realms/inception-demo/protocol/saml
spring.security.saml2.relyingparty.registration.inception-client.assertingparty.singlesignon.sign-request=false
logging.level.org.springframework.security.oauth=TRACE
logging.level.org.springframework.security.saml2=TRACE
logging.level.de.tudarmstadt.ukp.inception.security.oauth=TRACE
logging.level.de.tudarmstadt.ukp.inception.security.saml=TRACE
```

Here is a table summarizing the properties regarding the parameters of the authentication-side web application:

Configuration	Explanation
`spring.security.saml2.relyingparty.registration.inception-client.assertingparty.entity-id=https://vocabs.ilc4clarin.ilc.cnr.it/auth/realms/inception-demo`	*Defines the Entity ID of the SAML Identity Provider (IdP) — in this case, the Keycloak realm acting as the asserting party. This value must match the entityID defined in the IdP's metadata.*
`spring.security.saml2.relyingparty.registration.inception-client.assertingparty.verification.credentials[0].certificate-location=file:/export/keycloak-saml-idp.crt`	*Specifies the location of the public X.509 certificate used by INCEpTION to verify the SAML assertions received from the IdP. This certificate must match the one used by Keycloak to sign its SAML responses.*
`spring.security.saml2.relyingparty.registration.inception-client.assertingparty.singlesignon.url=https://vocabs.ilc4clarin.ilc.cnr.it/`	*Indicates the Single Sign-On (SSO) URL exposed by Keycloak for SAML logins. This is where INCEpTION sends its authentication requests.*

`auth/realms/inception-demo/protocol/saml`	
`spring.security.saml2.relyingparty.registration.inception-client.assertingparty.singlesignon.sign-request=false`	*Disables signing of SAML authentication requests sent by INCEpTION to the IdP. Some IdPs (like default Keycloak SAML setups) don't require signed requests, though enabling it may be preferred in production environments.*

Instead, this table describes the properties about the logging of the web-server:

Configuration	Explanation
`logging.level.org.springframework.security.oauth=TRACE`	*Enables verbose logging for Spring's OAuth2 support. Included here likely for environments that also support OAuth/OIDC.*
`logging.level.org.springframework.security.saml2=TRACE`	*Enables detailed logs for all SAML-related Spring Security operations. Essential for debugging authentication flows and assertion parsing.*
`logging.level.de.tudarmstadt.ukp.inception.security.oauth=TRACE`	*Activates trace logs for INCEpTION's OAuth-specific components.*
`logging.level.de.tudarmstadt.ukp.inception.security.saml=TRACE`	*Activates trace logs for INCEpTION's SAML integration layer, helping track the SAML authentication flow within the application.*

4.2.2 – Add a custom context path

To serve the INCEpTION application under a specific path (such as `/inception`) on a production domain like **https://vocabs.ilc4clarin.ilc.cnr.it**, it is necessary to modify the context path setting in the INCEpTION configuration file. This is done by editing the `settings.properties` file and setting the following parameter:

```
server.servlet.context-path=/inception
```

This directive ensures that the INCEpTION web application is accessible at the URL path /inception. As a result, users navigating to **https://vocabs.ilc4clarin.ilc.cnr.it/inception** will be correctly routed to the application.

However, this change must be reflected in the reverse proxy configuration as well—specifically in the Apache server that handles HTTPS requests. Apache must be configured to forward requests arriving at /inception to the appropriate internal container running INCEpTION.

Below is an example of a suitable Apache virtual host configuration:

```
<VirtualHost *:443>
    ServerName vocabs.ilc4clarin.ilc.cnr.it
```

```
    SSLEngine on
    SSLCertificateFile /etc/letsencrypt/live/vocabs.ilc4clarin.ilc.cnr.it/fullchain.pem
    SSLCertificateKeyFile /etc/letsencrypt/live/vocabs.ilc4clarin.ilc.cnr.it/privkey.pem
    Include /etc/letsencrypt/options-ssl-apache.conf

    RewriteEngine On
    ProxyRequests Off
    ProxyPreserveHost On

    RequestHeader set X-Forwarded-Proto "https"
    RequestHeader set X-Forwarded-Port "443"

    <Location "/inception">
        RequestHeader set "X-Forwarded-Proto" expr=%{REQUEST_SCHEME}
        RequestHeader set "X-Forwarded-SSL" expr=%{HTTPS}

        ProxyPass http://inception-app-1:8080/inception upgrade=websocket
        ProxyPassReverse http://inception-app-1:8080/inception
    </Location>
</VirtualHost>
```

This setup ensures that incoming HTTPS requests to /inception are securely forwarded to the internal INCEpTION container, preserving headers and allowing for proper session handling and redirection. It is important that both the INCEpTION application and the reverse proxy are aligned in their handling of the context path to ensure stable and predictable behavior in a production environment.

4.3 – Configuration of the SimpleSAMLphp Proxy

This section provides an overview of the key configuration aspects of the **SimpleSAMLphp proxy**, a central component in the federated authentication architecture adopted within the ILC4CLARIN infrastructure. The proxy is responsible for mediating between external Identity Providers—discovered via the CLARIN SPF federation—and internal authentication consumers, such as Keycloak or other service endpoints. It is configured to simultaneously act as a **Service Provider (SP)** and an **Identity Provider (IdP)**, allowing it to relay SAML assertions in both directions and enabling seamless single sign-on (SSO) across multiple services.

While the complete installation steps and exhaustive list of configuration files are documented in a dedicated technical report available at https://zenodo.org/records/15544532, this section focuses only on the most critical elements relevant to the functioning of the proxy. The goal is to highlight the files and parameters that directly affect interoperability, federation alignment, and service security. Readers are strongly encouraged to consult the full report for a comprehensive setup guide.

Through this configuration, the SimpleSAMLphp proxy becomes a fully compliant and reusable authentication gateway for research applications requiring access control based on federated identity standards.

Among the main configuration files used in the SimpleSAMLphp proxy, config.php plays a central role in defining global parameters for the system. This includes setting the base URL of the proxy according to its domain name, specifying the location of TLS certificates issued by Certbot for secure communication and assertion signing, and establishing session management policies to ensure stability and compliance with federation standards.

The `authsources.php` file is equally important, as it governs how authentication flows are initiated and processed. In this setup, the Identity Provider (IdP) is not hardcoded; instead, it is left undefined so that users are redirected to the CLARIN Discovery Service, allowing them to select their home institution dynamically. The configuration also points to OpenSSL private keys and certificates to secure SAML exchanges (included in the same location of the LetsEncrypt certificates of the Host Apache web-server). Furthermore, this file includes essential metadata describing the service provider itself, such as its name, description, and privacy policies, which help present the service clearly within the federation.

To manage metadata in a scalable and maintainable way, the proxy adopts a modular folder-based configuration specified in the `metadata.sources` section of `config.php`. This allows different sources of metadata—both local and from external federations like CLARIN and DFN—to be handled separately and updated dynamically through the `metarefresh` module.

Finally, the `saml20-sp-remote.php` file contains the metadata for the only remote Service Provider currently defined in the system: the Keycloak client (inception-client). This entry, based on Keycloak's published SAML descriptor, establishes the connection between the internal IdP of the proxy and the Keycloak instance used to manage authentication for INCEpTION.

4.3.1 – Editing config.php

The config.php file is the central configuration file in SimpleSAMLphp, responsible for defining core parameters that govern the behavior of the proxy component. This includes settings for the proxy's URL, certificate handling, session policies, storage mechanisms, and metadata structure. Proper configuration of this file is critical for ensuring that the authentication proxy behaves as expected and complies with federation standards, particularly in a federated setup like CLARIN SPF.

Below is a summary of the most relevant configurations applied in the deployed proxy setup:

Parameter	Value	Description
`baseurlpath`	`'https://sp-dev.ilc4clarin.ilc.cnr.it/'`	*Sets the base URL of the proxy, which must match the public-facing domain.*
`certdir`	`'/etc/letsencrypt/live/sp-dev.ilc4clarin.ilc.cnr.it'`	*Path to the TLS certificates used for signing SAML assertions.*
`assertion.validity`	`28800`	*Defines the lifespan of a SAML assertion in seconds (8 hours).*
`session.phpsession.httponly`	`true`	*Ensures that session cookies are flagged as HTTP-only for security.*
`store.type`	`'phpsession'`	*Defines the method for storing session state; here, it uses native PHP sessions.*

session.cookie.secure	true	*Enforces secure cookies for all sessions, requiring HTTPS connections.*

metadata.sources	array(
	  array('type' => 'flatfile', 'directory' => '/conf/metadata/idp-hosted'),
	  array('type' => 'flatfile', 'directory' => '/conf/metadata/sp-hosted'),
	  array('type' => 'flatfile', 'directory' => '/conf/metadata/dfn'),
	  array('type' => 'flatfile', 'directory' => '/conf/metadata/clarin'),
	  array('type' => 'flatfile', 'directory' => '/conf/metadata/clarin-idp')
	)
		Specifies folders used to load metadata (IdPs, SPs, CLARIN, DFN). Supports modular updates.

module.enable	array(
	  'exampleauth' => TRUE,
	  'saml' => TRUE,
	  'admin' => TRUE,
	  'core' => NULL,
	  'cron' => TRUE,
	  'metarefresh' => TRUE
	)
		Activates essential modules including those for testing, admin, metadata refresh and scheduled tasks.

4.3.2 – Editing authsources.php

The `authsources.php` file in SimpleSAMLphp is responsible for defining how authentication is initiated and processed by the proxy Service Provider. It contains critical directives such as which Identity Provider (IdP) to use, how to connect to the Discovery Service, and which certificates to employ for secure communications. Within the context of the CLARIN SPF integration, this configuration file has been carefully adjusted to reflect federation-compliant behavior while allowing for dynamic institution selection and secure assertion handling.

This section provides an overview of the essential parameters defined in the file, each of which contributes to ensuring that the proxy Service Provider operates correctly within a federated identity environment.

Parameter	Exact Value	Description
`"idp"`	`'null'`	*Leaves the IdP undefined to force the system to route authentication via the Discovery Service.*
`"discoURL"`	`'https://discovery.clarin.eu/'`	*Specifies the CLARIN Discovery Service used by users to select their institutional IdP.*
`"privatekey"`	`'privkey.pem'`	*Path to the private key file used for signing SAML assertions; generated with OpenSSL.*
`"certificate"`	`'cert.pem'`	*Path to the corresponding public certificate; also generated using OpenSSL and used in metadata.*
`Service Provider Info`	`DisplayName, Description, PrivacyStatement entries defined in metadata`	*Metadata values used to describe the SP to users and IdPs, required for clarity and GDPR compliance.*

4.3.2 – Editing saml20-sp-remote.php

Before editing the `saml20-sp-remote.php` file, it is necessary to complete the Keycloak setup: this includes creating a realm, defining a SAML client (e.g., `inception-client`), and adding the SimpleSAMLphp IdP as a trusted identity provider.

Once this is done, you can retrieve the metadata descriptor for the Keycloak client and use it to populate `saml20-sp-remote.php`. This file allows the SimpleSAMLphp IdP to recognize Keycloak as a valid Service Provider, enabling it to forward SAML assertions securely. In the current configuration, only the Keycloak SAML client is defined in this file.

4.4 – Configuration of Keycloak

This section introduces the configuration process for Keycloak, which plays a central role as the identity broker in the federated authentication workflow. In the setup adopted for the ILC4CLARIN infrastructure, Keycloak acts as an intermediary between the SAML-based proxy—responsible for interacting with external academic Identity Providers—and the final web applications, such as INCEpTION, which require user authentication.

The configuration of Keycloak includes defining a realm dedicated to federated access, registering a SAML Identity Provider (the proxy IdP), and setting up one or more clients that represent the consuming applications. Special attention is given to the handling of user identity attributes, certificate validation, and protocol bindings, to ensure a seamless and secure exchange of authentication data between components.

While Keycloak is highly flexible and supports multiple identity federation scenarios, this section focuses specifically on the minimal and necessary configurations to support a standard SAML 2.0 login flow initiated by applications like INCEpTION. The steps provided aim to guarantee compatibility with the metadata and assertions generated by the SimpleSAMLphp proxy, while also aligning with best practices for secure session handling and user management.

4.4.1 – Realm and client configuration

To configure Keycloak for federated SAML authentication with the INCEpTION platform, the first step is to define a new realm and a SAML client within the Keycloak administration console. Below are the steps to complete this configuration:

1. Create a Realm

1. Access the Keycloak Admin Console (e.g., at `https://<your-domain>/auth/admin/`).
2. From the left-hand menu, click on **Master** (top-left dropdown), then select **Create Realm**.
3. In the **Name** field, enter:
   `inception-demo`
4. Click **Create** to finalize the new realm.

➔ This realm will isolate the configuration and management of all authentication entities related to the INCEpTION deployment.

2. Create a SAML Client

Once the realm is created, you need to define a SAML client to represent INCEpTION.

1. Make sure you are in the `inception-demo` realm.
2. From the left-hand menu, go to **Clients** and click **Create**.
3. Fill out the form with the following information:
   - **Client ID**:
     `https://vocabs.ilc4clarin.ilc.cnr.it/inception/saml2/service-provider-metadata/inception-client`

- **Client Protocol**:
  `saml`
- Click **Save**.

4. After saving, you'll be redirected to the client configuration screen. Now update the following parameters:
   - **Client Signature Required**:
     Set this to OFF to indicate that INCEpTION will not sign its SAML authentication requests.
   - **Valid Redirect URIs**:
     Add the following URI:
     `https://vocabs.ilc4clarin.ilc.cnr.it/inception/login/saml2/sso/*`
     This ensures that Keycloak can redirect users back to INCEpTION after successful authentication.
5. Click **Save** again to apply all changes.

4.4.2 – Add a SAML Identity Provider in Keycloak

To complete the SAML-based integration of Keycloak with the federated proxy and the INCEpTION platform, an external Identity Provider must be registered within the Keycloak realm. This step ensures that Keycloak can delegate authentication to the upstream SAML proxy (based on SimpleSAMLphp) and process the assertions received.

The configuration involves adding a new SAML-based Identity Provider within the previously created realm (`inception-demo`). The following steps outline how to perform this operation in the Keycloak Admin Console.

1. Access the Keycloak Admin Console and ensure you're operating within the **inception-demo** realm.
2. In the left-hand navigation menu, go to **Identity Providers**.
3. Click the dropdown labeled **Add provider...** and select **SAML**.
4. Fill in the form with the following values:
   - **Redirect URI**:

     https://vocabs.ilc4clarin.ilc.cnr.it/auth/realms/inception-demo/broker/saml/endpoint

     → This is the URI that will be used by the Identity Provider (the SimpleSAMLphp proxy) to redirect back to Keycloak after authentication.
   - **Alias**:
     You can use a clear and descriptive alias, such as:

saml

- **Service Provider Entity ID**:

  https://vocabs.ilc4clarin.ilc.cnr.it/inception/saml2/service-provider-metadata/inception-client

  → This is the Entity ID that Keycloak uses to identify itself to the upstream SAML IdP.

- **Single Sign-On Service URL**:

  https://sp-dev.ilc4clarin.ilc.cnr.it/saml2/idp/SSOService.php

  → This is the SSO endpoint exposed by the SimpleSAMLphp proxy acting as the Identity Provider.

- **Single Logout Service URL**:

  https://sp-dev.ilc4clarin.ilc.cnr.it/saml2/idp/SingleLogoutService.php

5. Click **Save** to register the Identity Provider.

4.4.3 – Configuring Attribute Mappers in Keycloak

To complete the integration between Keycloak and the SimpleSAMLphp proxy acting as a SAML Identity Provider, it's essential to configure attribute mappers within Keycloak. These mappers ensure that the SAML assertions received from the upstream IdP are properly interpreted and translated into Keycloak's internal user model. Without these mappings, attributes such as username, email, or affiliation might not be available to the downstream application (e.g., INCEpTION).

The following SAML mappers have been configured within Keycloak to properly import and map identity attributes from the upstream Identity Provider (IdP) into Keycloak's internal user model:

Mapper Name	SAML Attribute	Category	Purpose
`sn`	**sn (surname)**	*Attribute Importer*	*Imports the user's surname (last name) into the user profile.*
`OriginalIDP`	**OriginalIDP**	*Attribute Importer*	*Stores the ID of the original IdP that issued the SAML assertion.*
`eduPersonPrincipalName`	**eduPersonPrincipalName**	*Attribute Importer*	*Sets a persistent, globally unique user identifier (often used as username).*

`mail`	**mail**	*Attribute Importer*	*Imports the user's email address into the Keycloak user profile.*
`name`	**givenName or displayName**	*Attribute Importer*	*Sets the user's first name or full name, depending on the upstream IdP.*

These mappers ensure that once a SAML assertion is received, Keycloak can reconstruct the user identity using standardized attributes. This is essential for downstream applications such as INCEpTION to receive consistent and meaningful user information. The selected attributes are aligned with CLARIN SPF federation recommendations and are commonly supported by academic and research institutions' IdPs.

4.4.3.1 – Configure sn Mapper

To create this mapper:

1. Go to **Identity Providers > saml > Mappers**.
2. Click on **Create**.

Then fill in the fields as follows:

Field	Value	Description
Name	`sn`	*The identifier for this mapper. Describes the attribute being mapped.*
Sync Mode Override	`inherit`	*Default behavior (can be overridden per client).*
Mapper Type	`Attribute Importer`	*Indicates that this mapper imports an attribute from the SAML assertion.*
Attribute Name	`urn:oid:2.5.4.4`	*The formal OID for the SAML attribute "sn" (surname/last name).*
Friendly Name	`sn`	*A human-readable alias for the attribute.*
User Attribute Name	`lastName`	*The internal Keycloak user model field where the surname will be stored.*

After filling in all fields, click Save to register the mapper.

4.4.3.2 – Configure OriginalIDP Mapper

Here's how the mapper is configured:

Field	Value	Description
Name	`OriginalIDP`	*Identifies the mapper; matches the attribute to be extracted.*
Sync Mode Override	`force`	*Ensures that the attribute is always synchronized on login.*
Mapper Type	`Attribute Importer`	*Specifies that this is importing a SAML attribute.*
Attribute Name	`OriginalIDP`	*Name of the attribute as received in the SAML assertion.*
Friendly Name	`OriginalIDP`	*Human-readable name for the attribute.*
User Attribute Name	`OriginalIDP`	*The internal Keycloak user model field where this attribute will be stored.*

This mapping is especially useful to keep track of the identity provider that issued the SAML assertion, which can be critical in federated contexts where multiple institutions authenticate users.

4.4.3.3 – Configure eduPersonPrincipalName Mapper

Field	Value	Description
Name	`eduPersonPrincipalName`	*Identifier for the mapper configuration.*
Sync Mode Override	`inherit`	*Inherits the default synchronization mode defined for the Identity Provider.*
Mapper Type	`Attribute Importer`	*Indicates this is a SAML attribute being imported into Keycloak.*
Attribute Name	`urn:oid:1.3.6.1.4.1.5923.1.1.1.6`	*The formal OID identifier for the eduPersonPrincipalName attribute.*
Friendly Name	`eduPersonPrincipalName`	*A human-readable label for the attribute.*
User Attribute Name	`eduPersonPrincipalName`	*The name used within Keycloak's internal user model to store the attribute.*

This mapper ensures that the eduPersonPrincipalName, a standard and persistent user identifier across academic institutions, is correctly captured and made available in the Keycloak user session. It's essential for recognizing users consistently across federated services.

4.4.3.4 – Configure mail Mapper

Field	Value	Description
Name	`mail`	*Name identifying the mapper.*
Sync Mode Override	`inherit`	*Inherits the synchronization mode from the parent Identity Provider.*
Mapper Type	`Attribute Importer`	*Indicates that this mapper imports a SAML attribute.*
Attribute Name	`urn:oid:0.9.2342.19200300.100.1.3`	*The OID corresponding to the standard email attribute.*
Friendly Name	`mail`	*A readable label for the attribute.*
User Attribute Name	`email`	*The internal name used in Keycloak's user model to store the email value.*

This mapper ensures that the user's email address, as defined in the federated SAML assertion, is properly stored and recognized by Keycloak. It plays a key role in user identification, communication, and session handling within INCEpTION and other integrated services.

4.4.3.5 – Configure name Mapper

Field	Value	Description
Name	`name`	*Internal identifier for the mapper.*
Sync Mode Override	`inherit`	*Uses the synchronization strategy defined by the parent IdP configuration.*
Mapper Type	`Attribute Importer`	*Indicates this mapper will import a SAML attribute.*
Attribute Name	`urn:oid:2.5.4.42`	*The OID for the givenName SAML attribute.*
Friendly Name	`givenName`	*Human-readable name for the SAML attribute.*

User Attribute Name	`firstName`	*The field in the Keycloak user model where the given name will be stored.*

4.4.3 – Export Broker Metadata

To enable SimpleSAMLphp to correctly forward authentication assertions to the Keycloak broker, the metadata describing the broker as a SAML Service Provider (SP) must be exported and integrated into the proxy configuration.

The metadata can be retrieved directly from the Keycloak endpoint:

```
https://vocabs.ilc4clarin.ilc.cnr.it/auth/realms/inception-demo/broker/saml/endpoint/descriptor
```

This URL returns a valid XML representation of the Keycloak Broker configuration (see Appendix A4). However, SimpleSAMLphp does not consume SAML metadata in XML format directly. Instead, this information must be manually transformed into a PHP associative array and inserted into the saml20-sp-remote.php configuration file.

Below is the correct conversion of the XML metadata into a format usable by SimpleSAMLphp. This configuration describes Keycloak as a remote SP:

```
'https://vocabs.ilc4clarin.ilc.cnr.it/inception/saml2/service-provider-metadata/inception-client' =>
array(
    'metadata-set' => 'saml20-sp-remote',
    'entityid' => 'https://vocabs.ilc4clarin.ilc.cnr.it/inception/saml2/service-provider-
metadata/inception-client',
    'AssertionConsumerService' => array(
        array(
            'Binding' => 'urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST',
            'Location' => 'https://vocabs.ilc4clarin.ilc.cnr.it/auth/realms/inception-
demo/broker/saml/endpoint',
            'index' => 1,
        ),
    ),
),
```

The **Location** must point to the actual Keycloak broker endpoint, **not** the entityID. This distinction is critical to ensure correct SAML assertion delivery.

Only one remote SP is typically needed in this setup—i.e., the Keycloak client representing INCEpTION—so the file `saml20-sp-remote.php` should contain only this block unless other services are also brokered.

Optionally, a script may be developed to extract and convert XML descriptors into PHP configuration, although for a single SP this manual setup is sufficient.

This setup completes the trust bridge between the SimpleSAMLphp proxy (acting as IdP) and the Keycloak broker (acting as SP), enabling successful SSO authentication from institutional IdPs to INCEpTION.

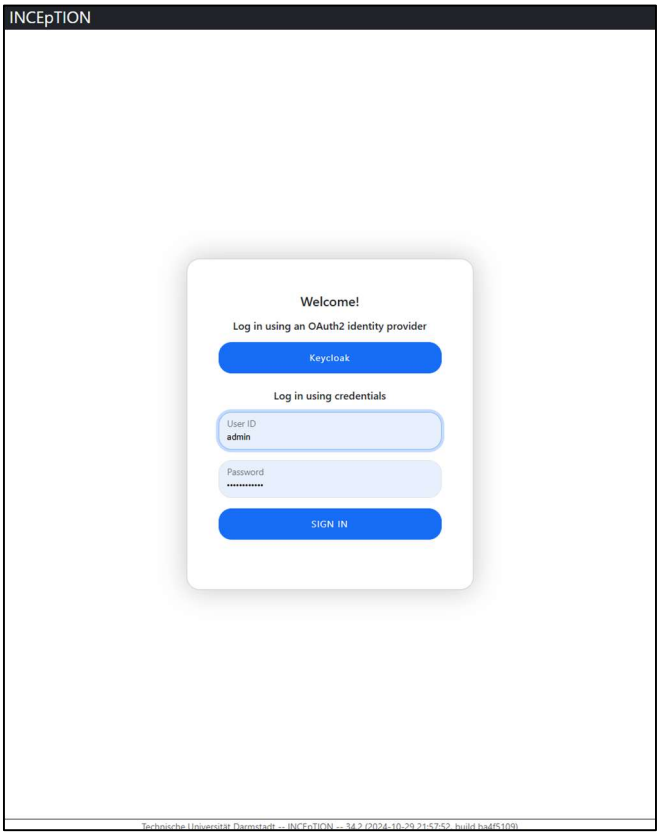
5 – Testing the authentication system

Once the configuration of all components—INCEpTION, Keycloak, and the SimpleSAMLphp proxy—has been completed, it is essential to verify that the federated authentication flow works as intended. This section focuses on the testing procedures designed to validate the full SSO pipeline, ensuring that each layer properly interacts with the next, from the initial user login to the final authenticated session within INCEpTION.

The tests will cover the discovery and selection of institutional Identity Providers, the transmission and validation of SAML assertions, the processing of identity attributes by Keycloak, and the final login confirmation by the INCEpTION platform. Both successful login flows and common misconfiguration scenarios will be analyzed to support troubleshooting.

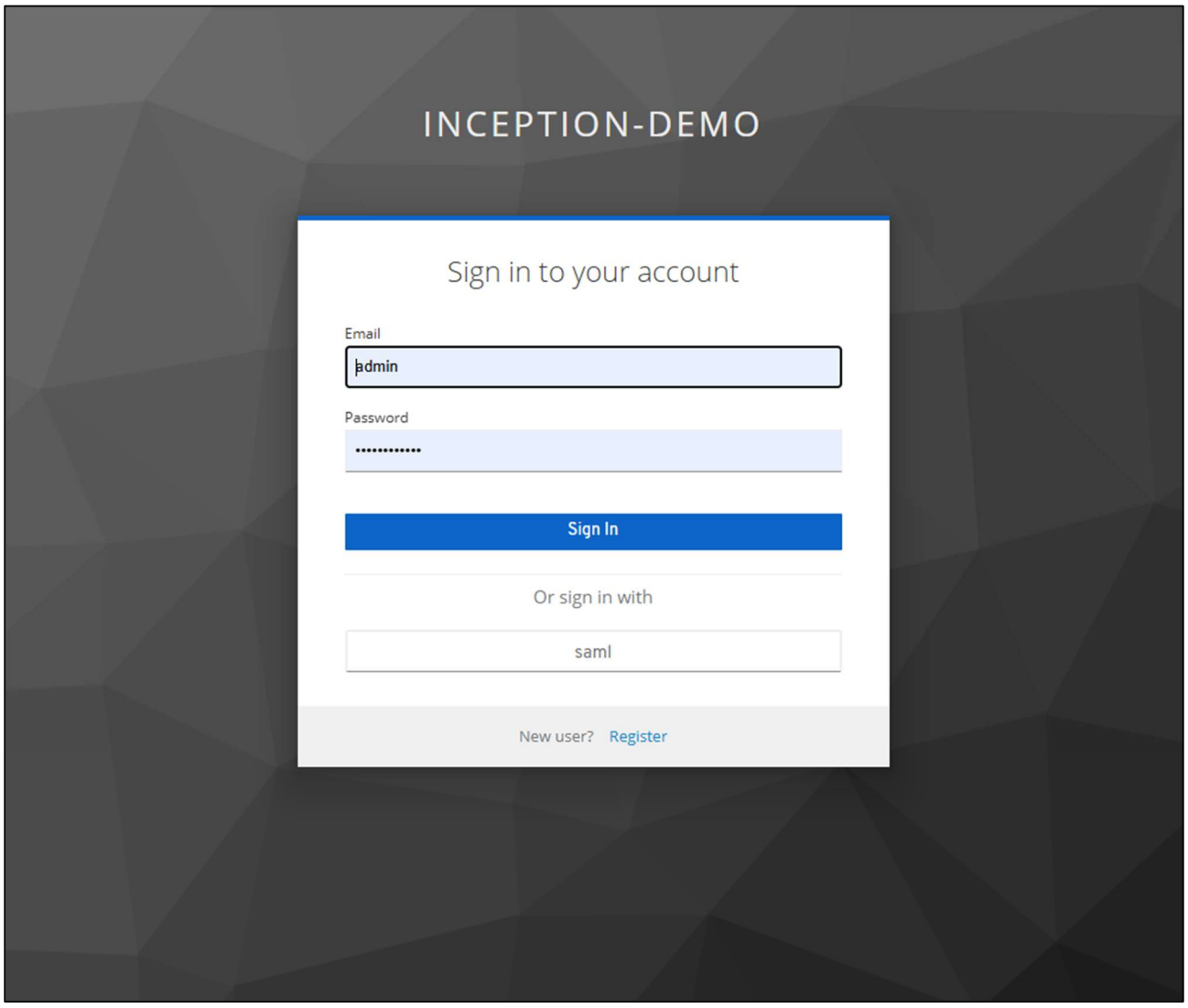
5.1 – Connect to the INCEpTION instance

The first step in testing the authentication system is to connect to the INCEpTION instance deployed at the following address: https://vocabs.ilc4clarin.ilc.cnr.it/inception/. This web-based interface serves as the entry point for accessing the platform and initiating the login process through the configured federated identity infrastructure. You should be able to see a screen like this:



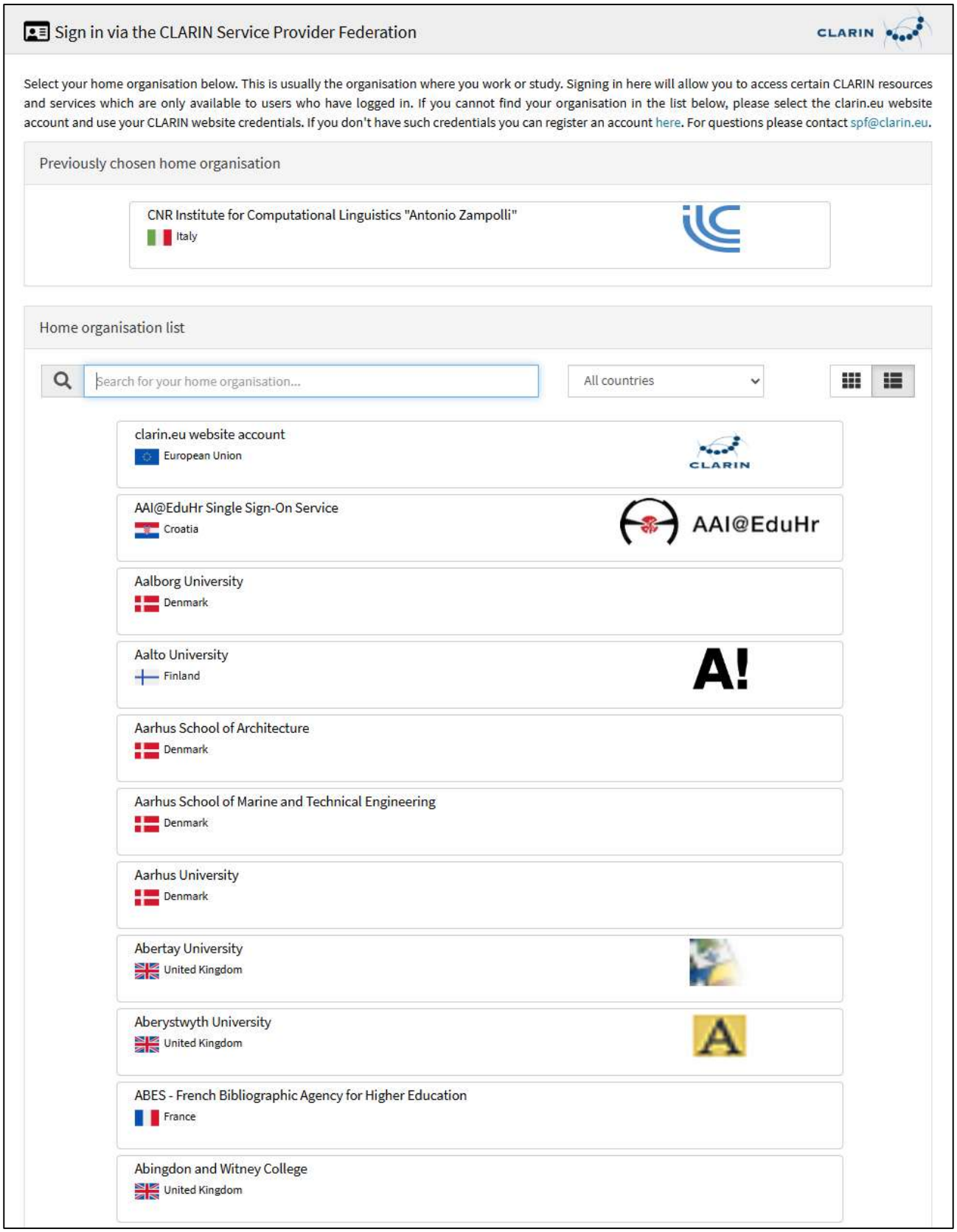
5.2 – Click on “Keycloak” button

To start the authentication flow, it is necessary to click on the “Keycloak” button in order to be redirected to the keycloak instance. You should then be able to see an image like this:



5.3 – Click on “saml” button

Having arrived at this point, it is necessary to click on the “saml” button so as to be redirected to the clarin discovery service page. You should then see a page like this:



On this page you will find a complete list of all institutions belonging to the CLARIN federation.

5.4 – Select your Identity Provider

At this point you must select your Identity Provider of choice. You will then be redirected to your Idp page and will need to enter your institute credentials. In our case, the identity provider page will look like this:

5.5 – Authorize from your IdP

After entering the credentials, it will be necessary to give permission for the passing of the IdP membership information to our service provider, so that this information is then passed to keycloak through the proxy to end up in the application. In our case we will need to press “I agree.”

Nome

Michele Mallia

ID personale

[redacted]

Affiliazione

staff@ilc.cnr.it

faculty@ilc.cnr.it

member@ilc.cnr.it

Identificatore opaco diverso per ogni servizio eduPersonTargetedID

[redacted]

Nome

Michele

E-mail

[redacted]

Cognome

Mallia

Se procedi le informazioni sopra riportate saranno trasmesse al servizio. Acconsenti a rilasciare queste informazioni al servizio ogni volta che accedi?

Seleziona la durata del consenso al rilascio informazioni:

◯ Chiedimelo di nuovo al prossimo accesso

Acconsento solo per questa volta all'invio delle mie informazioni.

◉ Chiedimelo di nuovo se le informazioni da fornire a questo servizio cambiano

Per il futuro acconsento ad inviare automaticamente le stesse informazioni al servizio.

◯ Non chiedermelo di nuovo

Acconsento a rilasciare **tutte** le mie informazioni a **qualunque** servizio.

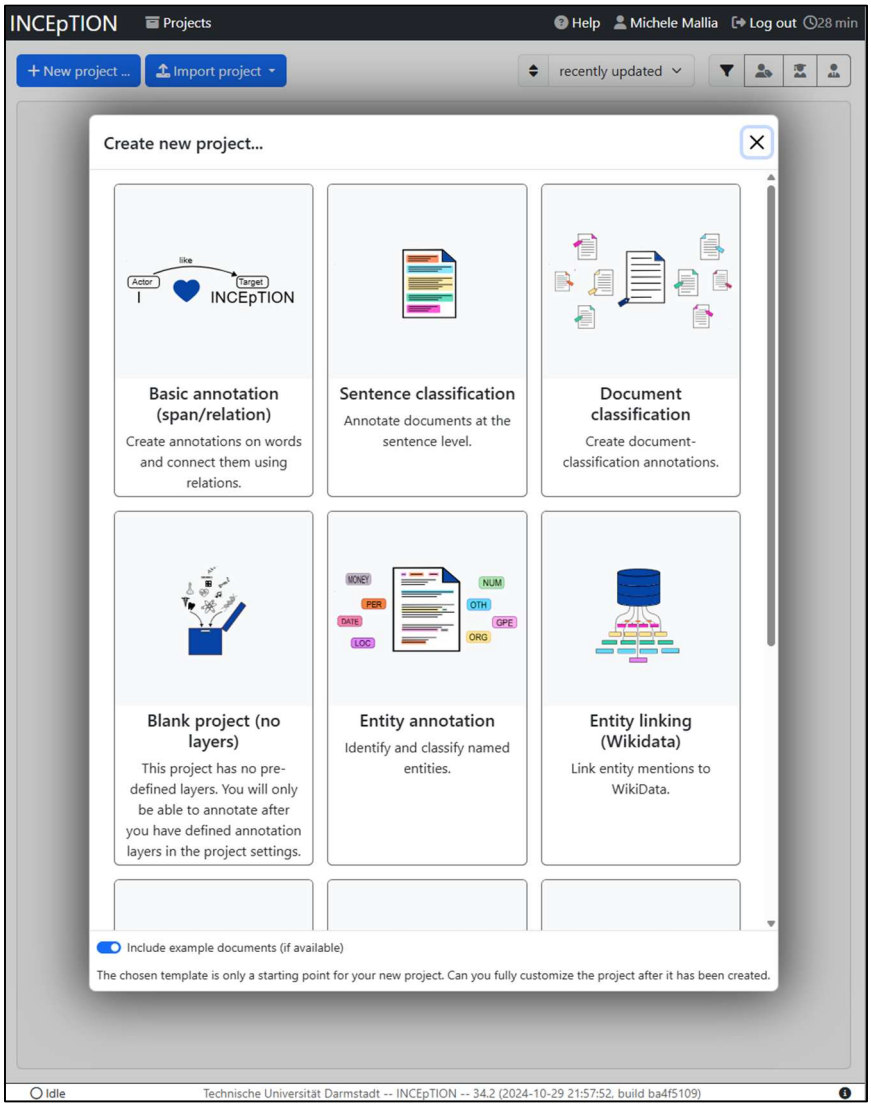
Accetto **Rifiuto**

Questa impostazione può essere revocata in qualsiasi momento tramite la casella di controllo della pagina di accesso.

Istituto di Linguistica Computazionale "Antonio Zampolli" - CNR
Area della Ricerca del CNR di Pisa
Via Giuseppe Moruzzi N° 1, 56124 Pisa, Tel. 050 315 8379

5.6 – Landing to the INCEpTION page

If the authentication process is successful, then at that point your request will be forwarded to INCEpTION which will automatically create a working environment as in this image:



At this point the process is complete and you can begin to semantically annotate texts.

6 – Conclusions and future developments

From a practical standpoint, the adoption of this federated authentication architecture brings substantial and multi-faceted benefits to research infrastructures, particularly in the areas of maintainability, institutional autonomy, and improved service delivery. One of the most immediate advantages lies in the enhanced maintainability of the authentication system. By relying on a modular and standards-compliant architecture—composed of the SimpleSAMLphp proxy, the Keycloak broker, and dedicated application connectors—the overall system is logically segmented into distinct and manageable components. This separation of concerns allows updates, debugging, security audits, and enhancements to be carried out more efficiently and with minimal impact on the operational continuity of the infrastructure.

Moreover, the architecture enables the hosting institution to function as a fully autonomous Service Provider (SP) within the broader federated environment. This autonomy makes it possible to onboard and configure new applications that support SSO without needing to coordinate changes with external authentication providers or modify federation-wide infrastructure. As a result, the institution gains greater control and flexibility in expanding its service offerings, streamlining internal workflows, and meeting the evolving needs of researchers and users.

Although the underlying system architecture may appear complex—incorporating layers such as Keycloak as an Identity Broker and SimpleSAMLphp as a federated SAML proxy—this design is crucial to overcoming a fundamental usability challenge. In a non-brokered environment, each application would have to manually implement and display a separate login button for each institutional Identity Provider, resulting in a fragmented and confusing user interface. Instead, by adopting a 1-to-N architecture, the application perceives a single Identity Provider interface (Keycloak), which then delegates authentication requests to the SAML proxy. This proxy, in turn, is capable of routing these requests to any federated Identity Provider registered in the CLARIN SPF or IDEM GARR federations. The result is a cleaner, unified user experience and significantly improved scalability for both existing and future services.

Another key practical implication of this system is its potential to foster broader engagement from software developers across the research ecosystem. Once the foundational identity architecture is in place, developers can plug into the shared authentication infrastructure without having to implement federation-specific logic or manage metadata manually. This promotes standardization and reusability across projects, while ensuring alignment with federation security policies and best practices.

Appendix

This appendix presents a curated collection of configuration excerpts used in the implementation of the federated Single Sign-On (SSO) infrastructure described in this document. These snippets are intended to provide the reader with concrete examples of how the various components—INCEpTION, Keycloak, and the SimpleSAMLphp-based proxy—have been customized to work together in a standards-compliant authentication flow based on the SAML 2.0 protocol.

While complete configurations are maintained in version-controlled repositories, this appendix highlights only the most relevant and instructive segments, grouped by component.

A1 – INCEpTION (docker-compose.yml)

```
version: '2.4'

networks:
  inception-net:

services:
  db:
    image: "mariadb:10.7"
    environment:
      - MYSQL_RANDOM_ROOT_PASSWORD=yes
      - MYSQL_DATABASE=inception
      - MYSQL_USER=${DBUSER:-inception}
      - MYSQL_PORT=3306
      - MYSQL_PASSWORD=${DBPASSWORD:-inception}
    volumes:
      - ${INCEPTION_DB_HOME:-db-data}:/var/lib/mysql
    command: ["--character-set-server=utf8mb4", "--collation-server=utf8mb4_bin"]
    healthcheck:
      test: ["CMD", "mysqladmin" ,"ping", "-h", "localhost", "-p${DBPASSWORD:-inception}", "-
u${DBUSER:-inception}"]
      interval: 20s
      timeout: 10s
      retries: 10
    networks:
      inception-net:

  app:
    image: "${INCEPTION_IMAGE:-ghcr.io/inception-project/inception}:${INCEPTION_VERSION:-34.2}"
    ports:
      - "${INCEPTION_PORT:-8085}:8080"
    environment:
      - INCEPTION_DB_DRIVER=org.mariadb.jdbc.Driver
      -
INCEPTION_DB_URL=jdbc:mariadb://db:3306/inception?useSSL=false&useUnicode=true&characterEncoding=UTF-8
      - INCEPTION_DB_USERNAME=${DBUSER:-inception}
      - INCEPTION_DB_PASSWORD=${DBPASSWORD:-inception}
      - JAVA_OPTS=-Dspring.jpa.properties.hibernate.dialect.storage_engine=innodb
    volumes:
      - ${INCEPTION_HOME:-app-data}:/export
    depends_on:
      db:
        condition: service_healthy
    restart: unless-stopped
    networks:
```

```
      inception-net:

volumes:
  app-data:
  db-data:
```

A2 – Keycloak (docker-compose.yml)

```
version: '3.1'

volumes:
  postgres_data:
      driver: local

services:

  postgres:
      image: postgres:11
      container_name: keycloak_db
      volumes:
        - ./keycloak/data/:/var/lib/postgresql/data
      environment:
        POSTGRES_DB: keycloak
        POSTGRES_USER: keycloak
        POSTGRES_PASSWORD: password
      ports:
        - "18056:5432"
      restart: always

  keycloak:
    image: quay.io/keycloak/keycloak:14.0.0
    container_name: keycloak
    environment:
      DB_VENDOR: POSTGRES
      DB_ADDR: postgres
      DB_DATABASE: keycloak
      DB_USER: keycloak
      DB_SCHEMA: public
      DB_PASSWORD: password
      KEYCLOAK_USER: admin
      KEYCLOAK_PASSWORD: admin
      PROXY_ADDRESS_FORWARDING: "true"

    volumes:
      - ./conf/standalone.xml:/opt/jboss/keycloak/standalone/configuration/standalone.xml
      - ./conf/standalone-ha.xml:/opt/jboss/keycloak/standalone/configuration/standalone-ha.xml
    ports:
      - "8180:8080"
    restart: always
    depends_on:
      - postgres
```

A3 – SimpleSAMLphp Proxy (docker-compose.yml)

```yaml
version: '2'
services:

  sp.ilc4clarin.ilc.cnr.it:
    build: build
    container_name: sp.ilc4clarin.ilc.cnr.it
    hostname: proxy
    volumes:
      - ./simplesamlphp:/code
      - ./proxy:/conf
      - /etc/letsencrypt:/etc/letsencrypt
    working_dir: /code
    environment:
      - SIMPLESAMLPHP_CONFIG_DIR=/conf/
    command: apache2 -D FOREGROUND
    entrypoint: ["/bin/sh", "-c", "exec apache2-foreground"]

  nginx:
    image: nginx:stable
    container_name: nginx_sso
    links:
      - sp.ilc4clarin.ilc.cnr.it
    volumes:
      - ./nginx:/etc/nginx:ro
    ports:
      - '84:80'
      - '2443:443'
```

A4 – Keycloak Broker Metadata (XML)

```xml
<md:EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
    xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
entityID="https://vocabs.ilc4clarin.ilc.cnr.it/inception/saml2/service-provider-metadata/inception-
client" ID="ID_8f76766c-69e2-4e91-b615-0446544a642e">
    <md:SPSSODescriptor protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol"
AuthnRequestsSigned="false" WantAssertionsSigned="false">
        <md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
Location="https://vocabs.ilc4clarin.ilc.cnr.it/auth/realms/inception-demo/broker/saml/endpoint"/>
        <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified</md:NameIDFormat>
        <md:AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
Location="https://vocabs.ilc4clarin.ilc.cnr.it/auth/realms/inception-demo/broker/saml/endpoint"
isDefault="true" index="1"/>
    </md:SPSSODescriptor>
</md:EntityDescriptor>
```

References

De Castilho, R. E., Klie, J.-C., Kumar, N., Boullosa, B., & Gurevych, I. (2018). Linking Text and Knowledge Using the INCEpTION Annotation Platform. *IEEE 14th International Conference on e-Science*, (p. 327-328).

Gavriilidou, M., Piperidis, S., Galanis, D., Pouli, K., Bakagianni, J., Tsiouli, I., . . . Gkirtzou, K. (2024). The CLARIN:EL infrastructure: Platform, Portal, K-Centre. *Linköping Electronic Conference Proceedings*.