Installing and configuring a Proxy with SimpleSAMLphp on ILC4CLARIN Infrastructure for SSO with CLARIN SPF

author: Michele Mallia

mail: michele.mallia@ilc.cnr.it | michele.mallia@cnr.it

Pisa : 05/2025

Summary

1. Introduction

This document provides a comprehensive technical guide for the installation, configuration, federation, and deployment of a **Service Provider (SP)** for federated authentication within the **ILC4CLARIN**[1] infrastructure. The primary goal is to equip technical personnel with the necessary knowledge and instructions to successfully implement this authentication solution. Given its complexity, this process requires expertise in multiple technical domains and can be particularly time-consuming for those unfamiliar with the underlying infrastructure that supports these technologies.

To facilitate this process, the document presents all the essential technical details for installing and configuring the application. Additionally, it offers contextual information about the ILC4CLARIN infrastructure and outlines the procedures for integrating the Service Provider with the **DFN-AAI**[2] authentication federation. Once these foundational steps are completed, deployment can proceed with a set of best practices and recommendations to ensure the secure and efficient operation of the service.

1.1 Overview of SimpleSAMLphp

SimpleSAMLphp[3] is an open-source application designed to facilitate authentication and identity federation through the **Security Assertion Markup Language (SAML) protocol**[4]. It serves as both an **Identity Provider (IdP)**[5] and a **Service Provider (SP)**[6], allowing seamless integration with various authentication systems. Its flexibility and modular architecture make it a widely adopted solution for managing authentication in distributed infrastructures.

At its core, SimpleSAMLphp acts as an intermediary between web applications and identity providers. As a Service Provider, it enables applications to delegate authentication to external identity federations, reducing the need for local user management. Conversely, as an Identity Provider, it allows an organization to centralize authentication and provide credentials to multiple connected services.

Configuration in SimpleSAMLphp is structured around metadata files and configuration scripts. These define how authentication requests and responses are processed, which identity providers are trusted, and what attributes should be exchanged. This metadata-driven approach ensures compatibility with a wide range of SAML-based authentication systems, including academic and research federations such as **CLARIN**[7], DFN-AAI and **eduGAIN**[8].

---

[1] https://ilc4clarin.ilc.cnr.it/
[2] https://www.aai.dfn.de/
[3] https://simplesamlphp.org/
[4] https://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html
[5] https://simplesamlphp.org/samlidp/
[6] https://simplesamlphp.org/samlsp/
[7] https://www.clarin.eu/
[8] https://edugain.org/

1.2 Overview of Federated Authentication

Federated authentication is a mechanism that allows users to access multiple services and applications using a single set of credentials, without the need to create and manage separate accounts for each platform. This approach enhances security, improves user experience, and simplifies identity management by leveraging existing identity providers (IdPs) to authenticate users across different organizations and infrastructures.

At its core, federated authentication operates through trust relationships established between identity providers and service providers (SPs). When a user attempts to access a federated service, the SP redirects the authentication request to the designated IdP, which verifies the user's credentials and provides an assertion containing relevant identity attributes. These attributes, such as **name**, **email**, and **roles**, are then used by the SP to grant appropriate access rights. To facilitate authentication, the service provider connects to a **discovery service**[9], which presents users with a list of available identity providers within the **CLARIN federation**. This allows users to select their home institution for authentication, ensuring seamless access without requiring additional account creation.

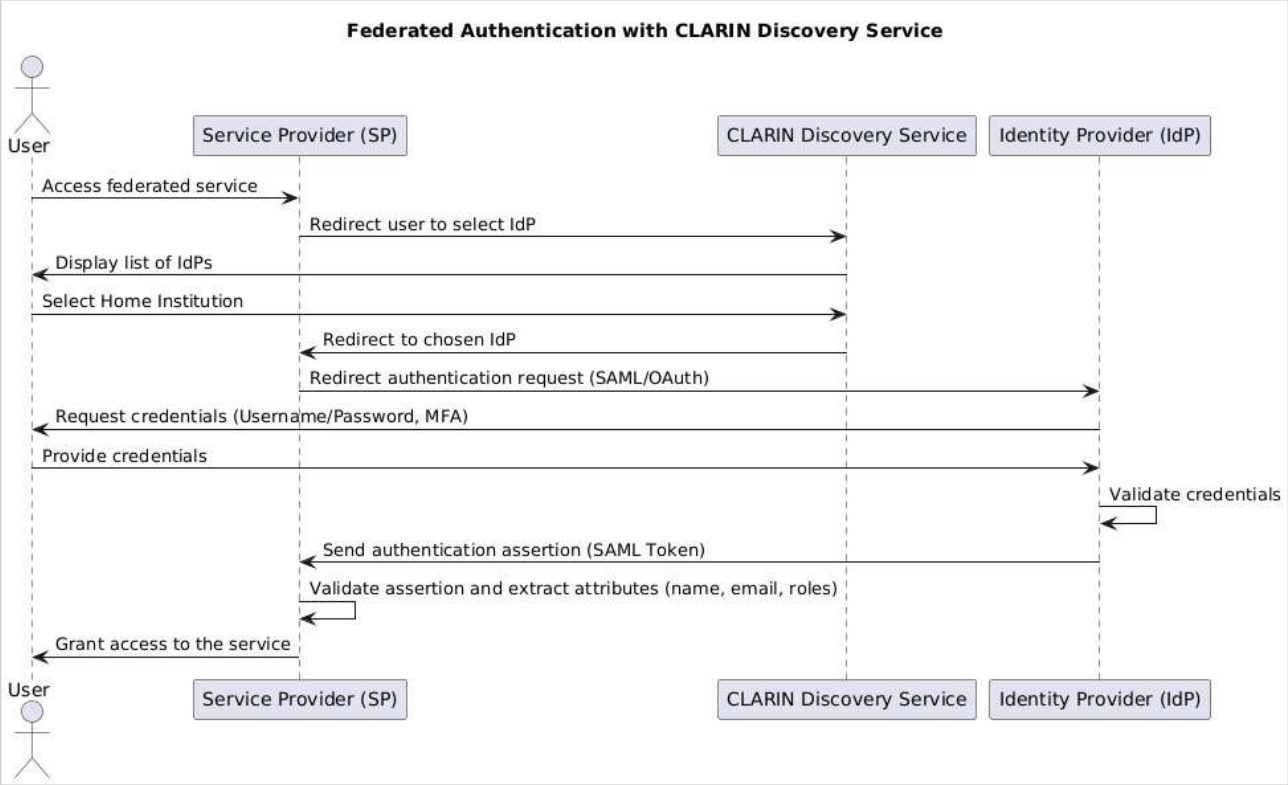


Figure 1: Federated authentication workflow

The Security Assertion Markup Language (SAML) is one of the most widely used protocols for federated authentication. It defines how authentication data is exchanged between an IdP and an SP, ensuring interoperability across different systems. Other protocols, such as OpenID Connect and OAuth 2.0,

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

offer alternative approaches to identity federation, particularly in environments where API-based authentication and authorization are required.

One of the key advantages of federated authentication is its ability to enhance security while reducing administrative overhead. By centralizing identity management within trusted identity providers, organizations can enforce consistent security policies, such as multi-factor authentication (MFA), password complexity rules, and session timeouts. Additionally, users benefit from a seamless login experience, avoiding the burden of remembering multiple passwords or creating redundant accounts.

1.3 Motivation of this work

This work originates from the need to integrate CLARIN's Single Sign-On (SSO) service into a web application for linguistic and semantic annotation, specifically **INCEpTION**[10] (De Castilho, Klie, Kumar, Boullosa, & Gurevych, 2018), developed by **Technische Universitat Darmstadt**[11]. Support for SSO implementation was provided by the **CLARIN-EL**[12] team, who have worked on deploying CLARIN's authentication infrastructure within the CLARIN-EL research environment (Gavriilidou, et al., 2024), and **Richard Eckart De Castilho**[13], a developer of INCEpTION's team. The solution adopted involves the use of **Keycloak**[14], an identity and access management system, to mediate authentication requests.

From a technical perspective, it is not feasible to directly connect a web application acting as a Service Provider (even if equipped with an integrated SAML module) to CLARIN's discovery service. Additionally, presenting an extensive list of Identity Providers (IdPs) as a graphical selection of buttons would be impractical. As you can see in the Figure 2 (next page), the only viable approach is to implement a **proxy-based authentication stack** that consolidates multiple IdPs into a **single authentication endpoint**, effectively acting as a bridge between the Service Provider and the federation's discovery service. This stack, composed of both **IdP and SP components**, enables seamless interaction with multiple identity providers while maintaining a streamlined authentication process.

[10] https://inception-project.github.io/
[11] https://www.tu-darmstadt.de/
[12] https://www.clarin.gr/en
[13] https://www.informatik.tu-darmstadt.de/ukp/ukp_home/staff_ukp/ukp_home_content_staff_1_details_42176.en.jsp
[14] https://www.keycloak.org/

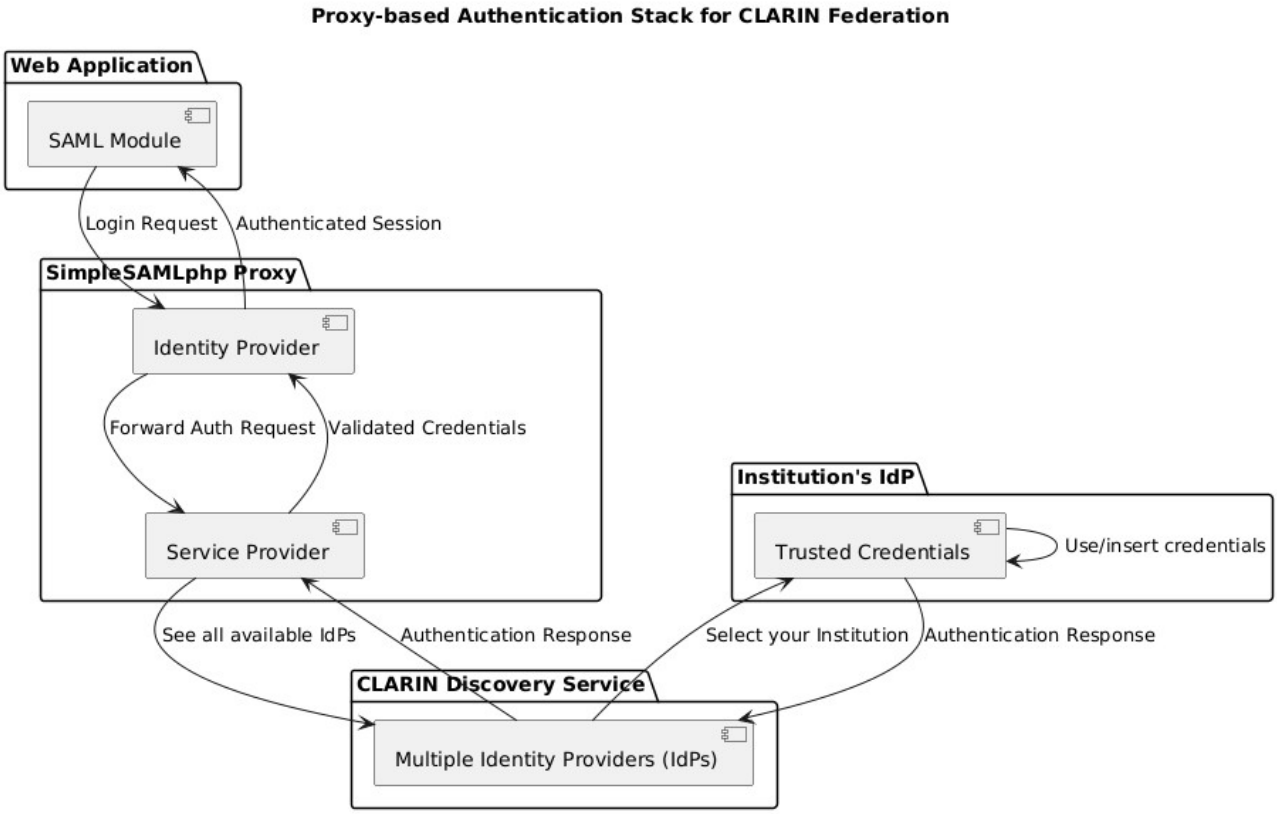


Figure 2: architectural overview of the proxy

A more detailed explanation of this architecture and its implementation will be provided in a **separate technical report** that follows this document. The present report, instead, focuses specifically on the **procedures for setting up the Service Provider** itself. In both this and the other technical report, every step will be described in precise detail to ensure that both **technologists and researchers** that works on CLARIN infrastructure have all the necessary information to implement this complex authentication architecture effectively.

1.4 Project context

This technical report is part of the activities carried out within the **H2IOSC (Humanities and Heritage Italian Open Science Cloud) project**[15]. H2IOSC is a national initiative aimed at **developing an open science cloud infrastructure** for the humanities and cultural heritage sectors, providing researchers with **federated access to distributed resources, tools, and services**. The project aligns with

[15] H2IOSC Project - Humanities and cultural Heritage Italian Open Science Cloud funded by the European Union NextGenerationEU - National Recovery and Resilience Plan (NRRP) - Mission 4 "Education and Research" Component 2 "From research to business" Investment 3.1 "Fund for the realization of an integrated system of research and innovation infrastructures" Action 3.1.1 "Creation of new research infrastructures strengthening of existing ones and their networking for Scientific Excellence under Horizon Europe" - Project code IR0000029 - CUP B63C22000730005. Implementing Entity CNR.

European open science principles and aims to integrate existing research infrastructures into a **cohesive and interoperable ecosystem**.

CLARIN-IT[16] is the Italian node of CLARIN ERIC, a European infrastructure dedicated to digital language resources and tools for the humanities and social sciences. One of its primary goals is to provide **seamless and secure access** to linguistic and textual resources for researchers, facilitating interoperability between different institutions.

[16] https://www.clarin-it.it/it

2. Architectural context and system requirements

From the point of view of architectural context, the context in which we find ourselves is as follows: the entire application is hosted at a virtual instance hosted at the **GARR Cloud**[17] data center in Palermo. This instance possesses the following characteristics:

- **Instance Name**: h2iosc_lod_test
- **Operating System**: Ubuntu 22.04.4 LTS
- **Kernel**: Linux 5.15.0-124-generic
- **Architecture**: x86_64
- **CPU**: Intel Xeon Processor (Icelake)
- **Number of processors**: 8
- **RAM**: 16GB
- **Disk Space**: 2TB

In addition to these features, rules for protection groups for network access to the instance should also be listed:

- Allow connections coming from any location to port **80** and **443** (http and https)
- Allow secure connection for **SSH** only from CNR-ILC internal network (port **22**)
- Allow connections to port **9000** from anywhere and/or CNR-ILC internal network

Leaving aside the first two rules (HTTP/S and SSH), the third rule is useful in that it is used to be able to access the **Portainer**[18][19] control panel, a very useful tool regarding the remote management of **Docker**[20] through an intuitive graphical interface. In fact, all services and applications within the instance are managed through Docker containers, which are all independent of each other and are accessed through a reverse proxy managed by **Apache**, which diverts connections to the containers based on the requested routes. Below, in **Figure 3**, you can observe the general architecture of the vocabs server.

[17] https://cloud.garr.it/
[18] https://www.portainer.io/
[19] Installation instructions:
https://docs.portainer.io/start/install-ce/server/docker/linux
[20] Installation instructions:
https://docs.docker.com/engine/install/ubuntu/

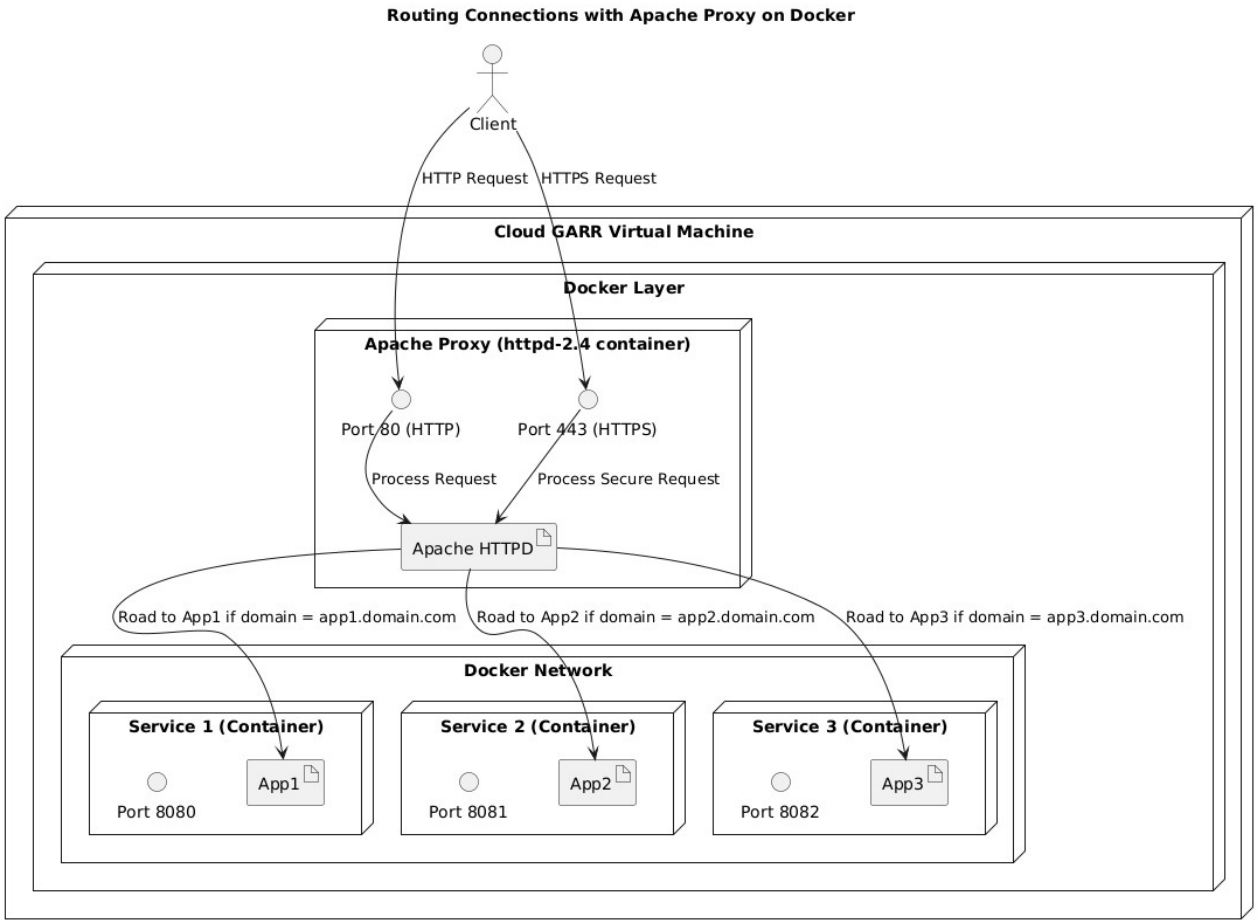


Figure 3: vocabs server architecture

When installing the proxy, this version of Docker (and Docker Compose) was used:

- Docker version 25.0.3, build 4debf41
- Docker Compose version v2.24.6

As for Portainer, these are the version details:

- Portainer Community Edition
- Server Version: 2.19.4
- Database Version: 2.19.4
- CI Build Number: 35428
- Image Tag: linux-amd64-2.19.4
- Compilation tools:
    - Nodejs v18.19.0
    - Yarn v1.22.21
    - Webpack v5.88.1
    - Go v1.20.5

2.1 Installing and configuring Apache Reverse Proxy (httpd:2.4)

The basis of it all is to set up the Apache reverse proxy to have connections diverted to the future simplesamlphp application container. To do this you need to connect to the Portainer instance that is reachable at port 9000 of your server address, in our case the Portainer responds to the following address:

```
http://vocabs.ilc4clarin.ilc.cnr.it:9000
```

To connect to the instance without having errors with SSL protocols, you need to connect in incognito mode and make sure to type 'http' so that by default you set an unsecure connection.[21]

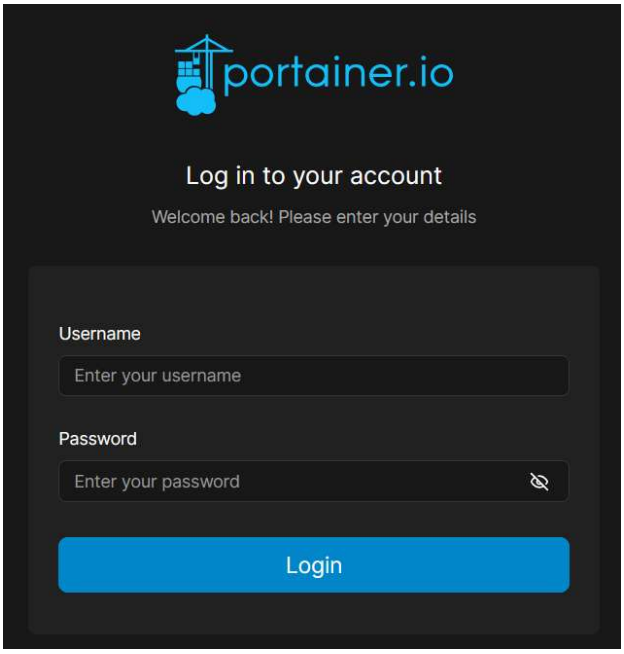


Figure 4: login view of Portainer

Once connected, you must enter credentials to access the control dashboard. After logging in, click on the environment created when installing Portainer (in our case it is named **H2IOSC_LOD_TEST**). Click on "Container" in the menu on the left so you can view all the containers on the machine. Currently, you should only be able to see the Portainer container (which is not selectable or editable).

[21] For adding SSL to Portainer: https://docs.portainer.io/advanced/ssl

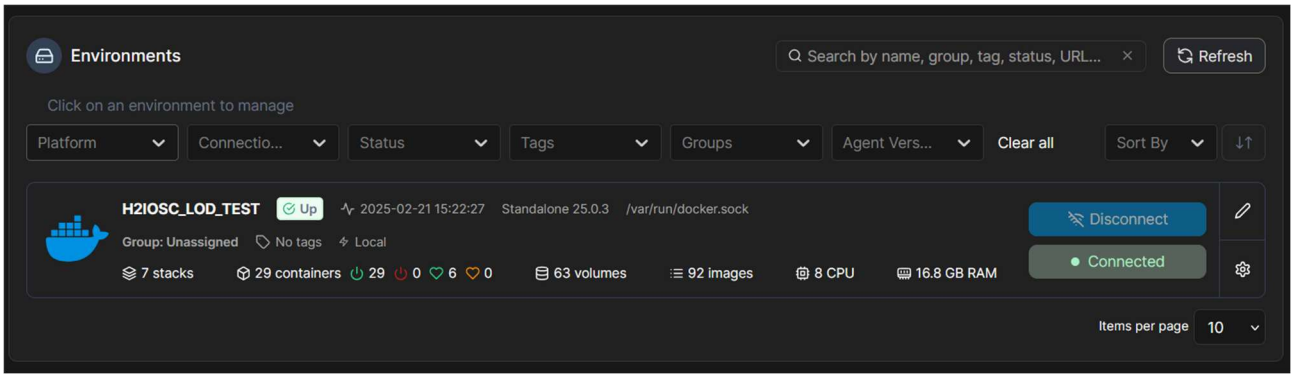


Figure 5: Environment view

Before proceeding with the creation of the Apache container, we need to make sure that two folders are present: the first one must be apache-server[22][23], the second one must be the letsencrypt folder[24].

Click on the "Add container" button and a page will appear where there is all the info for creating a container; then add the following information:

- **Name**: Apache
- **Image**: httpd:2.4
- **Network ports configuration**:
    - **Manual network port publishing**
        - (click on) **Publish a new network port**
            - **Host**: 80 > **Container**: 80
        - (click again on) **Publish a new network port**
            - **Host**: 443 > **Container**: 443
- **Restart policies**: Always
- **Volumes**:

[22] The folder must be located at this path: `/root/apache-server`
[23] If you have some problems with folder binding, **it's suggested to use volumes instead**. In theory, by binding an empty folder from the host to the container, the empty contents are expected to be copied to the target folder on the container. Personally, I don't remember the initial configuration of the Apache server very well. In any case, you have to use always volumes.
Please see the official guides:
https://docs.portainer.io/user/docker/volumes/add
[24] The folder is located under: `/etc/letsencrypt`
Ensure you have letsencrypt installed on your machine. Please see the footnote info at Section 5.1.

- (click on) **Map additional volume**:
    - Select "**Bind**"
        - **Container**: /usr/local/apache2
        - **Host**: /root/apache-server
- (click again on) **Map additional volume**:
    - Select "**Bind**":
        - **Container**: /etc/letsencrypt
        - **Host**: /etc/letsencrypt

- Click on "**Deploy the container**"

Once the container has been deployed you should be able to see this from the Portainer screen or by running the "`docker ps`" command and you should be able to see a container with the name "`Apache`". If the volume creation process went well, you should be able to see inside the folder all the Apache configuration files.

```
root@h2iosc-lod-test:~# docker ps --format
"{{.ID}}\t{{.Names}}\t{{.Image}}\t{{.Ports}}"

25f5e524c1aa    Apache  httpd:2.4       0.0.0.0:80->80/tcp, :::80->80/tcp,
0.0.0.0:443->443/tcp, :::443->443/tcp

b88d949dcf4d    portainer       portainer/portainer-ce:latest   0.0.0.0:8000-
>8000/tcp, :::8000->8000/tcp, 0.0.0.0:9000->9000/tcp, :::9000->9000/tcp, 9443/tcp
```

Figure 6: Output of docker ps command

The Apache folder should look like this:

```
drwxr-xr-x 4 root root 4096 Feb 21 16:51 conf
drwxr-xr-x 2 root root 4096 Feb  5 11:48 logs
drwxr-xr-x 2 root root 4096 Feb  3 17:23 htdocs
drwxr-xr-x 2 root root 4096 Feb 13  2024 build
drwxr-xr-x 2 root root 4096 Feb 13  2024 bin
drwxr-xr-x 2 root root 4096 Feb 13  2024 include
drwxr-xr-x 2 root root 4096 Feb 13  2024 cgi-bin
drwxr-xr-x 3 root root 4096 Feb 13  2024 error
drwxr-xr-x 3 root root 4096 Feb 13  2024 icons
drwxr-xr-x 2 root root 4096 Feb 13  2024 modules
```

The files we are interested in are inside the `/conf` folder. The first file we need to work on is the `httpd.conf` file, which represents the general configuration of Apache (you can see a complete version of this file at Appendix 11). Here are some key points to consider for proper web-server configuration:

2.1.1 Enabling Modules

Here's a list of relevant (highlighted) activated modules[25]:

- mpm_event_module
- authn_file_module
- authn_core_module
- authz_host_module
- authz_groupfile_module
- authz_user_module
- authz_core_module
- access_compat_module
- auth_basic_module
- **socache_shmcb_module**
- reqtimeout_module
- filter_module
- mime_module
- log_config_module
- env_module
- setenvif_module
- **headers_module**
- **proxy_module**
- **proxy_http_module**
- **proxy_wstunnel_module**
- **ssl_module**
- unixd_module
- status_module
- autoindex_module
- dir_module
- alias_module
- **rewrite_module**

[25] By uncommenting the LoadModule lines.

2.1.2 Enable configuration files

The server is initially set to listen on port 80 ("**Listen 80**" rule). Inside the extra folder are the additional configuration files for Apache. In fact, if you notice the "Include" part of the clauses there are several files. You can include as many separate configuration files as there are possible configurations and/or services within the infrastructure. In this case, it was decided to separate the configurations to simplify and better organize everything without creating conflicts or confusion.

The file we are interested in in this case is `h2iosc-sp.conf`, which contains information about our service provider that we are going to install and configure. At the moment, this file can also be empty; we will see later, in the section on certificate configuration (section 5.1), how and what to write to set up proper routing and request handling.

We now turn to an overview of the structure of the proxy configuration in simplesamlphp.

3. Docker SimpleSAMLphp project structure

For the realization of our goals, it is necessary to use a particular type of configuration found on GitHub at the following address, which I leave as a footnote[26]. This type of project is structured in the following way:

```
docker-simplesamlphp
├ build
│ ├ apache2.conf
│ ├ Dockerfile
│ ├ ...
├ ca
│ ├ ...
├ idp
│ ├ metadata
│ │ ├ ...
│ ├ authsources.php
│ ├ config.php
│ ├ saml20-idp-hosted.php
│ ├ server.crt
│ └ server.pem
├ nginx
│ ├ nginx.conf
├ proxy
│ ├ metadata
│ │ ├ ...
│ ├ authsources.php
│ ├ config.php
│ └ saml20-idp-hosted.php
├ simplesamlphp
│ ├ ...
├ sp1
│ ├ metadata
│ │ └ ...
│ ├ authsources.php
│ └ config.php
├ sp2
│ ├ metadata
│ │ └ ...
│ ├ authsources.php
│ └ config.php
├ docker-compose.yml
├ LICENSE
└ README.md
```

This project provides a containerized infrastructure for deploying a SimpleSAMLphp-based authentication proxy. The structure includes multiple components for managing Identity Providers (IdP), Service Providers (SP), an authentication proxy, and related configurations. Below is a breakdown of the key directories and files:

[26] https://github.com/simplesamlphp/docker-simplesamlphp

Root-Level files:

- 📜 **docker-compose.yml** – Docker Compose configuration, orchestrating the different containers (IdP, SPs, Proxy, Nginx, Memcached, Redis, Mysq).
- 📜 **LICENSE –** Licensing information for the project.
- 📜 **README.md –** Documentation explaining setup, usage, and configuration of the system.

Core Directories and Components:

📁 **build** → this directory contains files for building the Docker image, including**:**

- **Dockerfile** – Defines the instructions for building the SimpleSAMLphp container.
- **apache2.conf –** Configuration file for Apache, which serves SimpleSAMLphp.
- Other essential files for initializing and configuring the service.

📁 **ca** → Contains **Certificate Authority (CA) files** required for SSL/TLS encryption.

📁 **idp (Identity Provider Configuration)** → this folder holds the configuration files for the Identity Provider (IdP) component, responsible for authenticating users**:**

- 📁 **metadata/ –** Stores metadata for IdP configurations**.**
- **authsources.php –** Defines authentication sources for SimpleSAMLphp**.**
- **config.php –** General configuration settings**.**
- **saml20-idp-hosted.php –** Metadata about hosted IdP**.**
- **server.crt & server.pem –** SSL certificates for secure communication**.**

📁 **nginx** → Contains the Nginx configuration file (nginx.conf), likely used as a reverse proxy for handling requests between components.

📁 **proxy (IdP + SP Configurations)** → This directory includes configurations for a proxy-based authentication system that acts as an intermediary between multiple identity providers and service providers:

- 📁 **metadata/** → Stores metadata related to IdPs and SPs interacting with the proxy**.**
- **authsources.php –** Defines authentication sources for SimpleSAMLphp**.**
- **config.php –** General configuration settings**.**

- **saml20-idp-hosted.php –** Metadata about hosted IdP**.**

📁 **simplesamlphp** → Holds the SimpleSAMLphp application files necessary for authentication management.

📁 **sp1 &** 📁 **sp2 (Service Provider Configurations)** → Each directory represents a separate Service Provider (SP) that interacts with the authentication system.

- 📁 **metadata/ –** Metadata related to each service provider.
- **authsources.php –** Defines authentication sources for the specific SP.
- **config.php –** Configuration settings for the SP.

⚠ **This initial configuration serves as a starting point. The description of the individual initial components will help to understand how and what to go about modifying to achieve the exact same configuration used in production.**

3.1 Docker compose (original)

This Docker Compose (see Appendix, part 1) configuration defines a federated authentication environment based on SimpleSAMLphp, integrating an Identity Provider (IdP), a Proxy, multiple Service Providers (SPs), and supporting services like MySQL, Redis, Memcached, and Nginx. The setup simulates a federated authentication environment for testing and development.

3.1.1 Identity Provider (IdP)

- **idp.tutorial.stack-dev.cirrusidentity.com**
  - **Builds from the build directory**, which contains the Dockerfile and necessary configurations.
  - **Mounts two volumes:**
    - ./simplesamlphp → Contains the SimpleSAMLphp codebase.
    - ./idp → Stores configuration files for the IdP.
  - **Environment Variable:**

        - SIMPLESAMLPHP_CONFIG_DIR=/conf/ → Sets the directory where SimpleSAMLphp will look for its configurations.
    - **Links to MySQL** → Likely used for session storage or user authentication data.

3.1.2 Authentication Proxy

- **proxy.tutorial.stack-dev.cirrusidentity.com**
    - **Acts as a middle layer** between the IdP and SPs, allowing for federated authentication management.
    - **Same build structure as IdP**, but mounts the ./proxy directory for configuration.
    - **Environment Variable:**
        - SIMPLESAMLPHP_CONFIG_DIR=/conf/
    - **No direct links to MySQL or Redis**, meaning it mainly proxies authentication requests.

3.1.3 Service Providers (SPs)

- **sp1.tutorial.stack-dev.cirrusidentity.com & sp2.tutorial.stack-dev.cirrusidentity.com**
    - Each **acts as a separate Service Provider** in the authentication federation.
    - **Uses SimpleSAMLphp** but with different configurations (./sp1 for sp1 and ./sp2 for sp2).
    - **sp1 links to Redis, while sp2 links to Memcached** for session caching.
    - **Command (apache2 -D FOREGROUND)** → Ensures Apache stays in the foreground for proper container execution.

3.1.4 Supporting Services

- **Memcached & Redis**
    - **memcached & redis are used for caching sessions and authentication data.**
    - SP1 uses **Redis**, while SP2 relies on **Memcached**, showing flexibility in handling session storage.

3.1.5 MySQL

- Stores **session data and user-related authentication information**.
- Uses **MySQL native authentication** and defines a **dedicated database (sessions)**.
- Creates a **database user (dbuser) with specific credentials**.

- **gencert (Certificate Generation)**
  - Uses cfssl/cfssl for generating **SSL/TLS certificates**.
  - Runs a script (./ca/generate.sh) in /work, ensuring all components have proper certificates for secure authentication.

3.1.6 Nginx (Reverse Proxy)

- **Routes incoming HTTP/S requests** to the IdP, Proxy, and SPs.
- **Ensures secure communication via SSL/TLS**.
- **Volumes:**
  - ./nginx:/etc/nginx:ro → Uses a preconfigured **nginx.conf**.
- **Ports:**
  - 80:80 → Exposes HTTP traffic.
  - 443:443 → Handles secure HTTPS connections.

3.2 Dockerfile (original)

This Dockerfile (see Appendix, part 2) defines a PHP 7.3.13-based Apache container with additional extensions and configurations for SimpleSAMLphp and related authentication services. Below is a breakdown of its key components.

3.2.1 Base Image

```
FROM amd64/php:7.3.13-apache
```

- Uses an Apache-integrated PHP 7.3.13 image based on amd64 architecture.
- This provides an Apache HTTP server with PHP pre-installed, making it suitable for hosting web applications like SimpleSAMLphp.

3.2.2 Environment Variables

```
ENV APACHE_DOCUMENT_ROOT /code/www
```

- Sets `/code/www` as the document root for the Apache web server, ensuring that web files are served from this location.

3.2.3 Installing Dependencies

```
RUN apt-get update && apt-get install -y --no-install-recommends \
          curl \
          git \
          libmemcached-dev \
          libpng-dev \
          unzip \
          zlib1g-dev \
          && rm -rf /var/lib/apt/lists/*
```

- **Installs required system packages, including:**
  - **curl & git** → Used for fetching and version-controlling packages.
  - **libmemcached-dev** → Required for the Memcached PHP extension.
  - **libpng-dev** → Needed for GD image processing.
  - **unzip & zlib1g-dev** → Required for handling compressed files.
- **rm -rf /var/lib/apt/lists/*** → Cleans up unnecessary files to reduce the final image size.

3.2.4 Installing PHP Extensions

```
RUN docker-php-ext-install -j5 gd mbstring mysqli pdo pdo_mysql \
          && pecl install memcached redis xdebug \
          && docker-php-ext-enable memcached redis xdebug
```

- **Compiles and installs PHP extensions** using docker-php-ext-install:
  - **gd** → For image manipulation.
  - **mbstring** → Supports multibyte character encoding.
  - **mysqli & pdo_mysql** → Enables MySQL/MariaDB support.

- **Installs additional extensions via PECL:**
  - **memcached** → Improves caching performance.
  - **redis** → Enables Redis-based caching and session storage.
  - **xdebug** → Useful for debugging and profiling PHP applications.

- **Activates these extensions** using docker-php-ext-enable.

3.2.5 Installing Composer

```
RUN curl -sS https://getcomposer.org/installer | php -- --install-dir=/usr/local/bin --
filename=composer
```

- Downloads and installs Composer, a dependency manager for PHP.

3.2.6 Configuring Apache

```
RUN a2dismod mpm_event && a2enmod mpm_prefork
```

- Disables Apache's mpm_event module and enables mpm_prefork to optimize performance for PHP applications.

3.2.7 Copying Configuration Files

```
COPY apache2.conf /etc/apache2/
COPY mpm_prefork.conf /etc/apache2/mod_available/
COPY startup.sh /
```

- **apache2.conf** → Custom Apache configuration.
- **mpm_prefork.conf** → Custom tuning for MPM Prefork, optimizing how Apache handles concurrent requests.
- **startup.sh** → A shell script for initializing the container.

3.2.8 Entry Point and Default Command

```
ENTRYPOINT ["/startup.sh"]
CMD ["php", "-S", "0.0.0.0:8732", "-t", "/code/www"]
```

- **ENTRYPOINT**: Runs the startup.sh script upon container start.
- **CMD**: Starts a PHP built-in development server on port 8732, serving files from /code/www.

This Dockerfile contains outdated information that is not usable for the context in which we operate; it lacks information to account for changes over time, and therefore this file will be completely changed in production.

3.3 NGINX configuration (original)

This Nginx configuration (see Appendix 3) is designed to act as a reverse proxy that routes incoming requests to a backend service running on port 8732. It also enforces HTTPS, ensuring secure communication. Below is a breakdown of each section.

3.3.1 Global Events Block

```
events {}
```

- This section is empty, but it is required for Nginx's event-driven architecture.
- Typically, it contains worker connection settings, but in this case, it is left with default values.

3.3.2 HTTP Configuration

```
http {
   resolver 127.0.0.11;
```

- Defines the HTTP block, which contains all server configurations.
- `resolver 127.0.0.11;` → This DNS resolver is used inside Docker networks. It allows Nginx to dynamically resolve container names instead of requiring static IPs.

3.3.3 HTTP to HTTPS Redirection (First Server Block)

- This server block listens on **port 80 (HTTP)** and redirects all requests to HTTPS using a **301 permanent redirect**.
- `server_name _ default_server;` → This ensures that this block catches all incoming requests, even if no specific hostname is defined.
- `return 301 https://$host$request_uri;` → Redirects requests to HTTPS while maintaining the original host and URI.

3.3.4 HTTPS Reverse Proxy (Second Server Block)

```
server {
   listen 443 ssl;
   ssl_certificate /etc/nginx/cert.pem;
   ssl_certificate_key /etc/nginx/cert-key.pem;
```

- This block listens on port 443 (HTTPS).
- It enables SSL/TLS using:
    - `ssl_certificate /etc/nginx/cert.pem;` → SSL certificate file.
    - `ssl_certificate_key /etc/nginx/cert-key.pem;` → Private key file.

3.3.5 Reverse Proxy Configuration

```
location / {
   # To force re-resolution every time
   set $authengine_upstream "http://$host:8732";
   proxy_pass $authengine_upstream;
   proxy_http_version 1.1;
   proxy_set_header Upgrade $http_upgrade;
   proxy_set_header Connection 'upgrade';
   proxy_set_header Host $host;
   proxy_cache_bypass $http_upgrade;
}
```

- **Proxying Requests to a Backend Service**:
    - This block handles all requests (/) and forwards them to a backend service running on `port 8732`.
    - `set $authengine_upstream "http://$host:8732";` → Dynamically resolves the backend service's hostname ($host) and forwards traffic to port 8732.
    - `proxy_pass $authengine_upstream;` → Sends the request to the upstream service.
- **WebSocket and HTTP/1.1 Support**:
    - `proxy_http_version 1.1;` → Ensures compatibility with WebSockets.
    - `proxy_set_header Upgrade $http_upgrade; proxy_set_header Connection 'upgrade';`
        - These two directives enable WebSocket connections, allowing real-time communication when needed.
- **Header Adjustments**:
    - `proxy_set_header Host $host;` → Ensures the original host is preserved in proxied requests.

- `proxy_cache_bypass $http_upgrade;` → Prevents caching WebSocket requests, ensuring live updates.

3.4 Apache configuration (internal to simplesamlphp - original)

This Apache (see Appendix 4) configuration sets up a web server environment optimized for handling authentication-related services. The service mentioned is part of the internal configuration of simplesamlphp, in fact it is this service that is responsible for rendering the application for SSO management (in the Dockerfile, this file is copied inside the Apache folder and thus stands as the main configuration inside the container). It listens on port 8732 and applies security settings while enabling compression and logging. Below is a detailed breakdown for the principal components:

3.4.1 User and Group Configuration

```
User www-data
Group www-data
```

- Runs Apache as the www-data user and group, following standard permissions for web applications.

3.4.2 Listening Port Configuration

```
Listen 8732
```

- Configures Apache to listen on port 8732, making it the entry point for HTTP requests.

3.4.3 Directory Access Control

```
<Directory /code/www>
          Options Indexes FollowSymLinks
          AllowOverride None
          Require all granted
</Directory>
```

- Enables directory listing (Indexes) and symbolic links (FollowSymLinks).

- Grants full access to `/code/www`, where the web application is located.

3.5 SimpleSAMLphp (proxy) configuration

The way the code and structure of the project is organized, the main configuration and authentication files are located within the "proxy" folder. The items to mention in question are basically two[27]:

- **authsources.php**: is a core configuration file in SimpleSAMLphp that defines authentication sources used by the Service Provider (SP) or Identity Provider (IdP) to manage user authentication.
- **config.php**: is the main configuration file in SimpleSAMLphp, responsible for defining core system settings. It controls aspects such as logging, session management, security policies, and protocol behavior.

The metadata folder contains metadata related to:

- **Internal IdP (→ saml20-idp-hosted.php)**: to manage internal credentials and to divert requests to the internal service provider

- **External IdPs (→ saml20-idp-remote.php)**: contains the metadata of the Identity Providers of the research Institutions/Entities

- **External service providers (→ saml20-sp-remote.php)**: to host the metadata of other service providers (may be other web applications)

3.5.1 Anatomy of proxy/authsources.php (original)

The authsources.php file in SimpleSAMLphp defines authentication sources, specifying how users authenticate within a Service Provider (SP) or Identity Provider (IdP).

The file is structured as a PHP array (`$config`), where each key represents a different authentication source. The two primary authentication sources defined in this file are:

- **admin** → Provides authentication for **SimpleSAMLphp's administration interface**.

[27] Please see Appendix 5 and Appendix 6 to see the original structure of these files.

```
'admin' => array( 'core:AdminPassword', ),
```

- This section **configures administrative authentication** for **SimpleSAMLphp's web interface**.
- Uses **core:AdminPassword**, meaning that an **administrator must log in using a password set in config.php**.

- **default-sp** → Defines a **SAML 2.0 Service Provider (SP)** that connects to an external **Identity Provider (IdP)**.

There are some part of code we have to observe:

```
'default-sp' => array(
        'saml:SP',
```

➔ This defines a SAML 2.0 Service Provider (saml:SP), allowing the system to authenticate users via federated authentication.

```
'entityID' => null,
```

➔ Defines the **unique identifier for this Service Provider (SP)**.

- **If set to null**, an **entity ID is automatically generated** based on the metadata URL.

```
'idp' => 'https://idp.tutorial.stack-dev.cirrusidentity.com/saml2/idp/metadata.php',
```

➔ Specifies the **Identity Provider (IdP) this SP should contact** for authentication.

- In this case, the SP will always use idp.tutorial.stack-dev.cirrusidentity.com for login requests.
- **If unset (null)**, the user would be presented with a list of available IdPs.

```
'discoURL' => null,
```

➔ Defines the **URL of the discovery service**, which helps users choose an IdP.

- Since it's **set to null**, **SimpleSAMLphp's built-in discovery service** will be used instead.

3.5.2 Anatomy of proxy/config.php (original)

The **config.php** file is the **main configuration file** for **SimpleSAMLphp**, defining the system's core settings. It controls **authentication, security, logging, session management, metadata handling, and storage**. It's very long file and we analyze the principal sections.

3.5.2.1 Base URL Path

```
'baseurlpath' => 'https://proxy.tutorial.stack-dev.cirrusidentity.com/',
```

➔ Defines the **base URL** where SimpleSAMLphp is accessible.

3.5.2.2 Directories for Logs, Certificates, and Data

```
'certdir' => 'cert/',
'loggingdir' => 'log/',
'datadir' => 'data/',
'tempdir' => '/tmp/simplesaml',
```

➔ **Specifies filesystem paths** for key resources:

- certdir → SSL/TLS and SAML certificates.
- loggingdir → Where logs are stored.
- datadir → Stores persistent system data.
- tempdir → Directory for temporary files.

3.5.2.3 Contact Information

```
'technicalcontact_name' => 'Administrator',
'technicalcontact_email' => 'na@example.org',
```

➔ Defines the **technical contact details**, displayed in error messages and metadata.

3.5.2.4 Security Configuration

- **Secret Salt**: used for **cryptographic operations**, such as session signing.

```
'secretsalt' => 'a43x8k07rk8798ejy0ap659dvyvh90mw',
```

- **Admin authentication**: defines the **admin password** for accessing the SimpleSAMLphp web interface.

```
'auth.adminpassword' => '1234',
```

3.5.2.5 Logging Configuration

```
'logging.level' => SimpleSAML\Logger::DEBUG,
'logging.handler' => 'errorlog',
'logging.logfile' => 'simplesamlphp.log',
```

- **Logging level**:
  - `ERR` → Errors only.
  - `WARNING` → Warnings and errors.
  - `NOTICE` → Basic statistics.
  - `INFO` → Verbose logs.
  - `DEBUG` → Full debug logs (**not recommended for production**).

- **Logging handler**:
  - `syslog` → Uses the system log.
  - `file` → Logs to a file.
  - `errorlog` → Logs errors in PHP's error log.

3.5.2.6 Session Configuration

```
'session.duration' => 8 * (60 * 60), // 8 hours.
'session.cookie.name' => 'SimpleSAMLSessionID',
'session.cookie.lifetime' => 0,
'session.cookie.secure' => false,
```

- **Defines session duration** (8 hours).
- Uses a **secure cookie** to manage sessions.

3.5.2.7 Storage Configuration

```
'store.type' => 'phpsession',
```

- Defines **session storage type**:
  - `phpsession` → Uses PHP's built-in session storage.
  - `sql` → Uses a **MySQL/PostgreSQL database**.
  - `memcache` → Uses **Memcached** for distributed storage.
  - `redis` → Uses **Redis** for faster authentication handling.

3.5.2.8 Protocol Configuration

```
'enable.saml20-idp' => true,
'enable.shib13-idp' => true,
```

- Enables **SAML 2.0** and **Shibboleth 1.3** identity provider functionality.

3.5.2.9 Metadata Handling

```
'metadatadir' => '/conf/metadata',
'metadata.sources' => array(
          array('type' => 'flatfile')
),
```

- **Defines where metadata files are stored.**

- Uses a **flatfile-based metadata system**, meaning metadata is stored as local files instead of a database.

There are other configurations but for this guide we will leave them out because they are unnecessary for our context.

4. SimpleSAMLphp proxy configuration and customization.

In this section, we will explore how to customize the configurations we have previously analyzed. We will begin by modifying the necessary files **before deploying** the application with Docker Compose. Once the deployment is complete, we will proceed with adjusting certain settings directly on the newly deployed machine.

For the purpose of this guide, we assume that the software has neither been uploaded from a local machine nor downloaded from GitHub. Additionally, we consider that the user is already connected to the remote machine via SSH. Therefore, we will proceed directly with downloading, configuring, and installing the software from scratch on the target system.

Another crucial aspect to define is the **domain name** under which the application will run. It is essential to have a **domain name already assigned to the IP address** of the machine. However, this process is outside the scope of this guide, as it falls under a different area of responsibility. Specifically, in the context of **ILC4CLARIN**, our system administrator, **Alessandro Enea**, has requested the domain name from **GARR**, as he is the only authorized person to handle this task.

We assume that the developer is already connected in SSH to the reference machine[28] with the domain name already assigned. In our case, the domain name associated with the Palermo GARR node virtual machine is as follows: **sp.ilc4clarin.ilc.cnr.it**.

4.1 Download Docker simpleSAMLphp from GitHub

The first thing we need to do, after logging into SSH to the virtual machine, is to download the Docker configuration package of simpleSAMLphp. To do this we need to have Git installed on the system[29]. In addition, you must obtain super user privileges using the command:

```
sudo -su
```

We begin operations by placing ourselves in a file system layer where we can operate and launch the various deployment commands. Typically, the folder we will refer to is `/root`.

To download the configuration, you need to type the following command:

```
git clone https://github.com/simplesamlphp/docker-simplesamlphp.git
```

Once the command is run, a folder will be created which is located at the following path:

```
/root/docker-simplesamlphp
```

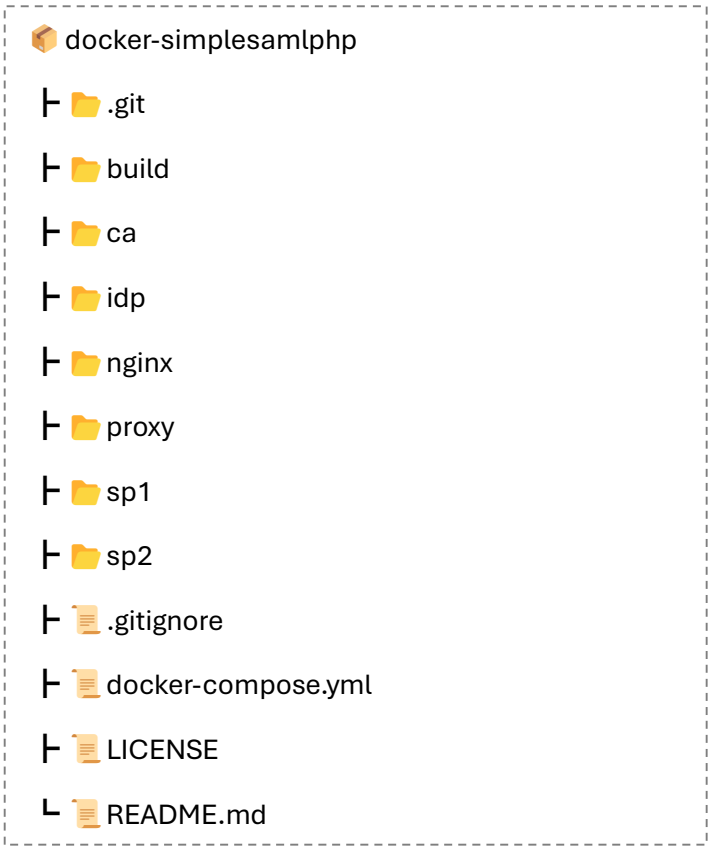
[28] For those interested, this is a good guide to connecting in SSH to a remote machine: https://www.digitalocean.com/community/tutorials/how-to-use-ssh-to-connect-to-a-remote-server

[29] This is a good guide for installing Git on a machine: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

Enter the configuration folder through the command:

```
cd docker-simplesamlphp
```

This is the folder as it appears at the time of download:



The next step is to download the simplesamlphp software[30] which will be taken from the docker-compose and will be copied to the docker container; to do this you need to run the following command[31]:

```
curl -L -o simplesamlphp-2.3.2-full.tar.gz
https://github.com/simplesamlphp/simplesamlphp/releases/download/v2.3.2/simplesamlphp-2.3.2-
full.tar.gz
```

[30] At the time this guide was created, the latest version available is 2.3.6; the version currently installed on the server is 2.3.2.

[31] To accomplish this step, make sure you have curl installed on your system; to do so, I recommend this guide: https://www.cyberciti.biz/faq/how-to-install-curl-command-on-a-ubuntu-linux/

Explanation of this command:

- **`curl`** → Command to download files via HTTP/HTTPS.
- **`-L`** → Automatically follows redirects (required for GitHub).
- **`-o simplesamlphp-2.3.2-full.tar.gz`** → Save the file with this name.

Now we need to extract the archive the folder with the name simplesamlphp with the command:

```
tar -xvf simplesamlphp-2.3.2-full.tar.gz
```

Rename the extracted folder:

```
mv simplesamlphp-2.3.2-full.tar.gz simplesamlphp
```

Remember that you must install the software through the ‘`composer install`’ command. This should be done once the machine has been deployed through the command ‘`docker compose up -d`’.

Now we have to modify the Dockerfile.

4.2 Editing Dockerfile

The modification of the Dockerfile is very important because it involves huge changes from the original version. We illustrate here the main points of modification.

4.2.1 Base Image

- **Original Dockerfile:** Uses **amd64/php:7.3.13-apache**, which is **outdated** (PHP 7.3 reached end-of-life in December 2021).

- **Modified Dockerfile:** Uses **php:8.1-apache**, a **newer, more secure, and performant** PHP version.

4.2.2 Apache Document Root

- **Original Dockerfile:** Sets the document root to `/code/www`.
- **Modified Dockerfile:** Changes it to `/code/public`.

4.2.3 Installed Dependencies (apt-get install)

- **Original Dockerfile**: Installs:

```
curl git libmemcached-dev libpng-dev unzip zlib1g-dev
```

- **Modified Dockerfile**: Installs more dependencies, including:

```
apt-transport-https ca-certificates gnupg libcurl4-openssl-dev \
          libssl-dev libonig-dev libicu-dev cron
```

4.2.4 Installed PHP Extensions

- **Original Dockerfile:** Installs:

```
docker-php-ext-install -j5 gd mbstring mysqli pdo pdo_mysql
```

- **Modified Dockerfile**: Adds:

```
docker-php-ext-install -j5 gd mbstring mysqli pdo pdo_mysql intl
```

4.2.5 PECL Extensions

- **Original Dockerfile:** Installs:

```
pecl install memcached redis xdebug
docker-php-ext-enable memcached redis xdebug
```

- **Modified Dockerfile:** Installs **Xdebug manually** instead of using PECL:

```
RUN curl -L https://xdebug.org/files/xdebug-3.1.6.tgz -o xdebug.tgz && \
     tar -xvzf xdebug.tgz && \
     rm xdebug.tgz && \
     cd xdebug-3.1.6 && \
     phpize && \
     ./configure && \
     make && \
     make install && \
     cd .. && \
     rm -rf xdebug-3.1.6 && \
     echo "zend_extension=$(find /usr/local/lib/php/extensions/ -name xdebug.so)" >
/usr/local/etc/php/conf.d/xdebug.ini
```

4.2.6 Apache Configuration

- **Original Dockerfile:** Copies `apache2.conf` and `mpm_prefork.conf` into:

```
/etc/apache2/
/etc/apache2/mod_available/
```

- **Modified Dockerfile:** Fixes a typo (`mod_available` → `mods-available`):

```
/etc/apache2/mods-available/
```

4.2.7 Startup Script Handling

- **Original Dockerfile:**

```
COPY startup.sh /
ENTRYPOINT ["/startup.sh"]
```

- **Modified Dockerfile:**

```
COPY startup.sh /startup.sh
RUN chmod +x /startup.sh
ENTRYPOINT ["/startup.sh"]
```

If you should happen to have problems with the ENTRYPOINT clause you can remove this instruction. Within this script there is only the 'composer-install' instruction to install the simplesamlphp packages, but this can also be done a-posteriori (after deploying the machine, see section 7.1)

4.2.8 CMD (Default Command Execution)

- **Original Dockerfile:** Runs:

```
CMD ["php", "-S", "0.0.0.0:8732", "-t", "/code/www"]
```

- **Modified Dockerfile:** Adjusts:

```
CMD ["php", "-S", "0.0.0.0:8732", "-t", "/code/public"]
```

You can see the full code for the production Dockerfile in Appendix A7.

It is possible to edit the Dockerfile using a command-line text editor such as `vim`[32]. To edit the file, let's assume that the user is already located at the root level of the folder, then run the `ls -ls` command. You should see a screen like this:

```
20 -rw-rw-rw-  1 root root 18431 Jun 13  2024 LICENSE
 4 -rw-rw-rw-  1 root root   800 Jun 13  2024 README.md
 4 -rw-rw-rw-  1 root root  1189 Sep 17 13:54 apache.md
 4 drwxrwxrwx  2 root root  4096 Nov 11 13:58 build
 4 drwxrwxrwx  2 root root  4096 Nov 11 13:58 ca
 4 -rw-rw-rw-  1 root root  1220 Feb 18 14:23 docker-compose.yml
 4 drwxrwxrwx  3 root root  4096 Nov 11 13:58 idp
 4 drwxrwxrwx  2 root root  4096 Feb  5 12:09 nginx
 4 drwxrwxrwx  3 root root  4096 Feb 18 15:49 proxy
 4 drwxrwxrwx 19 root root  4096 Feb  5 11:53 simplesamlphp
 4 drwxrwxrwx  3 root root  4096 Nov 11 13:58 sp1
 4 drwxrwxrwx  3 root root  4096 Nov 11 13:58 sp2
```

Go to the build folder using the `cd /build` command. To edit the file, use the `vi Dockerfile` command. A screen with the text of the Dockerfile will appear. Once you have changed all the lines in the file, you can exit and save by press ESC and typing `:wq`, then hitting ENTER.[33]

Now let's edit the Apache configuration.

[32] You can install vim in Ubuntu through these commands:

```
sudo apt-get update
sudo apt-get install vim
```

[33]

4.3 Editing Apache Configuration (internal to simplesamlphp)

The Apache configuration file is at the same level as the Dockerfile, so you will need to use the `vi apache2.conf` command to edit the file.

Here are some highlights on editing the original configuration file.

4.3.1 Change in Document Root

- **Original:**

```
<Directory /code/www>
  DocumentRoot /code/www
</Directory>
```

- **Customized:**

```
<Directory /code/public>
  DocumentRoot /code/public
</Directory>
```

4.3.2 VirtualHost ServerName Update

- **Original:**

```
ServerName https://${HOSTNAME}.tutorial.stack-dev.cirrusidentity.com
```

- **Customized:**

```
ServerName https://${HOSTNAME}.ilc4clarin.ilc.cnr.it
```

4.3.3 Security and Access Rules Changes

- **Original:**

```
<Directory /code/www>
  Options Indexes FollowSymLinks
```

```
    AllowOverride None
    Require all granted
</Directory>
```

- **Customized:**

```
<Directory /code/public>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
```

You can the see the entire configuration at Appendix 8.

Now let's proceed to edit the docker-compose.yml file.

4.4 Editing docker-compose.yml

The `docker-compose.yml` file is a very important file because it defines the system architecture of the virtual machines; in theory it should be edited last, because there are still so many changes to be made; however, we anticipate this topic since **we do not have to deploy the machines right away**; moreover, there are no particular configurations crucial to the operation of the whole thing and we therefore consider it as a weak customization (those related to proxy configuration will be considered strong customizations).

Here are the highlights of the changes from the original version.

4.4.1 Simplification of Services

The original docker-compose.yml defines multiple services:

- **IDP (idp.tutorial.stack-dev.cirrusidentity.com)**
- **Proxy (proxy.tutorial.stack-dev.cirrusidentity.com)**
- **Two Service Providers (sp1 and sp2)**
- **Databases (mysql, redis, memcached)**
- **Certificate Generator (gencert)**
- **Nginx**

The customized version **removes most services**, keeping only:

- **A single Service Provider (sp.ilc4clarin.ilc.cnr.it)**

- **Nginx**

4.4.2 Change in Service Provider Structure

- **Original**: Uses **separate services** for IDP, proxy, and SPs.
- **Customized**: Merges these into **one service (sp.ilc4clarin.ilc.cnr.it)** acting as the **proxy**.

4.4.3 Container Naming and Hostnames

- **Original**: Uses multiple hostnames (sp1, proxy.tutorial.stack-dev.cirrusidentity.com).
- **Customized**: Uses explicit **container names** (sp.ilc4clarin.ilc.cnr.it, nginx_sso) and sets hostname: proxy.

4.4.4 Command & Entrypoint Handling

- **Original**: Uses command: apache2 -D FOREGROUND for sp1.
- **Customized**: Uses:

```
entrypoint: ["/bin/sh", "-c", "exec apache2-foreground"]
```

4.4.5 Change in Volume Mounts

- **Original**: Mounts:

```
- ./simplesamlphp:/code
- ./idp:/conf
```

- **Customized**: Adds:

```
- /etc/letsencrypt:/etc/letsencrypt
```

These references to certificates are not yet present in your machine, as we have not yet covered this part (see Chapter 5). Also, sharing the entire Letsencrypt folder is ABSOLUTELY NOT GOOD PRACTICE[34].

4.4.6 Change in Nginx Configuration

- **Original**: Uses ports **80:80** and **443:443**.
- **Customized**: Changes ports to:

```
- '84:80'
- '2443:443'
```

You can see the entire cod of this file at Appendix 9.

4.5 Editing nginx.conf

The nginx configuration file is used to reroute connections from a previous Reverse Proxy (in our case, the Apache server operating in the main Portainer context) to the proxy/service provider machine.

This component could theoretically not exist, as it represents an additional layer that, however, is not important for the functional purposes of the infrastructure[35].

The file is located under the path `/nginx/nginx.conf`.

Here are the main differences between the two configurations.

[34] The sharing of the entire folder is due to the fact that the certificates in letsencrypt are organized in two different folders:

- `/archive/sp.ilc4clarin.ilc.cnr.it` → contains the original certificates, even the expired ones
- `/live/sp.ilc4clarin.ilc.cnr.it` → contains symbolic references only to the original certificates that have just been renewed

Bind only one folder would make no sense, as you would lose references to the original files. Ideally, two binding points should be created, each for each folder mentioned.

[35] In fact, a reverse proxy already exists, consisting of the httpd:2.4 machine located at the root node of the Docker configuration. An additional optimization step would be to directly configure the Apache server to reroute the connection to the machine named sp.ilc4clarin.ilc.cnr.it. This configuration persists as there has been no serious restructuring work, but this is the first optimization to be planned for the future.

4.5.1 Worker Connections Configuration

- **Original:**

```
events {
}
```

- **Customized:**

```
events {
    worker_connections 1024;
}
```

4.5.2 SSL Configuration

- **Original:**

```
server {
  listen 443 ssl;
  ssl_certificate /etc/nginx/cert.pem;
  ssl_certificate_key /etc/nginx/cert-key.pem;
}
```

- **Customized: SSL configuration is removed.**

4.5.3 Redirect from HTTP to HTTPS

- **Original:**

```
server {
  listen 80;
  server_name _ default_server;
  return 301 https://$host$request_uri;
}
```

- **Customized: This redirect is removed.**

4.5.4 Proxy Pass Configuration

- **Original:**

```
location / {
    set $authengine_upstream "http://$host:8732";
    proxy_pass $authengine_upstream;
}
```

- **Customized:**

```
location / {
    proxy_pass  http://172.20.0.3:8732/;
}
```

The IP address on this configuration is a hypothetical IP address, you need to connect the final address once the package has been deployed with docker compose. You can enter the machine name (sp.ilc4clarin.ilc.cnr.it) with the port on which the internal Apache server is listening (8732) attached.

You can see the entire nginx configuration at Appendix 10.

4.5.6 Server Name and Client Size Limits

- **Original:**

```
server_name _ default_server;
```

- **Customized:**

```
server_name sp.ilc4clarin.ilc.cnr.it;
client_max_body_size 3000m;
```

4.6 Editing simpleSAMLphp proxy configurations

This part represents the most crucial part of the configurations. In fact, it is the most complex part compared to all the other configurations, because there are so many aspects to take into account: on the one hand, some systemic aspects have to be taken into account, and on the other hand, some metadata configuration aspects have to be taken into account. Thanks to the intensive study of this

program/protocol and the support of the people mentioned in Section 1.3, it was possible to create a working configuration.

In the meantime, let us illustrate the main configuration, the one related to the operation of the proxy.

4.6.1 Editing config.php

The `config.php` file is the most meaty configuration file. It's located under `/proxy` folder and it consists of at least a thousand lines of configuration, but not all of them are important and require our attention. Let's go to analyze and explain the peculiar changes.

4.6.1.1 Base URL Configuration

- **Original:**

```
'baseurlpath' => 'https://proxy.tutorial.stack-dev.cirrusidentity.com/',
```

- **Customized:**

```
'baseurlpath' => 'https://sp.ilc4clarin.ilc.cnr.it/',
```

MANDATORY : you must to set the baseurl corresponding to your domain name

4.6.1.2 Certificate Directory

- **Original:**

```
'certdir' => 'cert/',
```

- **Customized:**

```
'certdir' => '/etc/letsencrypt/live/sp.ilc4clarin.ilc.cnr.it/',
```

Remember that still this reference to the folder does not exist, observe Chapter 5 for creating SSL certificates.

4.6.1.3 Contact Information

- **Original:**

```
'technicalcontact_email' => 'na@example.org',
```

- **Customized:**

```
'technicalcontact_email' => 'michele.mallia@cnr.it',
```

4.6.1.4 Assertion Validity Period

- **Only in Customized Version:**

```
'assertion.validity' => 28800, // 8 hours
```

MANDATORY: This configuration is necessary for the assertions to work, you cannot set the validity of the assertions to zero as the requests would last literally nothing.

4.6.1.5 Admin Password

- **Original:**

```
'auth.adminpassword' => '1234',
```

- **Customized:**

```
'auth.adminpassword' => ###############',
```

INFO: It is not clear if it is possible to generate a password dynamically through some command in php launched from this context. Use only secure passwords.

4.6.1.6 Database Configuration

- **Original:**

```
'database.dsn' => 'mysql:host=localhost;dbname=saml',
'database.username' => 'simplesamlphp',
'database.password' => 'secret',
```

- **Customized:**

```
'database.dsn' => 'mysql:host=saml-mysql;dbname=simplesamlphp',
'database.username' => 'dbuser',
'database.password' => 'dbpassword',
```

INFO: This configuration is only valid in cases where you want to enable the oidc module, as you do not need to have a database since the storage method set in the “store.type” variable is fixed to “phpsession”. It was not possible to test the database connection since there were some communication problems that have not yet been clarified, and due to business necessity it was not possible to go into this in depth.

4.6.1.7 Modules Enabled

- **Original:**

```
'module.enable' => array(
     'exampleauth' => TRUE,
     'saml' => TRUE,
     'core' => NULL,
),
```

- **Customized:**

```
'module.enable' => array(
     'exampleauth' => TRUE,
     'saml' => TRUE,
     'admin' => TRUE,
     'core' => NULL,
     'cron' => TRUE,
     'metarefresh' => TRUE
),
```

Don't forget to install the cron and metarefresh modules through the docker-php-ext-enable command! Please see section 7.4.

4.6.1.8 Metadata Configuration

- **Original:**

```
'metadatadir' => '/conf/metadata',
```

- **Customized:**

```
'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'),
),
```

INFO: This particular division into folders will be better explained when we discuss the metarefresh module. Please, see section 7.4.1.

4.6.1.9 Session Security

- **Original:**

```
'session.cookie.secure' => false,
```

- **Customized:**

```
'session.cookie.secure' => true,
```

MANDATORY: All requests must to be passed in a secure encrypted channel

[TODO: eventualmente, inserire modifiche per il logging, mostrando che ci sono diverse possibilità di scrittura su video o su file; in questo caso, bisognerà mostrare la modifica della configurazione su file per questioni di sicurezza e di vantaggi pratici]

4.6.2 Editing authsources.php

The `authsources.php` file is the one responsible for storing variables containing information about authentication processes and their sources. Thefollowing are mainly established in this file:

- Service provider definitions (default-sp)
- References to discovery services
- SSL certificates
- Metadata with information related to the institution in which the service provider operates

Here are the highlights of the configuration.

4.6.2.1 Service Provider (SP) Entity ID

- **Original:**

```
'entityID' => null,
```

- **Customized:**

```
'entityID' => 'https://sp.ilc4clarin.ilc.cnr.it',
```

MANDATORY: in version 2.3.2 of simplesamlphp this parameter is mandatory

4.6.2.2 Identity Provider (IdP) Configuration

- **Original:**

```
'idp' => 'https://idp.tutorial.stack-dev.cirrusidentity.com/saml2/idp/metadata.php',
```

- **Customized:**

```
'idp' => null,
```

INFO: the IdP field should be left at null because we need to make sure that the return request for external IdPs must go through the Service Provider and its internal IdP, not another IdP.

4.6.2.3 Discovery Service URL

- **Original:**

```
'discoURL' => null,
```

- **Customized:**

```
'discoURL' => 'https://discovery.clarin.eu/',
```

4.6.2.4 Private Key and Certificate

- **Original:**
  - ❌ No private key or certificate specified.

- **Customized:**

```
'privatekey' => 'private_key.pem',
'certificate' => 'certificate.pem',
```

These certificates are not the Letsencrypt certificates; although they are in the same folder, these are the openssl certificates of 3+ years duration[36].

4.6.2.5 Metadata description for CLARIN SPF

This part consists of defining the service provider's metadata; this information is EXTREMELY important as it is the service provider's own "identity card" that all other infrastructures need to be able to identify themselves as such.

Documenting with CLARIN's official instructions for building a service provider was essential in the creation of this section.[37]

Here is a description of the metadata.

4.6.2.5.1 General Service Provider Information

- **Display Name:**
  - **English:** *Service Provider for the CLARIN Research Infrastructure provided by ILC-CNR*
  - **Italian:** *Service Provider per l'infrastruttura di Ricerca CLARIN erogato da ILC-CNR*

- **Description:**
  - **English:** *ILC-CNR for the CLARIN-IT consortium: service provider for federated authentication.*

[36] For technical issues, federations require certificates that last more than six months as a minimum, up to a maximum of five years. Letsencrypt certificates last 90 days, which would imply significantly advanced monitoring regarding certificate rollover.

[37] Regarding Service Provider design, CLARIN by default tells you how to create a Service Provider in Shibboleth. Because it is an outdated technology, it was necessary to turn to simplesamlphp for this project. Documentation on metadata can be found at this address:
https://www.clarin.eu/content/guidelines-saml-metadata-about-your-sp

- **Italian:** *ILC-CNR per il consorzio CLARIN-IT: service provider per l'autenticazione federata*.

- **Privacy Policy URL[38][39]:**
  - Privacy Statement (available in English & Italian)

- **Information URL:**
  - Service Information → same reference of privacy policy

- **Logos:**
  -  [40](80x53 pixels)
  -  [41](16x16 pixels)

4.6.2.5.2 Attribute Requirements

This service provider requires specific attributes from identity providers for authentication:

- **Requested Attributes:**
  - eduPersonPrincipalName → urn:oid:1.3.6.1.4.1.5923.1.1.1.6
  - email → urn:oid:0.9.2342.19200300.100.1.3
  - displayName → urn:oid:2.16.840.1.113730.3.1.241
  - eduPersonTargetedID → urn:oid:1.3.6.1.4.1.5923.1.1.1.10
  - eduPersonScopedAffiliation → urn:oid:1.3.6.1.4.1.5923.1.1.1.9

- **Required Attributes (mandatory for authentication):**
  - eduPersonPrincipalName

[38] The privacy policy page is a static html page that is not integrated into the simplesamlphp package. It contains information about data processing from information on the ILC4CLARIN repository site found at this address: https://dspace-clarin-it.ilc.cnr.it/repository/xmlui/page/privacypolicy
[39] The procedures for creating this page will be explained in a separate section.
[40] This is the URL:
https://sp.ilc4clarin.ilc.cnr.it/img/ilc4clarin_80x53.png
[41] This is the URL:
https://sp.ilc4clarin.ilc.cnr.it/img/ilc4clarin_16x16.png

- email
- displayName
- eduPersonTargetedID
- eduPersonScopedAffiliation

- **Name Format:**
  - Uses the **URI-based** attribute name format: urn:oasis:names:tc:SAML:2.0:attrname-format:uri

4.6.2.5.3 Contact Information

The following individuals are listed as points of contact for different roles:

- **Administrative Contact:**
  - **Name:** Michele Mallia
  - **Email:** michele.mallia@cnr.it
  - **Phone:** (+39) 3392804180
  - **Organization:** *Consiglio Nazionale delle Ricerche (CNR)*
- **Technical Contact:**
  - **Same as Administrative Contact**
- **Support Contact:**
  - **Same as Administrative Contact**

4.6.2.5.4 Organization Information

- **Organization Name:**
  - **English:** *National Research Council (CNR)*
  - **Italian:** *Consiglio Nazionale delle Ricerche (CNR)*
- **Organization Display Name:**
  - **English:** *CNR Institute for Computational Linguistics "Antonio Zampolli"*
  - **Italian:** *CNR Istituto di Linguistica Computazionale "Antonio Zampolli"*

- **Organization Website:**
    - English Version
    - Italian Version

4.6.2.5.5 Registration Information

- **Authority:**
    - urn:mace:sp.ilc4clarin.ilc.cnr.it

- **Registration Date:**
    - *February 10, 2025, 14:01 UTC*

- **Registration Policies:**
    - Privacy Policy

4.6.2.5.6 Entity Categories & Compliance

The SP adheres to **recognized SAML entity categories** for compliance with international research federations:

- **Data Protection Code of Conduct:**
  http://www.geant.net/uri/dataprotection-code-of-conduct/v1
- **Research & Scholarship Category:**
  http://refeds.org/category/research-and-scholarship
- **CLARIN Member Category:**
  http://clarin.eu/category/clarin-member

4.7 Editing simpleSAMLphp internal files

In this section we explain how to modify some internal files in the source code of simplesamlphp. This is necessary to make it work, as the issue with source files is that there are some parts of the code that

make references to different metadata. In fact, for this step it is necessary to modify only two files that take as metadata identifier 'idpentityid' (in our context, in metadata and information exchanges through the SAML protocol, the field to be considered is 'entityID') changing it to the appropriate value.

The files in question are two:

- **SP.php** (line 760) → located under:

```
simplesamlphp/modules/saml/src/Auth/Source/SP.php
```

- **ServiceProvider.php** (line 208) → located under:

```
simplesamlphp/modules/saml/src/Controller/ServiceProvider.php
```

Editing these files should not be done since modifying the source code of a program is never a good practice. However, these changes have been tested and do not impact other parts of the code in any way. Keep these changes in mind when you will need to update the software.

4.7.1 Editing SP.php

To edit the file you have to use vim. If you are at the root level of the project, you can type the command:

```
vi /simplesamlphp/modules/saml/src/Auth/Source/SP.php
```

Then, type:

```
:760
```

And press ENTER.

You should be taken to line 760 of the code and you should be able to see this code snippet:

```
$params = [
            'entityID' => $this->entityId,
            'return' => $returnTo,
            'returnIDParam' => 'idpentityid',
        ];
```

This code snippet governs the type of value to be returned by the discovery service. You can see this peculiarity if you analyze a URL being tested for authentication (through the 'test' section of simplesamlphp or through any test call):

```
https://discovery.clarin.eu/?entityID=https%3A%2F%2Fsp.ilc4clarin.ilc.cnr.it&return=https%3A%2F%2Fsp.ilc4clarin.ilc.cnr.it%2Fmodule.php%2Fsaml%2Fsp%2FdiscoResponse%3FAuthID%3D_20a2d2fcdd0e78b22a6a326616c9490044deb96745%253Ahttps%253A%252F%252Fsp.ilc4clarin.ilc.cnr.it%252Fmodule.php%252Fadmin%252Ftest%252Fdefault-sp&returnIDParam=entityID
```

With the original configuration, the system would fail because, among the metadata of the federation's external IdPs, there is no attribute that matches that name. In the CLARIN federation, or more normally among all federation IdPs, the correct attribute is “entityID” (within the `md:EntityDescriptor` node).

The simplesamlphp system itself would fail. By changing the value from `‘idpentityid’` to `‘entityID’` the system no longer gave an error. So, to edit the file, press the I key to enter ‘Insert’ mode on vim, stand with the keyboard at that exact spot and start editing. Once the changes have been made, press ESC and then type `:wq` to edit and exit.

This is the final result:

```
$params = [
            'entityID' => $this->entityId,
            'return' => $returnTo,
            'returnIDParam' => 'entityID',
        ];
```

4.7.3 Editing ServiceProvider.php

It is also necessary to do the same thing for the other file. Then use this command to edit the file:

```
vi simplesamlphp/modules/saml/src/Controller/ServiceProvider.php
```

Then, type:

```
:208
```

And press ENTER.

You should be able to see this part of the code:

```
$idpEntityId = $request->query->get('idpentityid');
```

Then, use edit procedures with vim and change `‘idpentityid’` to `‘entityID’`. This is the result:

```
$idpEntityId = $request->query->get('entityID');
```

Then, save and exit the program.

5. Creation of SSL certificates

Having reached this point, we have already made the bulk of the changes. However, more changes still need to be made, this time outside the software context of simplesamlphp. In fact, it is necessary to move on to the creation of the SSL certificates, which will be of two types:

- The first are the **Letsencrypt certificates** that will be needed for the Apache server that acts as a higher-level reverse proxy (container httpd:2.4), with a 90-day duration;
- The second are the **self-signed certificates** generated with **openssl** with a duration of three years for assertions in SAML

Let us then proceed with the creation of these certificates and observe the appropriate configurations to exploit them.

Make sure you have both certbot (Letsencrypt) and openssl installed on the machine you are operating on[42].

5.1 Letsencrypt certificates

In order to install Letsencrypt certificates in our environment, we need to perform several operations in our context:

- First, we need to **turn off the Apache container** (httpd:2.4), as certbot, in order to create the certificates, needs to use port 80 to make queries to the web server and perform **ACME testing** and **Domain Validation[43]**
    - To do this, you can:
        - go to the Portainer in the container section, select with a checkbox the container and press on the "Stop" button

[42] Here are instructions for installing openssl and certbot:
https://www.webhi.com/how-to/how-to-install-openssl-on-ubuntu-linux/
https://www.digitalocean.com/community/tutorials/how-to-secure-apache-with-let-s-encrypt-on-ubuntu
[43] If you want to learn more about this topic, this is a good guide that gives an overview of the processes and protocols for creating Letsencrypt certificates:
https://letsencrypt.org/how-it-works/

- use the console, type "`docker ps`," check if there is a machine named "`Apache`" and use the container id for the command

```
docker container stop <CONTAINER_ID>
```

- Run the command to **generate the certificates**
    - Make sure you are **super user**
    - The type:

```
certbot certonly --apache -d sp.ilc4clarin.ilc.cnr.it
```

    - Certificates will be saved on:

`/etc/letsencrypt/live/sp.ilc4clarin.ilc.cnr.it`

- Make sure there are **no more active processes** on port 80
    - Use this command (Linux/Mac)

```
kill -9 `sudo lsof -t -i:80`
```

- Turn the http-2.4 container **back on**
    - **Using Portainer**
    - **Using the command:**

```
docker container start <CONTAINER_ID>
```

Now that you have generated the Letsencrypt certificates, let's go edit the Apache configuration files to hook them up to the main web-server.

5.1.1 Configuring h2iosc-sp.conf

To configure certificates, go to the binding folder shared between the host and the container and open the httpd.conf file using the command:

```
vi /root/apache-server/conf/httpd.conf
```

Look for the string “Include” to go to the configuration part where there are configuration inclusions by typing on the `vim` console:

```
/Include
```

Scroll using the N key until we find instructions such as:

```
# Virtual hosts
Include conf/extra/httpd-vhosts.conf
Include conf/extra/httpd-ssl.conf
```

Then type ENTER. Press I Key and insert:

```
Include conf/extra/h2iosc-sp.conf
```

Press ESC and type `:wq`.

Now you need to create the configuration file h2iosc-sp.conf. Use the command:

```
vi /root/apache-server/conf/extra/h2iosc-sp.conf
```

You can see the full code in Appendix 12, here are the highlights.

5.1.1.1 Automatic Redirect from HTTP to HTTPS

- The first <VirtualHost *:80> block ensures that any request made over HTTP (port 80) is **permanently redirected** to HTTPS.
- This is done via:

```
Redirect permanent / https://sp.ilc4clarin.ilc.cnr.it/
```

5.1.1.2 SSL Configuration for Secure Communication (Port 443)

- The second <VirtualHost *:443> block handles **HTTPS requests**.
- SSL is enabled via:

```
SSLEngine on
```

- Let's Encrypt certificates are used for encryption:

```
SSLCertificateFile /etc/letsencrypt/live/sp.ilc4clarin.ilc.cnr.it/fullchain.pem
SSLCertificateKeyFile /etc/letsencrypt/live/sp.ilc4clarin.ilc.cnr.it/privkey.pem
```

5.1.1.3 Reverse Proxy to nginx

- The ProxyPass and ProxyPassReverse directives **forward requests** from the public Apache server to an internal service running at http://172.20.0.4:80/.
- This means:
  - The Apache server is acting as a **reverse proxy** for another service.
  - External users interact with https://sp.ilc4clarin.ilc.cnr.it/, but internally, requests are served by http://172.20.0.4:80/.

The mentioned IP address refers to NGINX, which in turn acts as a reverse proxy to the proxy/service provider container. Then replace this IP address either with the name of the container.[44][45]

5.1.1.4 Preserving the Original Host Header

- ProxyPreserveHost On ensures that the **original Host header** is passed to the backend instead of being replaced by the proxy's hostname.
- This helps if the backend server expects requests for sp.ilc4clarin.ilc.cnr.it.

The next step is related to the creation of self-signed certificates for SAML assertions.

[44] Once the optimization is made, you will need to put the IP address of the proxy/service provider's machine in place of NGINX's IP address. Remember to enter the correct port!

[45]

5.2 Self-Signed certificates[46]

In order for the service provider to make assertions valid for more than three years (and also for technical issues required by the federations), it is necessary to create long-lived certificates, even self-signed ones.

To do this, you need to navigate from the project folder (/root/docker-simplesamlphp) to the Letsencrypt certificates folder (/etc/letsencrypt/live/sp.ilc4clarin.ilc.cnr.it) via the command:

```
cd /etc/letsencrypt/live/sp.ilc4clarin.ilc.cnr.it
```

Once you have positioned yourself at that point in the system, run the following command:

```
openssl req -x509 -nodes -days 1095 -newkey rsa:4096 \
  -keyout private_key.pem \
  -out certificate.pem \
  -subj "/C=IT/ST=Pisa/L=Pisa/O=ILC-CNR/OU=CLARIN/CN=sp.ilc4clarin.ilc.cnr.it"
```

Explanation of this command:

1. openssl req

This invokes the OpenSSL utility to create and process certificate signing requests (CSRs) and self-signed certificates.

2. -x509

- Tells OpenSSL to create a self-signed certificate instead of a certificate signing request (CSR).
- X.509 is the standard for SSL/TLS certificates.

3. -nodes

- Stands for "no DES encryption."
- This prevents the private key from being encrypted with a passphrase.
    - This is useful for automation but makes the key less secure.

4. -days 1095

- Specifies the **validity period** of the certificate in days.
- 1095 days = **3 years**.

[46] This part is closely related to section 4.6.2.4.

5. -newkey rsa:4096

- Generates a new private key along with the certificate.
- Uses RSA encryption with a 4096-bit key length (strong security).

6. -keyout private_key.pem

- Specifies the output filename for the generated private key.
- Saves it as private_key.pem.

7. -out certificate.pem

- Specifies the output filename for the self-signed certificate.
- Saves it as certificate.pem.

8. -subj "/C=IT/ST=Pisa/L=Pisa/O=ILC-CNR/OU=CLARIN/CN=sp.ilc4clarin.ilc.cnr.it"

- Provides the Distinguished Name (DN) for the certificate in one line, bypassing interactive input.
- Breakdown of the DN fields:
  - /C=IT → Country: Italy
  - /ST=Pisa → State/Region: Pisa
  - /L=Pisa → Locality/City: Pisa
  - /O=ILC-CNR → Organization: ILC-CNR
  - /OU=CLARIN → Organizational Unit: CLARIN
  - /CN=sp.ilc4clarin.ilc.cnr.it → Common Name (CN): This is the domain name for which the certificate is valid.

Once you run the command, you will then get two files, one called private_key.pem and another file called certificate.pem. These files will be the files for ensuring the encryption of SAML assertions.

These two files have already been declared in the proxy configuration file, please look at section 4.6.2.4 to cover the configurations on authentication.

5.3 DFN-AAI certificate

This has to be done as this certificate is needed to download metadata related to IdPs in the German DFN-AAI federation. It is an operation that is closely related to the metadata and metarefresh configuration part (please see section 7.4.1).

To obtain the certificate you must download it using curl. To do this, locate yourself in the Letsencrypt certificate folder through the command:

```
cd /etc/letsencrypt/live/sp.ilc4clarin.ilc.cnr.it/
```

Then, run the following command:

```
curl -o dfn-aai.pem https://www.aai.dfn.de/metadata/dfn-aai.pem
```

Once this is done, you can use the ls command to check for the presence of the file. You should find a list like this:

```
4 -rw-r--r-- 1 www-data www-data  692 Feb  5 11:28 README
0 lrwxrwxrwx 1 www-data www-data   48 Feb  5 11:28 cert.pem -> ../../archive/sp.ilc4clarin.ilc.cnr.it/
cert1.pem

4 -rw-r--r-- 1 www-data www-data 2057 Feb  7 15:13 certificate.pem
0 lrwxrwxrwx 1 www-data www-data   49 Feb  5 11:28 chain.pem -> ../../archive/sp.ilc4clarin.ilc.cnr.it/
chain1.pem

4 -rw-rw-rw- 1 www-data www-data 2813 Feb  5 09:58 dfn-aai.pem
0 lrwxrwxrwx 1 www-data www-data   53 Feb  5 11:28 fullchain.pem -> ../../archive/sp.ilc4clarin.ilc.cnr
.it/fullchain1.pem

4 -rw------- 1 root     root     2488 Feb 18 15:12 oidc_module.key
4 -rw------- 1 www-data www-data 3272 Feb  7 15:13 private_key.pem
0 lrwxrwxrwx 1 www-data www-data   51 Feb  5 11:28 privkey.pem -> ../../archive/sp.ilc4clarin.ilc.cnr
.it/privkey1.pem
```

The fact that you see www-data refers to a change you will have to make after deploying the service provider container, which you will find in section 7.2.

5.4 Permissions on certificates

To make sure that certificates can be enjoyed by any type of user, it is necessary that these files can be read by everyone through the setting of read, write, and possibly execute permissions. To do this, first of all it is necessary to digital the following command:

```
chmod -R 777 /etc/letsencrypt/archive/sp.ilc4clarin.ilc.cnr.it/*
```

Through this command, all certificates in the folder can be read correctly by the service provider container.

This is a partial operation, you will also need to give permissions to the www-data user to manage these files. In this regard, I refer you to section X to address this topic, as this change needs to be done post-deployment of the containers.

6. Deployment of Proxy

Having reached this point, we have all the cards on the table to be able to deploy our proxy. To do this, we need to get to the root level of the project folder (making sure we are in the same place as the docker-compose.yml) and run the command:

```
docker compose up -d
```

Once you run the command, you should be able to see the two compose containers:

- sp.ilc4clarin.ilc.cnr.it.
- nginx_sso

6.1 Link the proxy network to the Apache Reverse Proxy

In addition to the two containers, the command we just ran also creates a new network, which must be hooked to the container that serves as the main reverse proxy, so that our Apache container can communicate with the containers we just created. It is possible to see the new network through the command:

```
docker network ls
```

You should be able to see a list of such networks:

```
NETWORK ID     NAME                                                DRIVER    SCOPE
df8ba88d87c9   bridge                                              bridge    local
e9aa9cd323e9   docker-simplesamlphp                                bridge    local
```

To connect the reverse proxy to the network created by the compose, you must first get the `CONTAINER ID` of the reverse proxy (just use the `docker ps` command and get the `CONTAINER ID` of the Apache container) and then run this command:

```
docker network connect <network_name> <container_name>
```

If you want, you can also use the Portainer GUI, following the same steps:

- Go to the “Container” section
- Click on the container “Apache”
- Go to the bottom, look for section “Connected networks”

- In the “Join a network” section, go to the drop-down menu and select the proxy network (its name can be, depending on the compose configuration, docker-simplesamlphp_default)
- Once the network is selected, press on the blue “Join” button.

Figure x: overview of connected network of Apache container

6.1.1 Test the connection between containers

To verify that the connection is effective, you can connect to the Apache container console and run commands such as ping or curl to the nginx reverse proxy container and/or the service provider container.

Two routes can be followed to do this:

- via the command line
- through Portainer

6.1.1.1 Test via command line

Here are the steps to follow:

- Make sure you are connected to the remote server hosting the Portainer and Docker engine.
- Run the “docker ps” command to get the list of active containers
- Get the id of the Apache container
- Run the following command:
    - Using `bash`: `docker exec -it <container_id> bash`
    - Using `sh`: `docker exec -it <container_name> sh`
- Test whether there is already a “curl” or “ping” command.
    - Se non presenti, usare i seguenti comandi: `apt update && sudo apt install -y curl iputils-ping`

- Se presenti, lanciare i seguenti comandi:
    - Test connectivity with nginx_sso:
        - `curl http://nginx_sso:80`
        - `ping -c 4 nginx_sso`
    - Test connectivity with sp.ilc4clarin.ilc.cnr.it:
        - `curl http://sp.ilc4clarin.ilc.cnr.it:8732`
        - `ping -c 4 http://sp.ilc4clarin.ilc.cnr.it`
- If you performed the command with curl, you should see output like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8">
    <meta http-equiv="refresh"
content="0;URL='https://sp.ilc4clarin.ilc.cnr.it/module.php/core/welcome'">
    <title>Redirect</title>
  </head>
  <body>
    <h1>Redirect</h1>
      <p>You were redirected to: <a id="redirlink"
href="https://sp.ilc4clarin.ilc.cnr.it/module.php/core/welcome">https://sp.ilc4clarin.ilc.cnr.i
t/module.php/core/welcome</a>
        <script type="text/javascript">document.getElementById("redirlink").focus();</script>
      </p>
  </body>
```

- If you performed the command with ping, you should see output like this:
    - nginx_sso:

```
PING nginx_sso (172.20.0.4): 56 data bytes
64 bytes from 172.20.0.4: icmp_seq=0 ttl=64 time=0.147 ms
64 bytes from 172.20.0.4: icmp_seq=1 ttl=64 time=0.155 ms
64 bytes from 172.20.0.4: icmp_seq=2 ttl=64 time=0.159 ms
64 bytes from 172.20.0.4: icmp_seq=3 ttl=64 time=0.194 ms
--- nginx_sso ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.147/0.164/0.194/0.000 ms
```

    - sp.ilc4clarin.ilc.cnr.it:

```
PING sp.ilc4clarin.ilc.cnr.it (172.20.0.3): 56 data bytes
64 bytes from 172.20.0.3: icmp_seq=0 ttl=64 time=0.312 ms
64 bytes from 172.20.0.3: icmp_seq=1 ttl=64 time=0.158 ms
64 bytes from 172.20.0.3: icmp_seq=2 ttl=64 time=0.165 ms
64 bytes from 172.20.0.3: icmp_seq=3 ttl=64 time=0.185 ms
--- sp.ilc4clarin.ilc.cnr.it ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.158/0.205/0.312/0.063 ms
```

If you got a positive response, it means that you were able to correctly configure communication with containers. If not, try repeating the previous steps and verify that the “docker-simplesamlphp” stack network is connected to the Apache container.

6.1.1.2 Test via Portainer GUI

If you want instead to use the Portainer GUI to access the Apache container console, you need to go to the “Container” section, look for the Apache container row, and press the fourth button from the left, as you can see in the figure below.

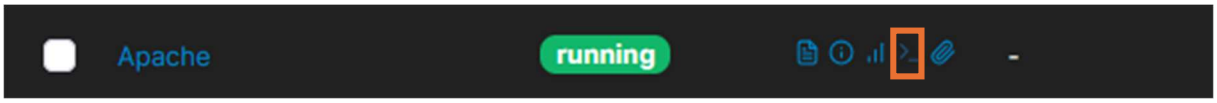


Figure 4: in orange, the icon you have to select for enter in the container's console

After clicking, a screen will pop up asking you which type of console to access the container with. You select “bash,” but if by chance it doesn't work, you can still log in with “sh.” Next, press “connect” and you will see the container console from where you can launch the commands illustrated above.

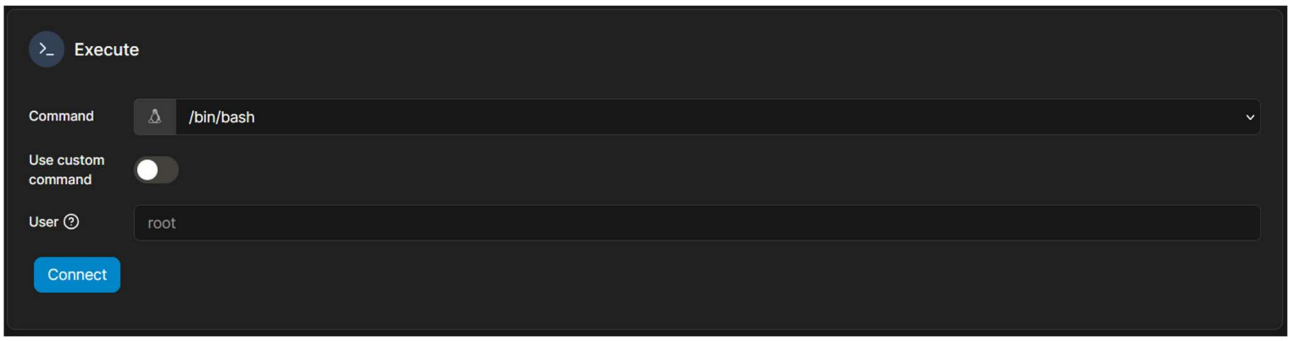


Figure x: Select the console

6.2 Update h2iosc-sp.conf and nginx.conf

Once we have deployed the two containers, we need to update both reverse proxies so that they point to the correct servers.

We start with Apache's h2iosc-sp.conf file:

```
vi /root/apache-server/conf/extra/h2iosc-sp.conf
```

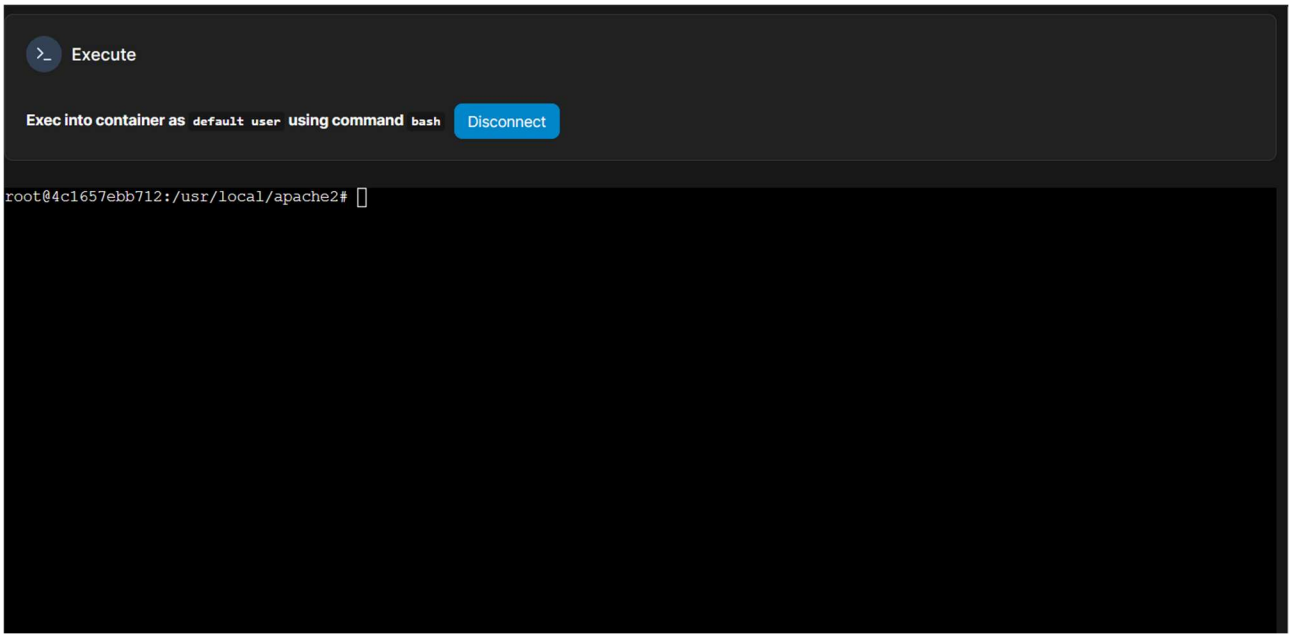


Figure x: container console screen

After, update the lines inside the <Location> clause:

```
ProxyPass http://nginx_sso:80/
ProxyPassReverse http://nginx_sso:80/
ProxyPreserveHost On
```

So the Apache server can reach the nginx reverse proxy.

Then, update the nginx.conf located under `/root/docker-simplesamlphp/nginx`:

```
vi /root/docker-simplesamlphp/nginx/nginx.conf
```

Change only the proxy_pass rule with:

```
proxy_pass  http://sp.ilc4clarin.ilc.cnr.it:8732/;
```

So the request can be forwarded to the simplesamlphp container.

Always remember to restart the Apache and NGINX containers to make the changes effective.

Now, the containers of the service provider and its reverse proxy are ready. We must now move on to activating the modules, extensions, and cron configuration and metarefresh after deployment.

7. Post-deployment configurations

In this section we will deal with how to set up the changes necessary for the application to work after its deployment with docker compose. Some key changes involving several components will be addressed, from PHP settings to configuring the metarefresh module and cron.

7.1 Install dependencies with composer

To install the required dependencies for SimpleSAMLphp using Composer, follow these steps:

Connect to the SimpleSAMLphp Container

First, you need to access the running SimpleSAMLphp Docker container shell. You can do this via Docker CLI or Portainer:

Using Docker CLI:

```
docker exec -it <container_name_or_id> /bin/bash
```

Replace <container_name_or_id> with your actual SimpleSAMLphp container name or ID (in our case the container is named **sp.ilc4clarin.ilc.cnr.it**).

Using Portainer:

- Log in to your Portainer instance.
- Navigate to **Containers** in the sidebar.
- Select your running SimpleSAMLphp container from the list.
- Click the **Exec Console** tab.
- Select **/bin/bash** from the shell dropdown.
- Click **Connect** to access the container shell.

Navigate to the Application Directory

Once inside the container shell, ensure you are located in the SimpleSAMLphp main directory. Run:

```
cd /code
```

This is the main working directory where the SimpleSAMLphp code and configuration files reside.

Install Dependencies with Composer

With the terminal positioned in the /code directory, execute the following command to install all dependencies defined in the composer.json file:

```
composer install
```

Composer will now download and install the necessary PHP packages and dependencies. Ensure there are no errors during installation. If you encounter any issues, carefully check the Composer output for specific details.

Restart the Container

After installing the dependencies, restart the container to ensure all changes take effect. You can restart it via Docker CLI or Portainer:

- **Using Docker CLI:**

```
docker restart <container_name_or_id>
```

- **Using Portainer:**
  - Go to the **Containers** section.
  - Find your SimpleSAMLphp container.
  - Click the **Restart** button from the container's action menu.

This step ensures that any new configurations or packages are properly loaded into the running environment.

7.2 Change ownership of SSL certificates

To ensure that the web server process has the correct permissions to access the SSL certificates, follow the steps below:

Access the Service Provider Container

Connect to the Docker container that runs the Service Provider instance.

Navigate to the Let's Encrypt Directory

Once inside the container, go to the Let's Encrypt directory:

```
cd /etc/letsencrypt
```

Change Ownership of SSL Certificate Directories

Run the following commands to assign the correct ownership to the SSL certificate folders:

```
chown -R www-data:www-data /live/sp.ilc4clarin.ilc.cnr.it/
chown -R www-data:www-data /archive/sp.ilc4clarin.ilc.cnr.it/
```

This ensures that the web server (typically running under the www-data user) can access and serve the SSL certificates.

7.3 Edit PHP.ini configuration

To ensure proper PHP behavior and compatibility for your SimpleSAMLphp service, follow these steps to adjust the PHP configuration inside the container:

Navigate to the PHP Configuration Directory

Inside the container, move to the directory containing the PHP configuration files:

```
cd /usr/local/etc/php
```

Identify Available Configuration Files

You will find the following files:

- `conf.d/`
- `php.ini-production`
- `php.ini-development`

Rename the Production Configuration File

Rename php.ini-production to php.ini to use it as the active configuration:

```
mv php.ini-production php.ini
```

Edit the PHP.ini File

Open the php.ini file using a text editor such as vim:

```
vi php.ini
```

Then make the following changes:

Uncomment the intl extension

Locate line 936 and remove the semicolon (;) from the beginning:

```
;extension=intl
```

becomes:

```
extension=intl
```

Increase the Memory Limit

Locate line 435 and change the value of memory_limit to 2000M:

```
memory_limit = 128M
```

becomes:

```
memory_limit = 2000M
```

Save and Exit

After editing the file in vim, save your changes by pressing `Esc`, then typing `:wq` and pressing `Enter`.

These adjustments ensure that the PHP runtime has sufficient memory and the required internationalization extension enabled for SimpleSAMLphp to operate correctly.

7.4 Install metarefresh and cron modules

Among the key post-deployment tasks is the installation and configuration of specific SimpleSAMLphp modules that ensure proper metadata management and automation. In the following section, we focus on setting up the metarefresh and cron modules, which play a critical role in maintaining up-to-date federation metadata and enabling scheduled operations within the service provider container.

7.4.1 Configuring metarefresh

The metarefresh module in SimpleSAMLphp is designed to automate the retrieval and refreshing of metadata from external sources, such as federations or trusted identity providers. This ensures that the Service Provider (SP) has up-to-date information about the available Identity Providers (IdPs), which is critical for maintaining interoperability and trust within federated identity infrastructures.

The module can be configured to periodically fetch metadata, apply transformations or filtering (such as blacklists or whitelists), and output it into SimpleSAMLphp's metadata format. It works in conjunction with the cron module to automate execution

To install and configure the metarefresh module for SimpleSAMLphp in the Service Provider container, follow these steps:

Navigate to the Application Directory

Access the /code directory inside the container:

```
cd /code
```

Install the metarefresh Module

Use Composer to install the simplesamlphp-module-metarefresh package:

```
composer require simplesamlphp/simplesamlphp-module-metarefresh
```

This will download and install the module into the `/code/modules/metarefresh` directory.

Module Activation

The metarefresh module is assumed to be already enabled in the config.php file, located at:

- Inside the container: /conf/config.php
- From the host: /root/docker-simplesamlphp/proxy/config.php

This configuration step has already been completed as described in Section 4.6.1.7 "Modules Enabled".

Copy the Configuration Template

After the module is installed, copy its default configuration template from the module's directory to the active configuration folder:

```
cp /code/modules/metarefresh/config-templates/*.php /conf/
```

This makes the metarefresh configuration files available for further customization, such as scheduling metadata refresh from trusted sources.

7.4.1.1 Metadata Sets

This configuration includes three key metadata sets, each serving a specific role in the federated identity scenario for ILC4CLARIN:

- **DFN**: The German Research Network (DFN) federation metadata is included to allow interoperability with German institutions and services that are part of the DFN-AAI infrastructure. This is particularly important for accessing resources or services shared through international academic collaborations.
- **CLARIN**: The CLARIN SPF metadata aggregates identity providers from the CLARIN Service Provider Federation. This enables users from participating institutions across Europe to authenticate and access services within the CLARIN infrastructure.

- **CLARIN-IDP**: This set contains metadata specifically about the CLARIN ERIC Identity Provider(s), ensuring that the SP can recognize and communicate with the central IdP(s) maintained by the CLARIN ERIC entity.

7.4.1.2 Key Configuration Elements

Below is a line-by-line explanation of the active configuration parts from the metarefresh.php file used in the SimpleSAMLphp Service Provider container:

```
<?php

$config = [
```

Defines the configuration array for the metarefresh module.

```
    'sets' => [
```

Begins the definition of metadata sets. Each set includes parameters for a metadata source.

```
        'dfn' => [
            'cron' => ['hourly'],
```

Defines the "dfn" metadata set. The cron schedule is set to run hourly.

```
        'sources' => [
                [
                    'src' => 'http://www.aai.dfn.de/metadata/dfn-aai-edugain+idp-metadata.xml',
```

Specifies the metadata source URL for the DFN federation.

```
                    'certificates' => [
                        'dfn-aai.pem',
                    ],
```

Points to the PEM file used to verify the digital signature of the metadata.

```
                    'template' => [
                        'tags' => ['dfn'],
                    ],
```

Assigns a tag to the imported metadata entities for internal identification.

```
        ],
```

```
        ],
            'expireAfter' => 34560060,
```

Sets the maximum validity duration for the metadata (in seconds).

```
        'outputDir' => '/conf/metadata/dfn/',
```

Directory path where the processed metadata files will be saved.

```
            'outputFormat' => 'flatfile',
```

Indicates the metadata will be stored in flat PHP files readable by SimpleSAMLphp.

```
            'types' => ['saml20-idp-remote'],
```

Only includes SAML 2.0 IdPs in the metadata output.

```
        ],
```

Closes the configuration block for the "dfn" set.

The other sections follow the same syntax. You can find the complete code of the metarefresh configuration in Appendix A13.

7.4.2 Configuring CRON

In order to automate the execution of the metarefresh module and ensure that the Service Provider (SP) continuously retrieves and processes updated metadata, it is necessary to configure a **CRON job** that triggers the process at regular intervals.

This metadata refresh is critical for maintaining up-to-date information about Identity Providers (IdPs) across federations. By ensuring the SP always has the latest metadata, we reduce the risk of failed assertions due to expired certificates, mismatched keys, or stale configurations. This is essential for the stability and security of federated authentication workflows.

7.4.2.1 Process Overview

The metarefresh process is executed by making an HTTP request to a specific URL on the Service Provider. This is typically done using the curl command, which targets the cron endpoint of SimpleSAMLphp and specifies the metadata set to refresh.

A typical command would look like this:

```
curl -sS https://sp.ilc4clarin.ilc.cnr.it/module.php/cron/run/hourly/<SECRET_KEY>
```

7.4.2.2 Running CRON Jobs on the Host System

Since this deployment runs within Docker containers, the CRON job must be configured **on the host machine**, not inside the container. The containerized nature of the application means that the

environment inside the container is transient and typically not suitable for persistent CRON scheduling.

Steps to set up the CRON process:

1. **Ensure the CRON service is active** On most Linux systems, verify that the CRON service is running and enabled:

```
sudo systemctl enable cron
sudo systemctl start cron
```

2. **Edit the host CRON configuration** Open the crontab editor for the desired user:

```
crontab -e
```

3. **Add a CRON entry** Schedule the job based on your refresh policy. For example, to run it every hour:

```
0 * * * *  curl -sS https://sp.ilc4clarin.ilc.cnr.it/module.php/cron/run/hourly/<SECRET_KEY>
```

You can replicate this line for other metadata sets like clarin and clarin-idp, or use a wildcard tag if configured accordingly.

For more details on available CRON services and options provided by SimpleSAMLphp, visit the built-in CRON info page:

https://sp.ilc4clarin.ilc.cnr.it/module.php/cron/info

This page lists all registered CRON modules, available tags, and their current status, making it easier to monitor and troubleshoot automated metadata refresh operations.

By configuring CRON correctly, your Service Provider remains synchronized with its federated partners, supporting secure, reliable, and seamless SSO authentication workflows.

7.5 Modify the home template

In this section, we explain how to update the footer of the SimpleSAMLphp home page to reflect hosting details.

Since the software is deployed on a server within the GARR Cloud infrastructure, it is important to mention this publicly on the site.

1. **Locate the Template Directory**

Access the host machine where the SimpleSAMLphp installation is located. Navigate to the templates directory:

```
cd /simplesamlphp/templates
```

The complete path must be:

```
/root/docker-simplesamlphp/simplesamlphp/templates
```

2. Edit the Footer Template File

Open the _footer.twig template file using your preferred text editor:

```
vi _footer.twig
```

3. Insert Hosting Information

Add the following HTML snippet where appropriate within the footer template:

```
Hosted by <a href="https://cloud.garr.it/" target="_blank">Cloud GARR</a>
```

4. Save and Exit

After editing, save the file and exit the editor.

There is no need to restart any services after this change; the updated footer will appear immediately on the next page load. You can see the complete code of the footer at Appendix 14.

7.6 Add privacy page

To comply with legal obligations regarding personal data processing, it was necessary to add a privacy policy page to the Service Provider.

Since SimpleSAMLphp does not natively manage content pages, a static HTML page was created outside the main application, placed under the web-accessible directory managed by Apache (`simplesamlphp/public`).

The process involves the following steps:

- Create a file named `privacy.html` and insert the necessary content as specified in Appendix 15.
- Place the `privacy.html` file inside the `simplesamlphp/public directory`.
- Restart the corresponding Docker container to ensure the new file is properly served.

This privacy page briefly describes the ILC4CLARIN Service Provider, outlining its function as a federated authentication service within the CLARIN SPF infrastructure, built on SimpleSAMLphp to provide secure and seamless Single Sign-On (SSO) access.

The page specifies the data controller and technical contact (Institute for Computational Linguistics "A. Zampolli" and Michele Mallia, respectively), the applicable jurisdiction (Italy), and details the types of personal data processed during authentication, such as identifiers, email addresses, affiliations, and logs containing metadata and access details.

It explains the purpose of processing personal data, which includes authentication, security, service monitoring, and compliance with federation policies. It also clearly states that personal data are not disclosed to third parties outside the ILC-CNR for CLARIN-IT team.

Finally, the page informs users about their rights to access, rectify, or delete their personal data, describes the data retention policy (deletion after five years of inactivity or upon user request), and commits to adhering to the Data Protection Code of Conduct for service providers in the research and education sector.

8. Federation with CLARIN SPF

At this stage, after completing all the required configurations for our proxy, the system is technically ready to operate. We have prepared the environment, adjusted system settings, installed and configured essential modules, and customized the deployment to match our specific infrastructure needs. However, before we can proceed with authentication tests and verify that the Service Provider (SP) operates correctly within the broader federated ecosystem, an essential final step must be completed: the official federation with the CLARIN Service Provider Federation (SPF).

The CLARIN SPF represents a vital component of the CLARIN infrastructure. It serves as a bridge connecting services and Identity Providers (IdPs) across different national, institutional, and international federations. By joining the CLARIN SPF, users from participating institutions gain the ability to seamlessly authenticate and access CLARIN services using the credentials issued by their home organizations. This fosters interoperability, increases accessibility, and supports broader collaboration across the research community. Membership in the CLARIN SPF not only facilitates access but also demonstrates compliance with rigorous technical and policy standards required for participation in trusted federated environments.

For a detailed understanding of the CLARIN SPF technical requirements and operational procedures, it is highly recommended to consult the official CLARIN documentation available at the following link:

https://www.clarin.eu/content/service-provider-federation-technical-details

This resource provides extensive information, including:

- The technical criteria and metadata requirements that an SP must fulfill to be accepted into the CLARIN SPF.
- Guidelines on metadata formatting, encryption standards, and supported protocols.
- Operational and organizational policies, including security, privacy, and attribute release guidelines.
- Contact information and step-by-step instructions for initiating the federation process.

Federating with the CLARIN SPF involves several administrative and technical steps:

- Preparing and validating the SP metadata to ensure it meets CLARIN's specifications.
- Submitting the metadata along with necessary documentation to the CLARIN SPF administrators.
- Participating in a review process during which the metadata and the SP's technical readiness are evaluated.
- Receiving final approval and integrating the SP into the CLARIN SPF metadata aggregates, allowing the service to interact with a wide range of IdPs.

It is crucial to highlight that proper metadata management is essential for a successful federation. Accurate, up-to-date metadata ensures that authentication assertions are trusted, that service

discovery is efficient, and that users encounter no technical barriers when accessing federated services.

Once the initial federation request has been submitted, and while waiting for final approval, it is necessary to verify that the configured metadata is correctly propagated and publicly available. Metadata propagation guarantees that the federation administrators can retrieve, validate, and integrate the SP into the global metadata aggregates without issues.

In the next section, we will provide detailed instructions on how to manage the metadata propagation process, verify its availability, and prepare for the final federation verification by CLARIN SPF administrators.

8.1 Metadata Propagation Process for CLARIN SPF Integration

Following the completion of the Service Provider (SP) configuration and preparation for federation, the next and highly critical phase involves the systematic propagation of its SAML metadata. Accurate metadata propagation is indispensable for the formal integration of the SP into the CLARIN Service Provider Federation (SPF) and is equally crucial for establishing seamless interoperability with broader federated infrastructures, including eduGAIN and various national identity federations.

The propagation process requires meticulous attention to detail. Even minor deviations, inconsistencies, or failures to meet federation standards can result in significant delays, additional review cycles, or outright rejection of the SP metadata. Thus, utmost care must be taken at every step.

8.1.1 General Process Overview

In alignment with CLARIN SPF's official technical documentation and best practices, the metadata propagation process is structured as follows:

1. **Prepare the SAML Metadata**

The foundation of a successful federation integration is properly structured SAML metadata. The metadata must conform rigorously to recognized standards such as the SAML 2.0 specifications.

It is strongly recommended to model the metadata structure on well-established examples from production Service Providers already federated within CLARIN SPF. Doing so ensures alignment with tested and accepted practices.

Important references include:

- The Shibboleth Metadata for SP Documentation, which provides detailed conceptual guidance.
- CLARIN-specific metadata preparation guidelines, which outline particular standards and expectations unique to the CLARIN environment.

2. **Obtain the SP Metadata**

For our specific deployment, the metadata has been made available through the public-facing SimpleSAMLphp instance. The metadata can be retrieved at:

- https://sp.ilc4clarin.ilc.cnr.it/module.php/saml/sp/metadata/default-sp

By visiting this URL, it is possible to download the latest XML-formatted metadata describing the Service Provider's operational parameters. You can see a complete version of the SP Metadata at Appendix 16.

3. **Fork the CLARIN Metadata Repository**

Given that our institution had previously formalized participation agreements with CLARIN, no new administrative registration was necessary. Consequently, the technical process simply involved:

- Forking the CLARIN SPF SPs Metadata GitHub repository to create a personal working copy.
- Cloning the forked repository locally to enable direct editing and management.

4. **Insert the SP Metadata**

The downloaded md:EntityDescriptor XML file must be correctly placed into the /metadata directory within the forked repository. It is advisable to validate the file manually using XML schema validators and SAML-specific validation tools to preempt any syntax or semantic issues.

5. **Submit a Pull Request**

After committing the new metadata file to the repository and ensuring consistency, a pull request (PR) must be initiated. The pull request must target the master branch of the upstream CLARIN SPF SPs Metadata repository.

6. **Automated Validation**

Once the pull request is submitted, an automated validation process is triggered. The check_saml_metadata.sh script automatically evaluates the metadata for both syntactic correctness and semantic soundness.

- The results of this validation will be publicly visible within the pull request page.
- Only pull requests that successfully pass all automated checks proceed to manual administrative review.

7. **Metadata Review and Merging**

Provided that the automated validation passes without errors, the CLARIN central office will conduct a final manual inspection. This review ensures full compliance with federation policies and technical standards.

Upon successful completion of the review, the new SP metadata will be merged into the pre-production aggregates and scheduled for publication in the live CLARIN SPF metadata feeds.

8. **Propagation Timing and Expectations**
    - Within a few hours after merging, the Service Provider's metadata becomes part of the active CLARIN SPF metadata distribution.
    - Further propagation to eduGAIN and various national federations may take additional time, typically ranging from one to several days, due to external synchronization cycles outside CLARIN's direct control.

Additional Considerations and Best Practices

- Ensure all metadata fields, including EntityID, organization names, technical contact information, and X.509 certificates, are accurate and correspond precisely to the deployed SP configuration.
- Maintain a local backup of the metadata submitted for future audits or reference.
- Regularly monitor the pull request and validation process, responding quickly to any identified issues or reviewer comments to minimize delays.
- Consider subscribing to updates from the GitHub repository to stay informed about future metadata policy changes.

By following these structured steps with diligence and precision, the Service Provider can achieve robust and recognized integration within the CLARIN SPF, thereby enabling secure, standards-compliant federated authentication and facilitating seamless access for users across a global research community.

8.1.2 Check the status of metadata propagation

To monitor the state of metadata propagation and verify its ingestion into various federations, the CLARIN SPF provides a public status page:

- https://centres.clarin.eu/spf

This page offers a comprehensive overview of:

- The ingestion status of Service Providers' metadata across multiple European and international federations.
- Real-time validation results indicating whether metadata has been successfully integrated, pending review, or encountered technical issues.
- A list of federations, each linked to detailed reports about metadata ingestion, validation timestamps, and encountered problems if any.

By consulting this HTML page, administrators and technical personnel can quickly assess the federation status of their SP, identify bottlenecks, and intervene if problems are detected during the propagation and ingestion process.

9. Test the authentication

At this stage, after completing all the required configurations and metadata propagation procedures, we are ready to proceed with testing the authentication setup of the Service Provider.

SimpleSAMLphp provides a dedicated interface for testing the federation and authentication processes, making it straightforward to verify that the SP is properly integrated and capable of interacting with external Identity Providers (IdPs).

Before performing any tests, it is crucial to ensure that the Service Provider's metadata has been fully propagated within the federation network. This step is essential because successful metadata ingestion ensures that the SP is recognized and trusted by external IdPs. If the metadata is not yet propagated, authentication attempts will fail, as the IdPs will not have the necessary information to establish trust and process authentication requests.

You can monitor metadata propagation status via the CLARIN SPF status page:

- https://centres.clarin.eu/spf

Once metadata propagation is confirmed, authentication testing can proceed.

9.1 How to Perform an Authentication Test in SimpleSAMLphp

1. **Access the Service Provider Web Interface** Navigate to the SP main page:

```
https://sp.ilc4clarin.ilc.cnr.it/
```

2. **Access the Admin Page** Open the administration interface:

```
https://sp.ilc4clarin.ilc.cnr.it/admin/
```

3. **Authenticate as Administrator** Enter the credentials specified in the authsources.php configuration file of the Service Provider.

4. **Navigate to the "Test" Section** Once logged in, access the "Test" area from the admin dashboard.
5. **Select the Service Provider Authentication Source** Choose the authentication source corresponding to your SP, typically named default-sp.

6. **Redirect to CLARIN Discovery Service** You will be redirected to the CLARIN Discovery Service at:

```
https://discovery.clarin.eu/
```

7. **Select Your Home Identity Provider** From the list of federated institutions, select your home IdP. In this case, it would be:
    - Istituto di Linguistica Computazionale "A. Zampolli"

8. **Authenticate with Your Institutional Credentials** Log in using the username and password provided by your home organization.

9. **Review the Authentication Results** If the authentication is successful, you will be redirected to a confirmation page displaying the SAML assertion attributes returned by your IdP. These typically include:
    - Assertion Identifier
    - Email Address
    - Display Name
    - Affiliation
    - Other metadata fields as released by your IdP

If all expected attributes are correctly displayed, it confirms that the authentication and attribute release processes are functioning properly and that your Service Provider is fully operational within the CLARIN Service Provider Federation.

10. Practical implications and future directions

The deployment, configuration, and operationalization of the ILC4CLARIN Service Provider within the CLARIN SPF federation represent a crucial advancement in strengthening the digital infrastructure of the Institute for Computational Linguistics "A. Zampolli."

The establishment of a dedicated institutional Service Provider introduces substantial strategic and operational advantages. It guarantees that internal services, research applications, and digital platforms can seamlessly participate in an international federated authentication framework. This ensures interoperability with a wide array of external research and academic entities across Europe and beyond, fostering collaborations and data-sharing initiatives that require trusted digital identities.

Most importantly, through the development of its own Service Provider, the institute has laid a foundational layer for the creation of a comprehensive, centralized Single Sign-On (SSO) environment. This framework enables any web-based application deployed within the institute's domain to integrate into the broader federated ecosystem using standardized authentication protocols such as SAML 2.0 or OpenID Connect (OIDC), without the need for independent identity management implementations.

The practical benefits of this approach are significant and multidimensional:

- **Streamlined User Access**: Researchers, technical staff, administrative personnel, and external collaborators will be able to access a multitude of services using a single institutional credential, significantly simplifying their digital workflows.

- **Enhanced Security and Compliance**: By consolidating authentication under a centralized, federation-compliant service, the institute improves its security posture, reduces risks associated with credential proliferation, and ensures adherence to data protection and security standards.

- **Simplified IT Operations and Management**: The centralized identity framework will ease the administrative burden on IT departments, facilitating faster onboarding, easier account management, and secure deprovisioning processes when users leave the institution.

- **Elevated User Experience**: A consistent, intuitive login experience across all applications encourages greater user engagement, satisfaction, and adoption of institutional digital services.

- **Strategic Federation Participation**: Participation in CLARIN SPF opens opportunities for integration with eduGAIN and other international federations, enhancing the visibility and reputation of the institute within the global research infrastructure.

Looking forward, the Service Provider infrastructure can be further leveraged by:

- Expanding support for additional modern protocols such as OAuth 2.0 and enhancing OpenID Connect (OIDC) capabilities to facilitate mobile and lightweight application integrations.
- Implementing Attribute-Based Access Control (ABAC) mechanisms to enable fine-grained authorization policies based on user roles, affiliations, and specific research project memberships.
- Integrating additional internal and external services such as repositories, virtual research environments, computational resources, and collaborative platforms.
- Introducing multi-factor authentication (MFA) solutions to strengthen security for critical applications.
- Establishing monitoring and auditing frameworks to ensure continual compliance with evolving federation and security standards.

In conclusion, the successful implementation of the ILC4CLARIN Service Provider not only addresses an immediate technical need for federated authentication but also positions the institute as a forward-looking, digitally integrated research organization. By capitalizing on this foundational work, the Institute for Computational Linguistics "A. Zampolli" can significantly enhance its operational efficiency, research capabilities, user engagement, and international collaborations, ensuring sustainable growth and leadership in the digital humanities and computational linguistics communities.

Appendix

A1. docker-compose.yml (original, pre-customization):

```
version: '2'
services:
  idp.tutorial.stack-dev.cirrusidentity.com:
    build: build
    volumes:
      - ./simplesamlphp:/code
      - ./idp:/conf
    working_dir: /code
    environment:
      - SIMPLESAMLPHP_CONFIG_DIR=/conf/
    links:
      - mysql

  proxy.tutorial.stack-dev.cirrusidentity.com:
    build: build
    volumes:
      - ./simplesamlphp:/code
      - ./proxy:/conf
    working_dir: /code
    environment:
      - SIMPLESAMLPHP_CONFIG_DIR=/conf/

  sp1.tutorial.stack-dev.cirrusidentity.com:
    build: build
    hostname: sp1
    volumes:
      - ./simplesamlphp:/code
      - ./sp1:/conf
    working_dir: /code
    environment:
      - SIMPLESAMLPHP_CONFIG_DIR=/conf/
    command: apache2 -D FOREGROUND
    links:
      - redis

  sp2.tutorial.stack-dev.cirrusidentity.com:
    build: build
    volumes:
      - ./simplesamlphp:/code
      - ./sp2:/conf
    working_dir: /code
    environment:
      - SIMPLESAMLPHP_CONFIG_DIR=/conf/
    links:
      - memcached

  memcached:
    image: memcached

  mysql:
    image: mysql
    command: --default-authentication-plugin=mysql_native_password
    restart: always
    environment:
      MYSQL_DATABASE: sessions
      MYSQL_USER: dbuser
      MYSQL_PASSWORD: dbpassword
      MYSQL_ROOT_PASSWORD: rootpassword

```

```
  redis:
    image: redis

  gencert:
    image: cfssl/cfssl
    volumes:
      - .:/work
    working_dir: /work
    entrypoint: /bin/bash
    command:
      - ./ca/generate.sh

  nginx:
    image: nginx:stable
    links:
      - idp.tutorial.stack-dev.cirrusidentity.com
      - proxy.tutorial.stack-dev.cirrusidentity.com
      - sp1.tutorial.stack-dev.cirrusidentity.com
      - sp2.tutorial.stack-dev.cirrusidentity.com
    volumes:
      - ./nginx:/etc/nginx:ro
    ports:
      - '80:80'
      - '443:443'
```

A2. Dockerfile (original, pre-customization):

```
FROM amd64/php:7.3.13-apache
ENV APACHE_DOCUMENT_ROOT /code/www
RUN apt-get update && apt-get install -y --no-install-recommends \
  curl \
  git \
  libmemcached-dev \
  libpng-dev \
  unzip \
  zlib1g-dev \
  && rm -rf /var/lib/apt/lists/*
RUN docker-php-ext-install -j5 gd mbstring mysqli pdo pdo_mysql \
   && pecl install memcached redis xdebug \
   && docker-php-ext-enable memcached redis xdebug
RUN curl -sS https://getcomposer.org/installer | php -- --install-dir=/usr/local/bin --
filename=composer
RUN a2dismod mpm_event && a2enmod mpm_prefork
COPY apache2.conf /etc/apache2/
COPY mpm_prefork.conf /etc/apache2/mod_available/
COPY startup.sh /
ENTRYPOINT ["/startup.sh"]
CMD ["php", "-S", "0.0.0.0:8732", "-t", "/code/www"]
```

A3. NGINX configuration (original):

```
events {
}

http {
  resolver 127.0.0.11;
  server {
    listen 80;
    server_name _ default_server;
    return 301 https://$host$request_uri;
  }
  server {
    listen 443 ssl;
    ssl_certificate /etc/nginx/cert.pem;
    ssl_certificate_key /etc/nginx/cert-key.pem;
#    listen 80;
    location / {
      # To force re-resolution every time
      set $authengine_upstream "http://$host:8732";
      proxy_pass $authengine_upstream;
      proxy_http_version 1.1;
      proxy_set_header Upgrade $http_upgrade;
      proxy_set_header Connection 'upgrade';
      proxy_set_header Host $host;
      proxy_cache_bypass $http_upgrade;
    }
  }
}
```

A4. Apache configuration (internal to simplesamlphp):

```
Mutex file:/var/ default
PidFile /tmp/pidfile
Timeout 300
KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 5
User www-data
Group www-data
HostnameLookups Off
ErrorLogFormat "[%{u}t] [%l] %7F: %E: [client\ %a] %M% ,\ referer\ %{Referer}i"
ErrorLog /dev/stderr
LogLevel warn
ExtendedStatus on
IncludeOptional mods-enabled/*.load
IncludeOptional mods-enabled/*.conf
IncludeOptional conf-enabled/docker-php.conf
AddOutputFilterByType DEFLATE image/svg+xml

Listen 8732

<Directory />
      Options FollowSymLinks
      AllowOverride None
      Require all denied
</Directory>
AccessFileName .htaccess
<FilesMatch "^\.ht">
```

```
      Require all denied
</FilesMatch>
<Directory /code/www>
      Options Indexes FollowSymLinks
      AllowOverride None
      Require all granted
</Directory>
LogFormat "%v:%p %h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" vhost_combined
LogFormat "%h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %O" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

# from conf-enabled/security.conf
ServerTokens OS
ServerSignature On
TraceEnable Off

<VirtualHost *:8732>
      ServerName https://${HOSTNAME}.tutorial.stack-dev.cirrusidentity.com
      SetEnvIf Authorization "(.*)" HTTP_AUTHORIZATION=$1
      SetEnv HTTPS "on"
      UseCanonicalName on

      DocumentRoot /code/www

</VirtualHost>
```

A5. authsources.php (initial configuration)

```
<?php

$config = array(

    // This is a authentication source which handles admin authentication.
    'admin' => array(
        // The default is to use core:AdminPassword, but it can be replaced with
        // any authentication source.

        'core:AdminPassword',
    ),


    // An authentication source which can authenticate against both SAML 2.0
    // and Shibboleth 1.3 IdPs.
    'default-sp' => array(
        'saml:SP',

        // The entity ID of this SP.
        // Can be NULL/unset, in which case an entity ID is generated based on the metadata URL.
        'entityID' => null,

        // The entity ID of the IdP this should SP should contact.
        // Can be NULL/unset, in which case the user will be shown a list of available IdPs.
        'idp' => 'https://idp.tutorial.stack-dev.cirrusidentity.com/saml2/idp/metadata.php',

        // The URL to the discovery service.
        // Can be NULL/unset, in which case a builtin discovery service will be used.
        'discoURL' => null,

        /*
         * WARNING: SHA-1 is disallowed starting January the 1st, 2014.
```

```
         *
         * Uncomment the following option to start using SHA-256 for your signatures.
         * Currently, SimpleSAMLphp defaults to SHA-1, which has been deprecated since
         * 2011, and will be disallowed by NIST as of 2014. Please refer to the following
         * document for more information:
         *
         * http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf
         *
         * If you are uncertain about identity providers supporting SHA-256 or other
         * algorithms of the SHA-2 family, you can configure it individually in the
         * IdP-remote metadata set for those that support it. Once you are certain that
         * all your configured IdPs support SHA-2, you can safely remove the configuration
         * options in the IdP-remote metadata set and uncomment the following option.
         *
         * Please refer to the hosted SP configuration reference for more information.
         */
        //'signature.algorithm' => 'http://www.w3.org/2001/04/xmldsig-more#rsa-sha256',

        /*
         * The attributes parameter must contain an array of desired attributes by the SP.
         * The attributes can be expressed as an array of names or as an associative array
         * in the form of 'friendlyName' => 'name'. This feature requires 'name' to be set.
         * The metadata will then be created as follows:
         * <md:RequestedAttribute FriendlyName="friendlyName" Name="name" />
         */
        /*'name' => array(
             'en' => 'A service',
             'no' => 'En tjeneste',
          ),

          'attributes' => array(
            'attrname' => 'urn:oid:x.x.x.x',
        ),*/
        /*'attributes.required' => array (
            'urn:oid:x.x.x.x',
        ),*/
    ),

);
```

A6. config.php (initial configuration)

```
<?php
/*
 * The configuration of SimpleSAMLphp
 *
 */

$config = array(

    /*******************************
     | BASIC CONFIGURATION OPTIONS |
     *******************************/

    /*
     * Setup the following parameters to match your installation.
     * See the user manual for more details.
     */

    /*
     * baseurlpath is a *URL path* (not a filesystem path).
     * A valid format for 'baseurlpath' is:
     * [(http|https)://(hostname|fqdn)[:port]]/[path/to/simplesaml/]
     * (note that it must end with a '/')
```

```
     *
     * The full url format is useful if your SimpleSAMLphp setup is hosted behind
     * a reverse proxy. In that case you can specify the external url here.
     *
     * Please note that SimpleSAMLphp will then redirect all queries to the
     * external url, no matter where you come from (direct access or via the
     * reverse proxy).
     */
    'baseurlpath' => 'https://proxy.tutorial.stack-dev.cirrusidentity.com/',

    /*
     * The following settings are *filesystem paths* which define where
     * SimpleSAMLphp can find or write the following things:
     * - 'certdir': The base directory for certificate and key material.
     * - 'loggingdir': Where to write logs.
     * - 'datadir': Storage of general data.
     * - 'temdir': Saving temporary files. SimpleSAMLphp will attempt to create
     *   this directory if it doesn't exist.
     * When specified as a relative path, this is relative to the SimpleSAMLphp
     * root directory.
     */
    'certdir' => 'cert/',
    'loggingdir' => 'log/',
    'datadir' => 'data/',
    'tempdir' => '/tmp/simplesaml',

    /*
     * Some information about the technical persons running this installation.
     * The email address will be used as the recipient address for error reports, and
     * also as the technical contact in generated metadata.
     */
    'technicalcontact_name' => 'Administrator',
    'technicalcontact_email' => 'na@example.org',

    /*
     * The timezone of the server. This option should be set to the timezone you want
     * SimpleSAMLphp to report the time in. The default is to guess the timezone based
     * on your system timezone.
     *
     * See this page for a list of valid timezones: http://php.net/manual/en/timezones.php
     */
    'timezone' => null,



    /**********************************
     | SECURITY CONFIGURATION OPTIONS |
     **********************************/

    /*
     * This is a secret salt used by SimpleSAMLphp when it needs to generate a secure hash
     * of a value. It must be changed from its default value to a secret value. The value of
     * 'secretsalt' can be any valid string of any length.
     *
     * A possible way to generate a random salt is by running the following command from a unix shell:
     * tr -c -d '0123456789abcdefghijklmnopqrstuvwxyz' </dev/urandom | dd bs=32 count=1
2>/dev/null;echo
     */
    'secretsalt' => 'a43x8k07rk8798ejy0ap659dvyvh90mw',

    /*
     * This password must be kept secret, and modified from the default value 123.
     * This password will give access to the installation page of SimpleSAMLphp with
     * metadata listing and diagnostics pages.
     * You can also put a hash here; run "bin/pwgen.php" to generate one.
     */
    'auth.adminpassword' => '1234',

    /*
     * Set this options to true if you want to require administrator password to access the web
interface
```

```
     * or the metadata pages, respectively.
     */
    'admin.protectindexpage' => false,
    'admin.protectmetadata' => false,

    /*
     * Set this option to false if you don't want SimpleSAMLphp to check for new stable releases when
     * visiting the configuration tab in the web interface.
     */
    'admin.checkforupdates' => true,

    /*
     * Array of domains that are allowed when generating links or redirects
     * to URLs. SimpleSAMLphp will use this option to determine whether to
     * to consider a given URL valid or not, but you should always validate
     * URLs obtained from the input on your own (i.e. ReturnTo or RelayState
     * parameters obtained from the $_REQUEST array).
     *
     * SimpleSAMLphp will automatically add your own domain (either by checking
     * it dynamically, or by using the domain defined in the 'baseurlpath'
     * directive, the latter having precedence) to the list of trusted domains,
     * in case this option is NOT set to NULL. In that case, you are explicitly
     * telling SimpleSAMLphp to verify URLs.
     *
     * Set to an empty array to disallow ALL redirects or links pointing to
     * an external URL other than your own domain. This is the default behaviour.
     *
     * Set to NULL to disable checking of URLs. DO NOT DO THIS UNLESS YOU KNOW
     * WHAT YOU ARE DOING!
     *
     * Example:
     *   'trusted.url.domains' => array('sp.example.com', 'app.example.com'),
     */
    'trusted.url.domains' => array(),

    /*
     * Enable regular expression matching of trusted.url.domains.
     *
     * Set to true to treat the values in trusted.url.domains as regular
     * expressions. Set to false to do exact string matching.
     *
     * If enabled, the start and end delimiters ('^' and '$') will be added to
     * all regular expressions in trusted.url.domains.
     */
    'trusted.url.regex' => false,

    /*
     * Enable secure POST from HTTPS to HTTP.
     *
     * If you have some SP's on HTTP and IdP is normally on HTTPS, this option
     * enables secure POSTing to HTTP endpoint without warning from browser.
     *
     * For this to work, module.php/core/postredirect.php must be accessible
     * also via HTTP on IdP, e.g. if your IdP is on
     * https://idp.example.org/ssp/, then
     * http://idp.example.org/ssp/module.php/core/postredirect.php must be accessible.
     */
    'enable.http_post' => false,



    /*************************
     | ERRORS AND DEBUGGING |
     *************************/

    /*
     * The 'debug' option allows you to control how SimpleSAMLphp behaves in certain
     * situations where further action may be taken
     *
     * It can be left unset, in which case, debugging is switched off for all actions.
     * If set, it MUST be an array containing the actions that you want to enable, or
```

```
     * alternatively a hashed array where the keys are the actions and their
     * corresponding values are booleans enabling or disabling each particular action.
     *
     * SimpleSAMLphp provides some pre-defined actiones, though modules could add new
     * actions here. Refer to the documentation of every module to learn if they
     * allow you to set any more debugging actions.
     *
     * The pre-defined actions are:
     *
     * - 'saml': this action controls the logging of SAML messages exchanged with other
     * entities. When enabled ('saml' is present in this option, or set to true), all
     * SAML messages will be logged, including plaintext versions of encrypted
     * messages.
     *
     * - 'backtraces': this action controls the logging of error backtraces. If you
     * want to log backtraces so that you can debug any possible errors happening in
     * SimpleSAMLphp, enable this action (add it to the array or set it to true).
     *
     * - 'validatexml': this action allows you to validate SAML documents against all
     * the relevant XML schemas. SAML 1.1 messages or SAML metadata parsed with
     * the XML to SimpleSAMLphp metadata converter or the metaedit module will
     * validate the SAML documents if this option is enabled.
     *
     * If you want to disable debugging completely, unset this option or set it to an
     * empty array.
     */
    'debug' => array(
        'saml' => false,
        'backtraces' => true,
        'validatexml' => false,
    ),

    /*
     * When 'showerrors' is enabled, all error messages and stack traces will be output
     * to the browser.
     *
     * When 'errorreporting' is enabled, a form will be presented for the user to report
     * the error to 'technicalcontact_email'.
     */
    'showerrors' => true,
    'errorreporting' => true,

    /*
     * Custom error show function called from SimpleSAML_Error_Error::show.
     * See docs/simplesamlphp-errorhandling.txt for function code example.
     *
     * Example:
     *   'errors.show_function' => array('sspmod_example_Error_Show', 'show'),
     */



    /**************************
     | LOGGING AND STATISTICS |
     **************************/

    /*
     * Define the minimum log level to log. Available levels:
     * - SimpleSAML\Logger::ERR     No statistics, only errors
     * - SimpleSAML\Logger::WARNING No statistics, only warnings/errors
     * - SimpleSAML\Logger::NOTICE  Statistics and errors
     * - SimpleSAML\Logger::INFO    Verbose logs
     * - SimpleSAML\Logger::DEBUG   Full debug logs - not recommended for production
     *
     * Choose logging handler.
     *
     * Options: [syslog,file,errorlog]
     *
     */
    'logging.level' => SimpleSAML\Logger::DEBUG,
    'logging.handler' => 'errorlog',
```

```
    /*
     * Specify the format of the logs. Its use varies depending on the log handler used (for instance,
you cannot
     * control here how dates are displayed when using the syslog or errorlog handlers), but in general
the options
     * are:
     *
     * - %date{<format>}: the date and time, with its format specified inside the brackets. See the PHP
documentation
     *   of the strftime() function for more information on the format. If the brackets are omitted,
the standard
     *   format is applied. This can be useful if you just want to control the placement of the date,
but don't care
     *   about the format.
     *
     * - %process: the name of the SimpleSAMLphp process. Remember you can configure this in the
'logging.processname'
     *   option below.
     *
     * - %level: the log level (name or number depending on the handler used).
     *
     * - %stat: if the log entry is intended for statistical purposes, it will print the string 'STAT '
(bear in mind
     *   the trailing space).
     *
     * - %trackid: the track ID, an identifier that allows you to track a single session.
     *
     * - %srcip: the IP address of the client. If you are behind a proxy, make sure to modify the
     *   $_SERVER['REMOTE_ADDR'] variable on your code accordingly to the X-Forwarded-For header.
     *
     * - %msg: the message to be logged.
     *
     */
    //'logging.format' => '%date{%b %d %H:%M:%S} %process %level %stat[%trackid] %msg',

    /*
     * Choose which facility should be used when logging with syslog.
     *
     * These can be used for filtering the syslog output from SimpleSAMLphp into its
     * own file by configuring the syslog daemon.
     *
     * See the documentation for openlog (http://php.net/manual/en/function.openlog.php) for available
     * facilities. Note that only LOG_USER is valid on windows.
     *
     * The default is to use LOG_LOCAL5 if available, and fall back to LOG_USER if not.
     */
    'logging.facility' => defined('LOG_LOCAL5') ? constant('LOG_LOCAL5') : LOG_USER,

    /*
     * The process name that should be used when logging to syslog.
     * The value is also written out by the other logging handlers.
     */
    'logging.processname' => 'simplesamlphp',

    /*
     * Logging: file - Logfilename in the loggingdir from above.
     */
    'logging.logfile' => 'simplesamlphp.log',

    /*
     * This is an array of outputs. Each output has at least a 'class' option, which
     * selects the output.
     */
    'statistics.out' => array(// Log statistics to the normal log.
        /*
        array(
            'class' => 'core:Log',
            'level' => 'notice',
        ),
        */
```

```
        // Log statistics to files in a directory. One file per day.
        /*
        array(
            'class' => 'core:File',
            'directory' => '/var/log/stats',
        ),
        */
    ),



    /***********************
     | PROXY CONFIGURATION |
     ***********************/

    /*
     * Proxy to use for retrieving URLs.
     *
     * Example:
     *   'proxy' => 'tcp://proxy.example.com:5100'
     */
    'proxy' => null,

    /*
     * Username/password authentication to proxy (Proxy-Authorization: Basic)
     * Example:
     *   'proxy.auth' = 'myuser:password'
     */
    'proxy.auth' => false,



    /**************************
     | DATABASE CONFIGURATION |
     **************************/

    /*
     * This database configuration is optional. If you are not using
     * core functionality or modules that require a database, you can
     * skip this configuration.
     */

    /*
     * Database connection string.
     * Ensure that you have the required PDO database driver installed
     * for your connection string.
     */
    'database.dsn' => 'mysql:host=localhost;dbname=saml',

    /*
     * SQL database credentials
     */
    'database.username' => 'simplesamlphp',
    'database.password' => 'secret',

    /*
     * (Optional) Table prefix
     */
    'database.prefix' => '',

    /*
     * True or false if you would like a persistent database connection
     */
    'database.persistent' => false,

    /*
     * Database slave configuration is optional as well. If you are only
     * running a single database server, leave this blank. If you have
     * a master/slave configuration, you can define as many slave servers
     * as you want here. Slaves will be picked at random to be queried from.
     *
```

```
     * Configuration options in the slave array are exactly the same as the
     * options for the master (shown above) with the exception of the table
     * prefix.
     */
    'database.slaves' => array(
        /*
        array(
            'dsn' => 'mysql:host=myslave;dbname=saml',
            'username' => 'simplesamlphp',
            'password' => 'secret',
            'persistent' => false,
        ),
        */
    ),



    /*************
     | PROTOCOLS |
     *************/

    /*
     * Which functionality in SimpleSAMLphp do you want to enable. Normally you would enable only
     * one of the functionalities below, but in some cases you could run multiple functionalities.
     * In example when you are setting up a federation bridge.
     */
    'enable.saml20-idp' => true,
    'enable.shib13-idp' => true,
    'enable.adfs-idp' => false,
    'enable.wsfed-sp' => false,
    'enable.authmemcookie' => false,

    /*
     * Default IdP for WS-Fed.
     */
    'default-wsfed-idp' => 'urn:federation:pingfederate:localhost',

    /*
     * Whether SimpleSAMLphp should sign the response or the assertion in SAML 1.1 authentication
     * responses.
     *
     * The default is to sign the assertion element, but that can be overridden by setting this
     * option to TRUE. It can also be overridden on a pr. SP basis by adding an option with the
     * same name to the metadata of the SP.
     */
    'shib13.signresponse' => true,



    /***********
     | MODULES |
     ***********/

    /*
     * Configuration to override module enabling/disabling.
     *
     * Example:
     *
     * 'module.enable' => array(
     *      'exampleauth' => TRUE, // Setting to TRUE enables.
     *      'saml' => FALSE, // Setting to FALSE disables.
     *      'core' => NULL, // Unset or NULL uses default.
     * ),
     *
     */
    'module.enable' => array(
           'exampleauth' => TRUE, // Setting to TRUE enables.
           'saml' => TRUE, // Setting to FALSE disables.
           'core' => NULL, // Unset or NULL uses default.
    ),
```

```
    /*************************
     | SESSION CONFIGURATION |
     *************************/

    /*
     * This value is the duration of the session in seconds. Make sure that the time duration of
     * cookies both at the SP and the IdP exceeds this duration.
     */
    'session.duration' => 8 * (60 * 60), // 8 hours.

    /*
     * Sets the duration, in seconds, data should be stored in the datastore. As the data store is used
for
     * login and logout requests, this option will control the maximum time these operations can take.
     * The default is 4 hours (4*60*60) seconds, which should be more than enough for these operations.
     */
    'session.datastore.timeout' => (4 * 60 * 60), // 4 hours

    /*
     * Sets the duration, in seconds, auth state should be stored.
     */
    'session.state.timeout' => (60 * 60), // 1 hour

    /*
     * Option to override the default settings for the session cookie name
     */
    'session.cookie.name' => 'SimpleSAMLSessionID',

    /*
     * Expiration time for the session cookie, in seconds.
     *
     * Defaults to 0, which means that the cookie expires when the browser is closed.
     *
     * Example:
     *  'session.cookie.lifetime' => 30*60,
     */
    'session.cookie.lifetime' => 0,

    /*
     * Limit the path of the cookies.
     *
     * Can be used to limit the path of the cookies to a specific subdirectory.
     *
     * Example:
     *  'session.cookie.path' => '/simplesaml/',
     */
    'session.cookie.path' => '/',

    /*
     * Cookie domain.
     *
     * Can be used to make the session cookie available to several domains.
     *
     * Example:
     *  'session.cookie.domain' => '.example.org',
     */
    'session.cookie.domain' => null,

    /*
     * Set the secure flag in the cookie.
     *
     * Set this to TRUE if the user only accesses your service
     * through https. If the user can access the service through
     * both http and https, this must be set to FALSE.
     */
    'session.cookie.secure' => false,

    /*
     * Options to override the default settings for php sessions.
```

```
     */
    'session.phpsession.cookiename' => 'SimpleSAML',
    'session.phpsession.savepath' => null,
    'session.phpsession.httponly' => true,

    /*
     * Option to override the default settings for the auth token cookie
     */
    'session.authtoken.cookiename' => 'SimpleSAMLAuthToken',

    /*
     * Options for remember me feature for IdP sessions. Remember me feature
     * has to be also implemented in authentication source used.
     *
     * Option 'session.cookie.lifetime' should be set to zero (0), i.e. cookie
     * expires on browser session if remember me is not checked.
     *
     * Session duration ('session.duration' option) should be set according to
     * 'session.rememberme.lifetime' option.
     *
     * It's advised to use remember me feature with session checking function
     * defined with 'session.check_function' option.
     */
    'session.rememberme.enable' => false,
    'session.rememberme.checked' => false,
    'session.rememberme.lifetime' => (14 * 86400),

    /*
     * Custom function for session checking called on session init and loading.
     * See docs/simplesamlphp-advancedfeatures.txt for function code example.
     *
     * Example:
     *   'session.check_function' => array('sspmod_example_Util', 'checkSession'),
     */



    /*************************
     | MEMCACHE CONFIGURATION |
     *************************/

    /*
     * Configuration for the 'memcache' session store. This allows you to store
     * multiple redundant copies of sessions on different memcache servers.
     *
     * 'memcache_store.servers' is an array of server groups. Every data
     * item will be mirrored in every server group.
     *
     * Each server group is an array of servers. The data items will be
     * load-balanced between all servers in each server group.
     *
     * Each server is an array of parameters for the server. The following
     * options are available:
     *  - 'hostname': This is the hostname or ip address where the
     *    memcache server runs. This is the only required option.
     *  - 'port': This is the port number of the memcache server. If this
     *    option isn't set, then we will use the 'memcache.default_port'
     *    ini setting. This is 11211 by default.
     *  - 'weight': This sets the weight of this server in this server
     *    group. http://php.net/manual/en/function.Memcache-addServer.php
     *    contains more information about the weight option.
     *  - 'timeout': The timeout for this server. By default, the timeout
     *    is 3 seconds.
     *
     * Example of redundant configuration with load balancing:
     * This configuration makes it possible to lose both servers in the
     * a-group or both servers in the b-group without losing any sessions.
     * Note that sessions will be lost if one server is lost from both the
     * a-group and the b-group.
     *
     * 'memcache_store.servers' => array(
```

```
     *     array(
     *         array('hostname' => 'mc_a1'),
     *         array('hostname' => 'mc_a2'),
     *     ),
     *     array(
     *         array('hostname' => 'mc_b1'),
     *         array('hostname' => 'mc_b2'),
     *     ),
     * ),
     *
     * Example of simple configuration with only one memcache server,
     * running on the same computer as the web server:
     * Note that all sessions will be lost if the memcache server crashes.
     *
     * 'memcache_store.servers' => array(
     *     array(
     *         array('hostname' => 'localhost'),
     *     ),
     * ),
     *
     */
    'memcache_store.servers' => array(
        array(
            array('hostname' => 'localhost'),
        ),
    ),

    /*
     * This value allows you to set a prefix for memcache-keys. The default
     * for this value is 'simpleSAMLphp', which is fine in most cases.
     *
     * When running multiple instances of SSP on the same host, and more
     * than one instance is using memcache, you probably want to assign
     * a unique value per instance to this setting to avoid data collision.
     */
    'memcache_store.prefix' => null,

    /*
     * This value is the duration data should be stored in memcache. Data
     * will be dropped from the memcache servers when this time expires.
     * The time will be reset every time the data is written to the
     * memcache servers.
     *
     * This value should always be larger than the 'session.duration'
     * option. Not doing this may result in the session being deleted from
     * the memcache servers while it is still in use.
     *
     * Set this value to 0 if you don't want data to expire.
     *
     * Note: The oldest data will always be deleted if the memcache server
     * runs out of storage space.
     */
    'memcache_store.expires' => 36 * (60 * 60), // 36 hours.



    /*********************************
     | LANGUAGE AND INTERNATIONALIZATION |
     *********************************/

    /*
     * Languages available, RTL languages, and what language is the default.
     */
    'language.available' => array(
        'en', 'no', 'nn', 'se', 'da', 'de', 'sv', 'fi', 'es', 'fr', 'it', 'nl', 'lb', 'cs',
        'sl', 'lt', 'hr', 'hu', 'pl', 'pt', 'pt-br', 'tr', 'ja', 'zh', 'zh-tw', 'ru', 'et',
        'he', 'id', 'sr', 'lv', 'ro', 'eu', 'el', 'af'
    ),
    'language.rtl' => array('ar', 'dv', 'fa', 'ur', 'he'),
    'language.default' => 'en',
```

```
    /*
     * Options to override the default settings for the language parameter
     */
    'language.parameter.name' => 'language',
    'language.parameter.setcookie' => true,

    /*
     * Options to override the default settings for the language cookie
     */
    'language.cookie.name' => 'language',
    'language.cookie.domain' => null,
    'language.cookie.path' => '/',
    'language.cookie.secure' => false,
    'language.cookie.httponly' => false,
    'language.cookie.lifetime' => (60 * 60 * 24 * 900),

    /*
     * Which i18n backend to use.
     *
     * "SimpleSAMLphp" is the home made system, valid for 1.x.
     * For 2.x, only "gettext/gettext" will be possible.
     *
     * Home-made templates will always use "SimpleSAMLphp".
     * To use twig (where avaliable), select "gettext/gettext".
     */
    'language.i18n.backend' => 'SimpleSAMLphp',

    /**
     * Custom getLanguage function called from SimpleSAML\Locale\Language::getLanguage().
     * Function should return language code of one of the available languages or NULL.
     * See SimpleSAML\Locale\Language::getLanguage() source code for more info.
     *
     * This option can be used to implement a custom function for determining
     * the default language for the user.
     *
     * Example:
     *   'language.get_language_function' => array('sspmod_example_Template', 'getLanguage'),
     */

    /*
     * Extra dictionary for attribute names.
     * This can be used to define local attributes.
     *
     * The format of the parameter is a string with <module>:<dictionary>.
     *
     * Specifying this option will cause us to look for
modules/<module>/dictionaries/<dictionary>.definition.json
     * The dictionary should look something like:
     *
     * {
     *     "firstattribute": {
     *         "en": "English name",
     *         "no": "Norwegian name"
     *     },
     *     "secondattribute": {
     *         "en": "English name",
     *         "no": "Norwegian name"
     *     }
     * }
     *
     * Note that all attribute names in the dictionary must in lowercase.
     *
     * Example: 'attributes.extradictionary' => 'ourmodule:ourattributes',
     */
    'attributes.extradictionary' => null,



    /**************
     | APPEARANCE |
     **************/
```

```
/*
 * Which theme directory should be used?
 */
'theme.use' => 'default',

/*
 * Templating options
 *
 * By default, twig templates are not cached. To turn on template caching:
 * Set 'template.cache' to an absolute path pointing to a directory that
 * SimpleSAMLphp has read and write permissions to. Then, set
 * 'template.auto_reload' to false.
 *
 * When upgrading or changing themes, delete the contents of the cache.
 */
'template.auto_reload' => true,
//'template.cache' => '',


/*********************
 | DISCOVERY SERVICE |
 *********************/

/*
 * Whether the discovery service should allow the user to save his choice of IdP.
 */
'idpdisco.enableremember' => true,
'idpdisco.rememberchecked' => true,

/*
 * The disco service only accepts entities it knows.
 */
'idpdisco.validate' => true,

'idpdisco.extDiscoveryStorage' => null,

/*
 * IdP Discovery service look configuration.
 * Wether to display a list of idp or to display a dropdown box. For many IdP' a dropdown box
 * gives the best use experience.
 *
 * When using dropdown box a cookie is used to highlight the previously chosen IdP in the dropdown.
 * This makes it easier for the user to choose the IdP
 *
 * Options: [links,dropdown]
 */
'idpdisco.layout' => 'dropdown',



/***********************************
 | AUTHENTICATION PROCESSING FILTERS |
 ***********************************/

/*
 * Authentication processing filters that will be executed for all IdPs
 * Both Shibboleth and SAML 2.0
 */
'authproc.idp' => array(
    /* Enable the authproc filter below to add URN prefixes to all attributes
     10 => array(
         'class' => 'core:AttributeMap', 'addurnprefix'
     ), */
    /* Enable the authproc filter below to automatically generated eduPersonTargetedID.
    20 => 'core:TargetedID',
    */

    // Adopts language from attribute to use in UI
    30 => 'core:LanguageAdaptor',
```

```
        45 => array(
            'class'         => 'core:StatisticsWithAttribute',
            'attributename' => 'realm',
            'type'          => 'saml20-idp-SSO',
        ),

        /* When called without parameters, it will fallback to filter attributes ‹the old way›
         * by checking the 'attributes' parameter in metadata on IdP hosted and SP remote.
         */
        50 => 'core:AttributeLimit',

        /*
         * Search attribute "distinguishedName" for pattern and replaces if found

        60 => array(
            'class' => 'core:AttributeAlter',
            'pattern' => '/OU=studerende/',
            'replacement' => 'Student',
            'subject' => 'distinguishedName',
            '%replace',
        ),
         */

        /*
         * Consent module is enabled (with no permanent storage, using cookies).

        90 => array(
            'class' => 'consent:Consent',
            'store' => 'consent:Cookie',
            'focus' => 'yes',
            'checked' => TRUE
        ),
         */
        // If language is set in Consent module it will be added as an attribute.
        99 => 'core:LanguageAdaptor',
    ),

    /*
     * Authentication processing filters that will be executed for all SPs
     * Both Shibboleth and SAML 2.0
     */
    'authproc.sp' => array(
        /*
        10 => array(
            'class' => 'core:AttributeMap', 'removeurnprefix'
        ),
        */

        /*
         * Generate the 'group' attribute populated from other variables, including
eduPersonAffiliation.
         60 => array(
            'class' => 'core:GenerateGroups', 'eduPersonAffiliation'
        ),
        */
        /*
         * All users will be members of 'users' and 'members'
        61 => array(
            'class' => 'core:AttributeAdd', 'groups' => array('users', 'members')
        ),
        */

        // Adopts language from attribute to use in UI
        90 => 'core:LanguageAdaptor',

    ),



    /***************************
     | METADATA CONFIGURATION |
```

```
     *************************/
    'metadatadir' => '/conf/metadata',
    /*
     * This option configures the metadata sources. The metadata sources is given as an array with
     * different metadata sources. When searching for metadata, SimpleSAMLphp will search through
     * the array from start to end.
     *
     * Each element in the array is an associative array which configures the metadata source.
     * The type of the metadata source is given by the 'type' element. For each type we have
     * different configuration options.
     *
     * Flat file metadata handler:
     * - 'type': This is always 'flatfile'.
     * - 'directory': The directory we will load the metadata files from. The default value for
     *                this option is the value of the 'metadatadir' configuration option, or
     *                'metadata/' if that option is unset.
     *
     * XML metadata handler:
     * This metadata handler parses an XML file with either an EntityDescriptor element or an
     * EntitiesDescriptor element. The XML file may be stored locally, or (for debugging) on a remote
     * web server.
     * The XML metadata handler defines the following options:
     * - 'type': This is always 'xml'.
     * - 'file': Path to the XML file with the metadata.
     * - 'url': The URL to fetch metadata from. THIS IS ONLY FOR DEBUGGING - THERE IS NO CACHING OF THE
RESPONSE.
     *
     * MDQ metadata handler:
     * This metadata handler looks up for the metadata of an entity at the given MDQ server.
     * The MDQ metadata handler defines the following options:
     * - 'type': This is always 'mdq'.
     * - 'server': Base URL of the MDQ server. Mandatory.
     * - 'validateFingerprint': The fingerprint of the certificate used to sign the metadata. You don't
need this
     *                          option if you don't want to validate the signature on the metadata.
Optional.
     * - 'cachedir': Directory where metadata can be cached. Optional.
     * - 'cachelength': Maximum time metadata cah be cached, in seconds. Default to 24
     *                  hours (86400 seconds). Optional.
     *
     * PDO metadata handler:
     * This metadata handler looks up metadata of an entity stored in a database.
     *
     * Note: If you are using the PDO metadata handler, you must configure the database
     * options in this configuration file.
     *
     * The PDO metadata handler defines the following options:
     * - 'type': This is always 'pdo'.
     *
     * Examples:
     *
     * This example defines two flatfile sources. One is the default metadata directory, the other
     * is a metadata directory with auto-generated metadata files.
     *
     * 'metadata.sources' => array(
     *     array('type' => 'flatfile'),
     *     array('type' => 'flatfile', 'directory' => 'metadata-generated'),
     * ),
     *
     * This example defines a flatfile source and an XML source.
     * 'metadata.sources' => array(
     *     array('type' => 'flatfile'),
     *     array('type' => 'xml', 'file' => 'idp.example.org-idpMeta.xml'),
     * ),
     *
     * This example defines an mdq source.
     * 'metadata.sources' => array(
     *      array(
     *          'type' => 'mdq',
     *          'server' => 'http://mdq.server.com:8080',
     *          'cachedir' => '/var/simplesamlphp/mdq-cache',
```

```
     *          'cachelength' => 86400
     *      )
     * ),
     *
     * This example defines an pdo source.
     * 'metadata.sources' => array(
     *     array('type' => 'pdo')
     * ),
     *
     * Default:
     * 'metadata.sources' => array(
     *     array('type' => 'flatfile')
     * ),
     */

    /*
     * Should signing of generated metadata be enabled by default.
     *
     * Metadata signing can also be enabled for a individual SP or IdP by setting the
     * same option in the metadata for the SP or IdP.
     */
    'metadata.sign.enable' => false,

    /*
     * The default key & certificate which should be used to sign generated metadata. These
     * are files stored in the cert dir.
     * These values can be overridden by the options with the same names in the SP or
     * IdP metadata.
     *
     * If these aren't specified here or in the metadata for the SP or IdP, then
     * the 'certificate' and 'privatekey' option in the metadata will be used.
     * if those aren't set, signing of metadata will fail.
     */
    'metadata.sign.privatekey' => null,
    'metadata.sign.privatekey_pass' => null,
    'metadata.sign.certificate' => null,



    /**************************
     | DATA STORE CONFIGURATION |
     **************************/

    /*
     * Configure the data store for SimpleSAMLphp.
     *
     * - 'phpsession': Limited datastore, which uses the PHP session.
     * - 'memcache': Key-value datastore, based on memcache.
     * - 'sql': SQL datastore, using PDO.
     * - 'redis': Key-value datastore, based on redis.
     *
     * The default datastore is 'phpsession'.
     *
     * (This option replaces the old 'session.handler'-option.)
     */
    'store.type'                    => 'phpsession',

    /*
     * The DSN the sql datastore should connect to.
     *
     * See http://www.php.net/manual/en/pdo.drivers.php for the various
     * syntaxes.
     */
    'store.sql.dsn'                 => 'sqlite:/path/to/sqlitedatabase.sq3',

    /*
     * The username and password to use when connecting to the database.
     */
    'store.sql.username' => null,
    'store.sql.password' => null,
```

```
    /*
     * The prefix we should use on our tables.
     */
    'store.sql.prefix' => 'SimpleSAMLphp',

    /*
     * The hostname and port of the Redis datastore instance.
     */
    'store.redis.host' => 'localhost',
    'store.redis.port' => 6379,

    /*
     * The prefix we should use on our Redis datastore.
     */
    'store.redis.prefix' => 'SimpleSAMLphp',
);
```

A7. Dockerfile (customized, production ready)

```
# Use a base image with PHP 8.1 and Apache
FROM php:8.1-apache

# Set the Apache document root
ENV APACHE_DOCUMENT_ROOT /code/public

# Update and install dependencies
RUN apt-get update && apt-get install -y --no-install-recommends \
  apt-transport-https \
  ca-certificates \
  curl \
  gnupg \
  git \
  libmemcached-dev \
  libpng-dev \
  unzip \
  zlib1g-dev \
  libcurl4-openssl-dev \
  libssl-dev \
  libonig-dev \
  libicu-dev \
  cron \
  && apt-get clean \
  && rm -rf /var/lib/apt/lists/*

# Install PHP extensions
RUN docker-php-ext-install -j5 gd mbstring mysqli pdo pdo_mysql intl

# Install PECL extensions memcached and redis
RUN pecl install memcached && \
    pecl install redis && \
    docker-php-ext-enable memcached redis

# Manually install Xdebug
RUN curl -L https://xdebug.org/files/xdebug-3.1.6.tgz -o xdebug.tgz && \
    tar -xvzf xdebug.tgz && \
    rm xdebug.tgz && \
    cd xdebug-3.1.6 && \
    phpize && \
    ./configure && \
    make && \
    make install && \
    cd .. && \
```

```
    rm -rf xdebug-3.1.6 && \
    echo "zend_extension=$(find /usr/local/lib/php/extensions/ -name xdebug.so)" >
/usr/local/etc/php/conf.d/xdebug.ini

# Install Composer
RUN curl -sS https://getcomposer.org/installer | php -- --install-dir=/usr/local/bin --
filename=composer

# Configure Apache
RUN a2dismod mpm_event && a2enmod mpm_prefork

# Copy Apache configuration files
COPY apache2.conf /etc/apache2/
COPY mpm_prefork.conf /etc/apache2/mods-available/

# Copy the startup script
COPY startup.sh /

# Ensure the startup script has executable permissions
RUN chmod +x /startup.sh

# Set the default command to run the startup script
CMD ["/startup.sh"]

# Default command to start PHP server if ENTRYPOINT is overridden
CMD ["php", "-S", "0.0.0.0:8732", "-t", "/code/public"]
```

A8. Apache configuration (customized, internal to simplesamlphp)

```
Mutex file:/var/ default
PidFile /tmp/pidfile
Timeout 300
KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 5
User www-data
Group www-data
HostnameLookups Off
ErrorLogFormat "[%{u}t] [%l] %7F: %E: [client\ %a] %M% ,\ referer\ %{Referer}i"
ErrorLog /dev/stderr
LogLevel warn
ExtendedStatus on
IncludeOptional mods-enabled/*.load
IncludeOptional mods-enabled/*.conf
IncludeOptional conf-enabled/docker-php.conf
AddOutputFilterByType DEFLATE image/svg+xml


Listen 8732

<Directory />
    Options FollowSymLinks
    AllowOverride None
    Require all denied
</Directory>
AccessFileName .htaccess
<FilesMatch "^\.ht">
    Require all denied
```

```
</FilesMatch>
<Directory /code/public>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
LogFormat "%v:%p %h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" vhost_combined
LogFormat "%h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %O" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

# from conf-enabled/security.conf
ServerTokens OS
ServerSignature On
TraceEnable Off

<VirtualHost *:8732>
    ServerName https://${HOSTNAME}.ilc4clarin.ilc.cnr.it
    SetEnvIf Authorization "(.*)" HTTP_AUTHORIZATION=$1
    SetEnv HTTPS "on"
    UseCanonicalName on

    DocumentRoot /code/public

</VirtualHost>
```

A9. docker-compose.yml (customized)

```
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
      # - /etc/letsencrypt:/etc/letsencrypt
    ports:
      - '84:80'
      - '2443:443'
```

A10. NGINX configuration (customized):

```
events {
    worker_connections  1024;
}

http {
    #include       /etc/nginx/mime.types;

    server {
        listen              80;
        server_name         sp.ilc4clarin.ilc.cnr.it;
        server_tokens off;
        client_max_body_size 3000m;

        location / {
            proxy_set_header Host $host;
            proxy_set_header Upgrade $http_upgrade;
            proxy_pass  http://172.20.0.3:8732/;
        }
    }
}
```

A11. Apache httpd.conf

```
#
# This is the main Apache HTTP server configuration file.  It contains the
# configuration directives that give the server its instructions.
# See <URL:http://httpd.apache.org/docs/2.4/> for detailed information.
# In particular, see
# <URL:http://httpd.apache.org/docs/2.4/mod/directives.html>
# for a discussion of each configuration directive.
#
# Do NOT simply read the instructions in here without understanding
# what they do.  They're here only as hints or reminders.  If you are unsure
# consult the online docs. You have been warned.
#
# Configuration and logfile names: If the filenames you specify for many
# of the server's control files begin with "/" (or "drive:/" for Win32), the
# server will use that explicit path.  If the filenames do *not* begin
# with "/", the value of ServerRoot is prepended -- so "logs/access_log"
# with ServerRoot set to "/usr/local/apache2" will be interpreted by the
# server as "/usr/local/apache2/logs/access_log", whereas "/logs/access_log"
# will be interpreted as '/logs/access_log'.

#
# ServerRoot: The top of the directory tree under which the server's
# configuration, error, and log files are kept.
#
# Do not add a slash at the end of the directory path.  If you point
```

```
# ServerRoot at a non-local disk, be sure to specify a local disk on the
# Mutex directive, if file-based mutexes are used.  If you wish to share the
# same ServerRoot for multiple httpd daemons, you will need to change at
# least PidFile.
#
ServerRoot "/usr/local/apache2"

#
# Mutex: Allows you to set the mutex mechanism and mutex file directory
# for individual mutexes, or change the global defaults
#
# Uncomment and change the directory if mutexes are file-based and the default
# mutex file directory is not on a local disk or is not appropriate for some
# other reason.
#
# Mutex default:logs

#
# Listen: Allows you to bind Apache to specific IP addresses and/or
# ports, instead of the default. See also the <VirtualHost>
# directive.
#
# Change this to Listen on specific IP addresses as shown below to
# prevent Apache from glomming onto all bound IP addresses.
#
#Listen 12.34.56.78:80
Listen 80

#
# Dynamic Shared Object (DSO) Support
#
# To be able to use the functionality of a module which was built as a DSO you
# have to place corresponding `LoadModule' lines at this location so the
# directives contained in it are actually available _before_ they are used.
# Statically compiled modules (those listed by `httpd -l') do not need
# to be loaded here.
#
# Example:
# LoadModule foo_module modules/mod_foo.so
#
LoadModule mpm_event_module modules/mod_mpm_event.so
#LoadModule mpm_prefork_module modules/mod_mpm_prefork.so
#LoadModule mpm_worker_module modules/mod_mpm_worker.so
LoadModule authn_file_module modules/mod_authn_file.so
#LoadModule authn_dbm_module modules/mod_authn_dbm.so
#LoadModule authn_anon_module modules/mod_authn_anon.so
#LoadModule authn_dbd_module modules/mod_authn_dbd.so
#LoadModule authn_socache_module modules/mod_authn_socache.so
LoadModule authn_core_module modules/mod_authn_core.so
LoadModule authz_host_module modules/mod_authz_host.so
LoadModule authz_groupfile_module modules/mod_authz_groupfile.so
LoadModule authz_user_module modules/mod_authz_user.so
#LoadModule authz_dbm_module modules/mod_authz_dbm.so
#LoadModule authz_owner_module modules/mod_authz_owner.so
#LoadModule authz_dbd_module modules/mod_authz_dbd.so
LoadModule authz_core_module modules/mod_authz_core.so
#LoadModule authnz_ldap_module modules/mod_authnz_ldap.so
#LoadModule authnz_fcgi_module modules/mod_authnz_fcgi.so
LoadModule access_compat_module modules/mod_access_compat.so
LoadModule auth_basic_module modules/mod_auth_basic.so
#LoadModule auth_form_module modules/mod_auth_form.so
#LoadModule auth_digest_module modules/mod_auth_digest.so
#LoadModule allowmethods_module modules/mod_allowmethods.so
#LoadModule isapi_module modules/mod_isapi.so
#LoadModule file_cache_module modules/mod_file_cache.so
#LoadModule cache_module modules/mod_cache.so
#LoadModule cache_disk_module modules/mod_cache_disk.so
#LoadModule cache_socache_module modules/mod_cache_socache.so
LoadModule socache_shmcb_module modules/mod_socache_shmcb.so
#LoadModule socache_dbm_module modules/mod_socache_dbm.so
#LoadModule socache_memcache_module modules/mod_socache_memcache.so
```

```
#LoadModule socache_redis_module modules/mod_socache_redis.so
#LoadModule watchdog_module modules/mod_watchdog.so
#LoadModule macro_module modules/mod_macro.so
#LoadModule dbd_module modules/mod_dbd.so
#LoadModule bucketeer_module modules/mod_bucketeer.so
#LoadModule dumpio_module modules/mod_dumpio.so
#LoadModule echo_module modules/mod_echo.so
#LoadModule example_hooks_module modules/mod_example_hooks.so
#LoadModule case_filter_module modules/mod_case_filter.so
#LoadModule case_filter_in_module modules/mod_case_filter_in.so
#LoadModule example_ipc_module modules/mod_example_ipc.so
#LoadModule buffer_module modules/mod_buffer.so
#LoadModule data_module modules/mod_data.so
#LoadModule ratelimit_module modules/mod_ratelimit.so
LoadModule reqtimeout_module modules/mod_reqtimeout.so
#LoadModule ext_filter_module modules/mod_ext_filter.so
#LoadModule request_module modules/mod_request.so
#LoadModule include_module modules/mod_include.so
LoadModule filter_module modules/mod_filter.so
#LoadModule reflector_module modules/mod_reflector.so
#LoadModule substitute_module modules/mod_substitute.so
#LoadModule sed_module modules/mod_sed.so
#LoadModule charset_lite_module modules/mod_charset_lite.so
#LoadModule deflate_module modules/mod_deflate.so
#LoadModule xml2enc_module modules/mod_xml2enc.so
#LoadModule proxy_html_module modules/mod_proxy_html.so
#LoadModule brotli_module modules/mod_brotli.so
LoadModule mime_module modules/mod_mime.so
#LoadModule ldap_module modules/mod_ldap.so
LoadModule log_config_module modules/mod_log_config.so
#LoadModule log_debug_module modules/mod_log_debug.so
#LoadModule log_forensic_module modules/mod_log_forensic.so
#LoadModule logio_module modules/mod_logio.so
#LoadModule lua_module modules/mod_lua.so
LoadModule env_module modules/mod_env.so
#LoadModule mime_magic_module modules/mod_mime_magic.so
#LoadModule cern_meta_module modules/mod_cern_meta.so
#LoadModule expires_module modules/mod_expires.so
LoadModule headers_module modules/mod_headers.so
#LoadModule ident_module modules/mod_ident.so
#LoadModule usertrack_module modules/mod_usertrack.so
#LoadModule unique_id_module modules/mod_unique_id.so
LoadModule setenvif_module modules/mod_setenvif.so
LoadModule version_module modules/mod_version.so
#LoadModule remoteip_module modules/mod_remoteip.so
LoadModule proxy_module modules/mod_proxy.so
#LoadModule proxy_connect_module modules/mod_proxy_connect.so
#LoadModule proxy_ftp_module modules/mod_proxy_ftp.so
LoadModule proxy_http_module modules/mod_proxy_http.so
#LoadModule proxy_fcgi_module modules/mod_proxy_fcgi.so
#LoadModule proxy_scgi_module modules/mod_proxy_scgi.so
#LoadModule proxy_uwsgi_module modules/mod_proxy_uwsgi.so
#LoadModule proxy_fdpass_module modules/mod_proxy_fdpass.so
LoadModule proxy_wstunnel_module modules/mod_proxy_wstunnel.so
#LoadModule proxy_ajp_module modules/mod_proxy_ajp.so
#LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
#LoadModule proxy_express_module modules/mod_proxy_express.so
#LoadModule proxy_hcheck_module modules/mod_proxy_hcheck.so
#LoadModule session_module modules/mod_session.so
#LoadModule session_cookie_module modules/mod_session_cookie.so
#LoadModule session_crypto_module modules/mod_session_crypto.so
#LoadModule session_dbd_module modules/mod_session_dbd.so
#LoadModule slotmem_shm_module modules/mod_slotmem_shm.so
#LoadModule slotmem_plain_module modules/mod_slotmem_plain.so
LoadModule ssl_module modules/mod_ssl.so
#LoadModule optional_hook_export_module modules/mod_optional_hook_export.so
#LoadModule optional_hook_import_module modules/mod_optional_hook_import.so
#LoadModule optional_fn_import_module modules/mod_optional_fn_import.so
#LoadModule optional_fn_export_module modules/mod_optional_fn_export.so
#LoadModule dialup_module modules/mod_dialup.so
#LoadModule http2_module modules/mod_http2.so
```

```
#LoadModule proxy_http2_module modules/mod_proxy_http2.so
#LoadModule md_module modules/mod_md.so
#LoadModule lbmethod_byrequests_module modules/mod_lbmethod_byrequests.so
#LoadModule lbmethod_bytraffic_module modules/mod_lbmethod_bytraffic.so
#LoadModule lbmethod_bybusyness_module modules/mod_lbmethod_bybusyness.so
#LoadModule lbmethod_heartbeat_module modules/mod_lbmethod_heartbeat.so
LoadModule unixd_module modules/mod_unixd.so
#LoadModule heartbeat_module modules/mod_heartbeat.so
#LoadModule heartmonitor_module modules/mod_heartmonitor.so
#LoadModule dav_module modules/mod_dav.so
LoadModule status_module modules/mod_status.so
LoadModule autoindex_module modules/mod_autoindex.so
#LoadModule asis_module modules/mod_asis.so
#LoadModule info_module modules/mod_info.so
#LoadModule suexec_module modules/mod_suexec.so
<IfModule !mpm_prefork_module>
        #LoadModule cgid_module modules/mod_cgid.so
</IfModule>
<IfModule mpm_prefork_module>
        #LoadModule cgi_module modules/mod_cgi.so
</IfModule>
#LoadModule dav_fs_module modules/mod_dav_fs.so
#LoadModule dav_lock_module modules/mod_dav_lock.so
#LoadModule vhost_alias_module modules/mod_vhost_alias.so
#LoadModule negotiation_module modules/mod_negotiation.so
LoadModule dir_module modules/mod_dir.so
#LoadModule imagemap_module modules/mod_imagemap.so
#LoadModule actions_module modules/mod_actions.so
#LoadModule speling_module modules/mod_speling.so
#LoadModule userdir_module modules/mod_userdir.so
LoadModule alias_module modules/mod_alias.so
LoadModule rewrite_module modules/mod_rewrite.so

<IfModule unixd_module>
#
# If you wish httpd to run as a different user or group, you must run
# httpd as root initially and it will switch.
#
# User/Group: The name (or #number) of the user/group to run httpd as.
# It is usually good practice to create a dedicated user and group for
# running httpd, as with most system services.
#
User www-data
Group www-data

</IfModule>

# 'Main' server configuration
#
# The directives in this section set up the values used by the 'main'
# server, which responds to any requests that aren't handled by a
# <VirtualHost> definition.  These values also provide defaults for
# any <VirtualHost> containers you may define later in the file.
#
# All of these directives may appear inside <VirtualHost> containers,
# in which case these default settings will be overridden for the
# virtual host being defined.
#


#
# ServerAdmin: Your address, where problems with the server should be
# e-mailed.  This address appears on some server-generated pages, such
# as error documents.  e.g. admin@your-domain.com
#
ServerAdmin you@example.com

#
# ServerName gives the name and port that the server uses to identify itself.
# This can often be determined automatically, but we recommend you specify
# it explicitly to prevent problems during startup.
#
```

```
# If your host doesn't have a registered DNS name, enter its IP address here.
#
ServerName localhost:80

#
# Deny access to the entirety of your server's filesystem. You must
# explicitly permit access to web content directories in other
# <Directory> blocks below.
#
<Directory />
    AllowOverride none
    Require all denied
</Directory>

#
# Note that from this point forward you must specifically allow
# particular features to be enabled - so if something's not working as
# you might expect, make sure that you have specifically enabled it
# below.
#

#
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.
#
DocumentRoot "/usr/local/apache2/htdocs"
<Directory "/usr/local/apache2/htdocs">
    #
    # Possible values for the Options directive are "None", "All",
    # or any combination of:
    #   Indexes Includes FollowSymLinks SymLinksifOwnerMatch ExecCGI MultiViews
    #
    # Note that "MultiViews" must be named *explicitly* --- "Options All"
    # doesn't give it to you.
    #
    # The Options directive is both complicated and important.  Please see
    # http://httpd.apache.org/docs/2.4/mod/core.html#options
    # for more information.
    #
    Options Indexes FollowSymLinks

    #
    # AllowOverride controls what directives may be placed in .htaccess files.
    # It can be "All", "None", or any combination of the keywords:
    #   AllowOverride FileInfo AuthConfig Limit
    #
    AllowOverride None

    #
    # Controls who can get stuff from this server.
    #
    Require all granted
</Directory>

#
# DirectoryIndex: sets the file that Apache will serve if a directory
# is requested.
#
<IfModule dir_module>
    DirectoryIndex index.html
</IfModule>

#
# The following lines prevent .htaccess and .htpasswd files from being
# viewed by Web clients.
#
<Files ".ht*">
    Require all denied
</Files>
```

```
#
# ErrorLog: The location of the error log file.
# If you do not specify an ErrorLog directive within a <VirtualHost>
# container, error messages relating to that virtual host will be
# logged here.  If you *do* define an error logfile for a <VirtualHost>
# container, that host's errors will be logged there and not here.
#
ErrorLog /proc/self/fd/2

#
# LogLevel: Control the number of messages logged to the error_log.
# Possible values include: debug, info, notice, warn, error, crit,
# alert, emerg.
#
LogLevel warn

<IfModule log_config_module>
    #
    # The following directives define some format nicknames for use with
    # a CustomLog directive (see below).
    #
    LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
    LogFormat "%h %l %u %t \"%r\" %>s %b" common

    <IfModule logio_module>
      # You need to enable mod_logio.c to use %I and %O
      LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" %I %O" combinedio
    </IfModule>

    #
    # The location and format of the access logfile (Common Logfile Format).
    # If you do not define any access logfiles within a <VirtualHost>
    # container, they will be logged here.  Contrariwise, if you *do*
    # define per-<VirtualHost> access logfiles, transactions will be
    # logged therein and *not* in this file.
    #
    CustomLog /proc/self/fd/1 common

    #
    # If you prefer a logfile with access, agent, and referer information
    # (Combined Logfile Format) you can use the following directive.
    #
    #CustomLog "logs/access_log" combined
</IfModule>

<IfModule alias_module>
    #
    # Redirect: Allows you to tell clients about documents that used to
    # exist in your server's namespace, but do not anymore. The client
    # will make a new request for the document at its new location.
    # Example:
    # Redirect permanent /foo http://www.example.com/bar

    #
    # Alias: Maps web paths into filesystem paths and is used to
    # access content that does not live under the DocumentRoot.
    # Example:
    # Alias /webpath /full/filesystem/path
    #
    # If you include a trailing / on /webpath then the server will
    # require it to be present in the URL.  You will also likely
    # need to provide a <Directory> section to allow access to
    # the filesystem path.

    #
    # ScriptAlias: This controls which directories contain server scripts.
    # ScriptAliases are essentially the same as Aliases, except that
    # documents in the target directory are treated as applications and
    # run by the server when requested rather than as documents sent to the
    # client.  The same rules about trailing "/" apply to ScriptAlias
    # directives as to Alias.
```

```
    #
    ScriptAlias /cgi-bin/ "/usr/local/apache2/cgi-bin/"

</IfModule>

<IfModule cgid_module>
    #
    # ScriptSock: On threaded servers, designate the path to the UNIX
    # socket used to communicate with the CGI daemon of mod_cgid.
    #
    #Scriptsock cgisock
</IfModule>

#
# "/usr/local/apache2/cgi-bin" should be changed to whatever your ScriptAliased
# CGI directory exists, if you have that configured.
#
<Directory "/usr/local/apache2/cgi-bin">
    AllowOverride None
    Options None
    Require all granted
</Directory>

<IfModule headers_module>
    #
    # Avoid passing HTTP_PROXY environment to CGI's on this or any proxied
    # backend servers which have lingering "httpoxy" defects.
    # 'Proxy' request header is undefined by the IETF, not listed by IANA
    #
    RequestHeader unset Proxy early
</IfModule>

<IfModule mime_module>
    #
    # TypesConfig points to the file containing the list of mappings from
    # filename extension to MIME-type.
    #
    TypesConfig conf/mime.types

    #
    # AddType allows you to add to or override the MIME configuration
    # file specified in TypesConfig for specific file types.
    #
    #AddType application/x-gzip .tgz
    #
    # AddEncoding allows you to have certain browsers uncompress
    # information on the fly. Note: Not all browsers support this.
    #
    #AddEncoding x-compress .Z
    #AddEncoding x-gzip .gz .tgz
    #
    # If the AddEncoding directives above are commented-out, then you
    # probably should define those extensions to indicate media types:
    #
    AddType application/x-compress .Z
    AddType application/x-gzip .gz .tgz

    #
    # AddHandler allows you to map certain file extensions to "handlers":
    # actions unrelated to filetype. These can be either built into the server
    # or added with the Action directive (see below)
    #
    # To use CGI scripts outside of ScriptAliased directories:
    # (You will also need to add "ExecCGI" to the "Options" directive.)
    #
    #AddHandler cgi-script .cgi

    # For type maps (negotiated resources):
    #AddHandler type-map var

    #
```

```
    # Filters allow you to process content before it is sent to the client.
    #
    # To parse .shtml files for server-side includes (SSI):
    # (You will also need to add "Includes" to the "Options" directive.)
    #
    #AddType text/html .shtml
    #AddOutputFilter INCLUDES .shtml
</IfModule>

#
# The mod_mime_magic module allows the server to use various hints from the
# contents of the file itself to determine its type.  The MIMEMagicFile
# directive tells the module where the hint definitions are located.
#
#MIMEMagicFile conf/magic

#
# Customizable error responses come in three flavors:
# 1) plain text 2) local redirects 3) external redirects
#
# Some examples:
#ErrorDocument 500 "The server made a boo boo."
#ErrorDocument 404 /missing.html
#ErrorDocument 404 "/cgi-bin/missing_handler.pl"
#ErrorDocument 402 http://www.example.com/subscription_info.html
#

#
# MaxRanges: Maximum number of Ranges in a request before
# returning the entire resource, or one of the special
# values 'default', 'none' or 'unlimited'.
# Default setting is to accept 200 Ranges.
#MaxRanges unlimited

#
# EnableMMAP and EnableSendfile: On systems that support it,
# memory-mapping or the sendfile syscall may be used to deliver
# files.  This usually improves server performance, but must
# be turned off when serving from networked-mounted
# filesystems or if support for these functions is otherwise
# broken on your system.
# Defaults: EnableMMAP On, EnableSendfile Off
#
#EnableMMAP off
#EnableSendfile on

# Supplemental configuration
#
# The configuration files in the conf/extra/ directory can be
# included to add extra features or to modify the default configuration of
# the server, or you may simply copy their contents here and change as
# necessary.

# Server-pool management (MPM specific)
#Include conf/extra/httpd-mpm.conf

# Multi-language error messages
#Include conf/extra/httpd-multilang-errordoc.conf

# Fancy directory listings
#Include conf/extra/httpd-autoindex.conf

# Language settings
#Include conf/extra/httpd-languages.conf

# User home directories
#Include conf/extra/httpd-userdir.conf

# Real-time info on requests and configuration
#Include conf/extra/httpd-info.conf
```

```
# Virtual hosts
Include conf/extra/httpd-vhosts.conf
Include conf/extra/h2iosc-unibol.conf
Include conf/extra/h2iosc-ilc.conf
Include conf/extra/h2iosc-sp.conf
#Include conf/extra/h2iosc-escriptorium.conf
# Local access to the Apache HTTP Server Manual
#Include conf/extra/httpd-manual.conf

# Distributed authoring and versioning (WebDAV)
#Include conf/extra/httpd-dav.conf

# Various default settings
#Include conf/extra/httpd-default.conf

# Configure mod_proxy_html to understand HTML4/XHTML1
<IfModule proxy_html_module>
Include conf/extra/proxy-html.conf
</IfModule>

# Secure (SSL/TLS) connections
Include conf/extra/httpd-ssl.conf
#
# Note: The following must must be present to support
#       starting without SSL on platforms with no /dev/random equivalent
#       but a statically compiled-in mod_ssl.
#
<IfModule ssl_module>
SSLRandomSeed startup builtin
SSLRandomSeed connect builtin
</IfModule>
```

A12. Apache h2iosc-sp.conf

```
<VirtualHost *:80>
    ServerName sp.ilc4clarin.ilc.cnr.it
    Redirect permanent / https://sp.ilc4clarin.ilc.cnr.it/
</VirtualHost>

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

    SSLEngine on
    SSLCertificateFile /etc/letsencrypt/live/sp.ilc4clarin.ilc.cnr.it/fullchain.pem
    SSLCertificateKeyFile /etc/letsencrypt/live/sp.ilc4clarin.ilc.cnr.it/privkey.pem

    <Location "/">
       ProxyPass http://172.20.0.4:80/
       ProxyPassReverse http://172.20.0.4:80/

       ProxyPreserveHost On
    </Location>
</VirtualHost>
```

A13. Metarefresh configuration file

```
<?php

$config = [
    // Optional global blacklist for all sets
    #'blacklist' = array(
    #    'http://my.own.uni/idp'
    #),

    // Use Conditional GET to avoid re-downloading unchanged metadata (requires writable data dir)
    #'conditionalGET' => true,

    'sets' => [

        // DFN federation metadata
        'dfn' => [
            'cron' => ['hourly'], // Schedule the metadata refresh every hour
            'sources' => [
                [
                    // Optional blacklist or whitelist specific to this source
                    #'blacklist' => array('http://some.other.uni/idp'),
                    #'whitelist' => array('http://some.uni/idp'),

                    // Optional conditional GET setting for this source
                    #'conditionalGET' => true,

                    'src' => 'http://www.aai.dfn.de/metadata/dfn-aai-edugain+idp-metadata.xml',
                    'certificates' => ['dfn-aai.pem'], // Signature validation certificates

                    'template' => [
                        'tags' => ['dfn'], // Tag assigned to imported entities
                        // Optional authproc settings
                        /* 'authproc' => [
                            51 => ['class' => 'core:AttributeMap', 'oid2name'],
                        ], */
                    ],

                    // Regex-based configuration overrides per entityID (optional)
                    /* 'regex-template' => [
                        "#^https://www\\.example\\.com/sp$#" => [
                            'assertion.encryption' => false,
                        ],
                    ], */

                    // Optionally restrict to specific metadata types
                    //'types' => [],
                ],
            ],

            'expireAfter' => 34560060, // Maximum 4 days validity (in seconds)
            'outputDir' => '/conf/metadata/dfn/', // Where the parsed metadata will be saved
            'outputFormat' => 'flatfile', // Supported: 'flatfile' or 'serialize'
            'types' => ['saml20-idp-remote'], // Limit to IdPs only
        ],

        // CLARIN SPF IdPs
        'clarin' => [
            'cron' => ['hourly'],
            'sources' => [
                [
                    'src' => 'https://infra.clarin.eu/aai/prod_md_about_spf_idps.xml',
                    // Certificate validation not enforced here
                    'template' => [
                        'tags' => ['clarin'],
                    ],
                ],
            ],
            'expireAfter' => 34560060,
            'outputDir' => '/conf/metadata/clarin/',
```

```
            'outputFormat' => 'flatfile',
            'types' => ['saml20-idp-remote'],
        ],

        // CLARIN ERIC IdPs
        'clarin-idp' => [
            'cron' => ['hourly'],
            'sources' => [
                [
                    'src' => 'https://infra.clarin.eu/aai/prod_md_about_clarin_erics_idp.xml',
                    'template' => [
                        'tags' => ['clarin-idp'],
                    ],
                ],
            ],
            'expireAfter' => 34560060,
            'outputDir' => '/conf/metadata/clarin-idp/',
            'outputFormat' => 'flatfile',
            'types' => ['saml20-idp-remote'],
        ],
    ],
];
```

A14. Footer template

```
<footer id="footer">
  <div class="wrap">
    <div class="center copyrights">© 2007-{{ year }} <a
href="https://simplesamlphp.org/">SimpleSAMLphp</a>
    <br>
    Hosted by <a href="https://cloud.garr.it/" target="_blank">Cloud GARR</a>
    </div>
    <div class="logo-footer-space show-for-large">
      <div class="logo-footer">
        <img class="pure-img" src="{{ asset("icons/ssplogo-fish-small.png") }}" alt="Small fish logo">
      </div>
    </div>
  </div>
</footer>
```

A15. Privacy Page

```
<!DOCTYPE html>
<html lang="it">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Privacy Policy - ILC4CLARIN Service Provider</title>
    <style>
      /* ----------------------------------------------------------------
       Esempio minimale di Bootstrap CSS (solo per layout e utility)
       ---------------------------------------------------------------- */
      *,
      *::before,
      *::after {
```

```
      box-sizing: border-box;
    }
    body {
      margin: 0;
      font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto,
        "Helvetica Neue", Arial, sans-serif;
      line-height: 1.5;
      background-color: #f8f9fa;
      color: #212529;
    }
    .container {
      width: 100%;
      padding-right: 1.5rem;
      padding-left: 1.5rem;
      margin-right: auto;
      margin-left: auto;
    }
    .row {
      display: flex;
      flex-wrap: wrap;
      margin-right: -0.75rem;
      margin-left: -0.75rem;
    }
    /* Per centrare la colonna */
    .justify-content-center {
      justify-content: center;
    }
    .col-6 {
      position: relative;
      width: 100%;
      max-width: 50%;
      padding-right: 0.75rem;
      padding-left: 0.75rem;
    }
    .text-center {
      text-align: center !important;
    }
    .mt-4 {
      margin-top: 1.5rem !important;
    }
    .mb-3 {
      margin-bottom: 1rem !important;
    }
    img {
      max-width: 100%;
      height: auto;
    }
    address {
      font-style: normal;
      line-height: 1.6;
    }
    h1,
    h3 {
      color: #343a40;
    }
    /* ----------------------------------------------------------------
     Fine Bootstrap CSS minimale
     ---------------------------------------------------------------- */
  </style>
</head>
<body>
  <div class="container">
    <div class="row justify-content-center">
      <div class="col-6">
        <!-- Titolo principale -->
        <h1 class="text-center mt-4">Privacy Policy</h1>

        <!-- Sezione immagini: CLARIN e H2IOSC -->
        <div class="text-center mb-4">
          <a href="https://ilc4clarin.ilc.cnr.it/" target="_blank" rel="noopener noreferrer">
            <img
```

```
                src="img/ilc4clarin.png"
                alt="CLARIN Logo"
                style="max-height: 100px; margin-right: 20px"
              />
            </a>
            <a href="https://www.h2iosc.cnr.it/" target="_blank" rel="noopener noreferrer">
              <img
                src="img/h2iosc.png"
                alt="H2IOSC Logo"
                style="max-height: 100px"
              />
            </a>
          </div>

          <!-- Sezioni della privacy policy -->
          <section class="mb-3">
            <h3>Name of the service</h3>
            <p>ILC4CLARIN Service Provider</p>
          </section>

          <section class="mb-3">
            <h3>Description of the service</h3>
            <p>
              The ILC4CLARIN Service Provider enables federated authentication within the
<strong>CLARIN Service Provider Federation</strong>, allowing seamless and secure access to digital
resources. Built on <strong>SimpleSAMLphp</strong>, this system ensures compliance with identity
federation standards, providing Single Sign-On (SSO) capabilities for applications of the
<strong>Institute for Computational Linguistics "A. Zampolli"</strong>. <br> <br>
              By leveraging this authentication framework, users can securely access various services
while maintaining a unified and efficient login experience.
            </p>
          </section>

          <section class="mb-3">
            <h3>Data controller and contact person</h3>
            <address>
              Institute for Computational Linguistics "Antonio Zampolli"<br />
              c/o Area della Ricerca di Pisa<br />
              Via G. Moruzzi, 1<br />
              56124 Pisa<br />
              Italy<br />
              <strong>Phone:</strong> +39 050 315 8379<br />
              <strong>Fax:</strong> +39 050 315 2839
            </address>
          </section>

          <section class="mb-3">
            <h3>Technical contact</h3>
            <p>
              Michele Mallia<br />
              michele.mallia@ilc.cnr.it<br />
              michele.mallia@cnr.it
            </p>
          </section>

          <section class="mb-3">
            <h3>Jurisdiction</h3>
            <p>IT, Italy</p>
          </section>

          <section class="mb-3">
            <h3>Personal Data Processed</h3>
            <p>
              The following personal information is fetched from the Identity Provider server
              of your home organisation every time you log in to the service:
            </p>
            <ul>
              <li><strong>urn:oid:1.3.6.1.4.1.5923.1.1.1.6</strong> (eduPersonPrincipalName)</li>
              <li><strong>urn:oid:0.9.2342.19200300.100.1.3</strong> (email)</li>
              <li><strong>urn:oid:2.16.840.1.113730.3.1.241</strong> (displayName)</li>
              <li><strong>urn:oid:1.3.6.1.4.1.5923.1.1.1.10</strong> (eduPersonTargetedID)</li>
```

```
              <li><strong>urn:oid:1.3.6.1.4.1.5923.1.1.1.9</strong> (eduPersonScopedAffiliation)</li>
            </ul>

            <p>If not all of the attributes above are released by the user's Identity Provider the user
might be asked to provide these fields herself.<br />
             Log files are created at various levels which include:</p>
             <ul>
                <li><strong>Personal data information</strong>: metadata, IP addresses, User-Agent</li>
                <li><strong>SAML Requests information</strong>: SAML assertions, user's attributes,
Issuer, Destination, AuthnRequests</li>
                <li><strong>System's Information</strong>: SAML assertions, user's attributes, Issuer,
Destination, AuthnRequests</li>
             </ul>
          </section>

          <section class="mb-3">
            <h3>Purpose of the Processing of Personal Data</h3>
            <p>
              The personal data collected during authentication is processed for the following
purposes:
            </p>
            <ul>
              <li><strong>Authentication and Access Control:</strong> To verify user identity and grant
access to services.</li>
              <li><strong>Single Sign-On (SSO) Functionality:</strong> To enable seamless login across
multiple federated services.</li>
              <li><strong>Security and Fraud Prevention:</strong> To detect unauthorized access
attempts and enhance system security.</li>
              <li><strong>Logging and Auditing:</strong> To maintain security logs and comply with
legal or institutional requirements.</li>
              <li><strong>Personalized User Experience:</strong> To tailor service access based on user
roles and affiliations.</li>
              <li><strong>Federation Compliance:</strong> To ensure compatibility with the CLARIN
Service Provider Federation and adhere to its policies.</li>
              <li><strong>Service Improvement and Monitoring:</strong> To analyze authentication trends
and enhance system performance.</li>
              <li><strong>Incident Response and Troubleshooting:</strong> To diagnose technical issues
and support users in case of authentication failures.</li>
            </ul>
          </section>


          <section class="mb-3">
            <h3>Third parties to whom personal data is disclosed</h3>
            <p>The personal data are not disclosed to anyone outside of the ILC-CNR for CLARIN-IT
team.</p>
          </section>

          <section class="mb-3">
            <h3>How to access, rectify and delete the personal data</h3>
            <p>
                The personal information is visible in your profile. If there is no profile, no
personal data is directly stored. Use the technical contact above for any requests. To rectify the data
released by your Home Organisation, contact your Home Organisation's IT helpdesk.
            </p>
          </section>

          <section class="mb-3">
            <h3>Data retention</h3>
            <p>Personal data is deleted on request of the user or if the user hasn't used the service
for five years.</p>
          </section>

          <section class="mb-3">
            <h3>Data Protection Code of Conduct</h3>
            <p>Your personal data will be protected according to the <a
href="https://geant3plus.archive.geant.net/Pages/uri/V1.html" target="_blank">Code of Conduct for
Service Providers</a>, a common standard for the research and higher education sector to protect your
privacy.</p>
          </section>
        </div>
```

```
      </div>
    </div>
    <footer class="text-center mt-4" style="background-color: #f8f9fa; padding: 20px; font-size:
14px;">
      <p>
        © 2025 ILC4CLARIN Service Provider | Institute for Computational Linguistics "A. Zampolli" <br>
        Hosted by Cloud GARR
      </p>
      <p>
        For inquiries, contact:
        <a href="mailto:dspace-clarin-it-ilc-help@ilc.cnr.it">dspace-clarin-it-ilc-help@ilc.cnr.it</a>
      </p>
    </footer>
  </body>
</html>
```

A16. SP Metadata

```
<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
    xmlns:mdattr="urn:oasis:names:tc:SAML:metadata:attribute"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:mdrpi="urn:oasis:names:tc:SAML:metadata:rpi"
    xmlns:mdui="urn:oasis:names:tc:SAML:metadata:ui"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#" entityID="https://sp.ilc4clarin.ilc.cnr.it">
    <md:Extensions>
        <mdattr:EntityAttributes xmlns:mdattr="urn:oasis:names:tc:SAML:metadata:attribute">
            <saml:Attribute xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
Name="http://macedir.org/entity-category" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
                <saml:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">http://www.geant.net/uri/dataprotection-code-of-conduct/v1</saml:AttributeValue>
                <saml:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">http://refeds.org/category/research-and-scholarship</saml:AttributeValue>
                <saml:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">http://clarin.eu/category/clarin-member</saml:AttributeValue>
            </saml:Attribute>
        </mdattr:EntityAttributes>
        <mdrpi:RegistrationInfo xmlns:mdrpi="urn:oasis:names:tc:SAML:metadata:rpi"
registrationAuthority="urn:mace:sp.ilc4clarin.ilc.cnr.it" registrationInstant="2025-02-10T14:01:00Z">
            <mdrpi:RegistrationPolicy
xml:lang="en">https://sp.ilc4clarin.ilc.cnr.it/privacy.html</mdrpi:RegistrationPolicy>
        </mdrpi:RegistrationInfo>
    </md:Extensions>
    <md:SPSSODescriptor protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
        <md:Extensions>
            <mdui:UIInfo xmlns:mdui="urn:oasis:names:tc:SAML:metadata:ui">
                <mdui:DisplayName xml:lang="en">Service Provider for the CLARIN Research Infrastructure
provided by ILC-CNR</mdui:DisplayName>
                <mdui:DisplayName xml:lang="it">Service Provider per l'infrastruttura di Ricerca CLARIN
erogato da ILC-CNR</mdui:DisplayName>
                <mdui:Description xml:lang="en">ILC-CNR for the CLARIN-IT consortium: service provider
for federated authentication</mdui:Description>
                <mdui:Description xml:lang="it">ILC-CNR per il consorzio CLARIN-IT: service provider
per l'autenticazione federata</mdui:Description>
                <mdui:InformationURL
xml:lang="en">https://sp.ilc4clarin.ilc.cnr.it/privacy.html</mdui:InformationURL>
                <mdui:PrivacyStatementURL
xml:lang="en">https://sp.ilc4clarin.ilc.cnr.it/privacy.html</mdui:PrivacyStatementURL>
                <mdui:PrivacyStatementURL
xml:lang="it">https://sp.ilc4clarin.ilc.cnr.it/privacy.html</mdui:PrivacyStatementURL>
                <mdui:Logo width="80"
height="53">https://sp.ilc4clarin.ilc.cnr.it/img/ilc4clarin_80x53.png</mdui:Logo>
```

```
                <mdui:Logo width="16"
height="16">https://sp.ilc4clarin.ilc.cnr.it/img/ilc4clarin_16x16.png</mdui:Logo>
            </mdui:UIInfo>
        </md:Extensions>
        <md:KeyDescriptor use="signing">
            <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <ds:X509Data>
                    <ds:X509Certificate>MIIFwzCC...</ds:X509Certificate>
                </ds:X509Data>
            </ds:KeyInfo>
        </md:KeyDescriptor>
        <md:KeyDescriptor use="encryption">
            <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <ds:X509Data>
                    <ds:X509Certificate>MIIFwz... </ds:X509Certificate>
                </ds:X509Data>
            </ds:KeyInfo>
        </md:KeyDescriptor>
        <md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
Location="https://sp.ilc4clarin.ilc.cnr.it/module.php/saml/sp/saml2-logout.php/default-sp"/>
        <md:AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="https://sp.ilc4clarin.ilc.cnr.it/module.php/saml/sp/saml2-acs.php/default-sp" index="0"/>
        <md:AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
Location="https://sp.ilc4clarin.ilc.cnr.it/module.php/saml/sp/saml2-acs.php/default-sp" index="1"/>
        <md:AttributeConsumingService index="0">
            <md:ServiceName xml:lang="it">Service Provider per l'infrastruttura di Ricerca CLARIN
erogato da ILC-CNR</md:ServiceName>
            <md:ServiceName xml:lang="en">Service Provider for the CLARIN Research Infrastructure
provided by ILC-CNR</md:ServiceName>
            <md:ServiceDescription xml:lang="en">ILC-CNR for the CLARIN-IT consortium: service provider
for federated authentication</md:ServiceDescription>
            <md:ServiceDescription xml:lang="it">ILC-CNR per il consorzio CLARIN-IT: service provider
per l'autenticazione federata</md:ServiceDescription>
            <md:RequestedAttribute Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" FriendlyName="eduPersonPrincipalName"
isRequired="true"/>
            <md:RequestedAttribute Name="urn:oid:0.9.2342.19200300.100.1.3"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" FriendlyName="email" isRequired="true"/>
            <md:RequestedAttribute Name="urn:oid:2.16.840.1.113730.3.1.241"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" FriendlyName="displayName"
isRequired="true"/>
            <md:RequestedAttribute Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.10"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" FriendlyName="eduPersonTargetedID"
isRequired="true"/>
            <md:RequestedAttribute Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.9"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" FriendlyName="eduPersonScopedAffiliation"
isRequired="true"/>
        </md:AttributeConsumingService>
    </md:SPSSODescriptor>
    <md:Organization>
        <md:OrganizationName xml:lang="en">National Research Council (CNR)</md:OrganizationName>
        <md:OrganizationName xml:lang="it">Consiglio Nazionale delle Ricerche
(CNR)</md:OrganizationName>
        <md:OrganizationDisplayName xml:lang="it">CNR Istituto di Linguistica Computazionale "Antonio
Zampolli"</md:OrganizationDisplayName>
        <md:OrganizationDisplayName xml:lang="en">CNR Institute for Computational Linguistics "Antonio
Zampolli"</md:OrganizationDisplayName>
        <md:OrganizationURL xml:lang="it">http://www.ilc.cnr.it</md:OrganizationURL>
        <md:OrganizationURL xml:lang="en">http://www.ilc.cnr.it/en</md:OrganizationURL>
    </md:Organization>
    <md:ContactPerson contactType="administrative">
        <md:Company>Consiglio Nazionale delle Ricerche</md:Company>
        <md:GivenName>Michele</md:GivenName>
        <md:SurName>Mallia</md:SurName>
        <md:EmailAddress>mailto:michele.mallia@cnr.it</md:EmailAddress>
        <md:TelephoneNumber>(+39)3392804180</md:TelephoneNumber>
    </md:ContactPerson>
    <md:ContactPerson contactType="technical">
        <md:Company>Consiglio Nazionale delle Ricerche</md:Company>
        <md:GivenName>Michele</md:GivenName>
        <md:SurName>Mallia</md:SurName>
```

```
        <md:EmailAddress>mailto:michele.mallia@cnr.it</md:EmailAddress>
        <md:TelephoneNumber>(+39)3392804180</md:TelephoneNumber>
    </md:ContactPerson>
    <md:ContactPerson contactType="support">
        <md:Company>Consiglio Nazionale delle Ricerche</md:Company>
        <md:GivenName>Michele</md:GivenName>
        <md:SurName>Mallia</md:SurName>
        <md:EmailAddress>mailto:michele.mallia@cnr.it</md:EmailAddress>
        <md:TelephoneNumber>(+39)3392804180</md:TelephoneNumber>
    </md:ContactPerson>
    <md:ContactPerson contactType="technical">
        <md:GivenName>Administrator</md:GivenName>
        <md:EmailAddress>mailto:michele.mallia@cnr.it</md:EmailAddress>
    </md:ContactPerson>
</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*, (pp. 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*.