



SEAL

Student and Citizen Identities Linked

D6.1 Technical documentation on SEAL integration package for HEIs

Document Identification			
Status	Final	Due Date	31/05/2021
Version	1.0	Submission Date	03/06/2021

Related Activity	Act 6	Document Reference	D6.1
Related Deliverable(s)	D4.2 D3.3 D3.1	Dissemination Level (*)	PU
Lead Organization	UMA	Lead Author	Victoriano Giralt
Contributors	Francisco José Aragó (UJI) Nikos Triantafyllou (UAegean) Alberto Basurte (UMA) Raúl Ocaña (UMA) Laura Domínguez (UMA)	Reviewers	Petros Kavassalis UAEGEAN
			Valentini Paparrizou MINISTRY

Keywords:
Academic Online Services, HEI, Verifiable Credentials, Permissioned Blockchain, Self-sovereign Identity Technologies, eIdentity Management, Personal Attributes, Data Transfer, eIDAS Network, eIDAS eID, eduGAIN, eMRTD, ePassport, Cloud, Distributed Storage, EWP, Academic IT Systems, mobile apps.

This document is issued within the frame and for the purpose of the SEAL project. This project has received funding from the European Union's Innovation and Networks Executive Agency – Connecting Europe Facility (CEF) under Grant AGREEMENT No INEA/CEF/ICT/A2018/1633170; Action No 2018-EU-IA-0024. The opinions expressed, and arguments employed herein do not necessarily reflect the official views of the European Commission.

This document and its content are the property of the SEAL Consortium. All rights relevant to this document are determined by the applicable laws. Access to this document does not grant any right or license on the document or its contents. This document or its contents are not to be used or treated in any manner inconsistent with the rights or interests of the SEAL Consortium or the Partners detriment and are not to be disclosed externally without prior written consent from the SEAL Partners.

Each SEAL Partner may use this document in conformity with the SEAL Consortium Grant Agreement provisions.

(*) Dissemination level.-**PU**: Public, fully open, e.g. web; **CO**: Confidential, restricted under conditions set out in Model Grant Agreement; **CI**: Classified, **Int** = Internal Working Document, information as referred to in Commission Decision 2001/844/EC.

Document Information

List of Contributors	
Name	Partner
Francisco José Aragó	UJI
Nikos Triantafyllou	UAegean
Alberto Basurte	UMA
Raúl Ocaña	UMA
Laura Domínguez	UMA

Document History			
Version	Date	Change editors	Changes
0.1	25/05/2021	Francisco Aragó UJI	Added UJI contribution
0.2	28/05/2021	Valentini Paparrizou	Review of UJI Contribution
0.3	28/05/2021	Nikos Triantafyllou	Added SSI and VC contribution
0.4	28/05/2021	Victoriano Giralt, Alberto Basurte, Raúl Ocaña, Laura Domínguez	Added UMAcontribution
0.5	31/05/2021	Petros Kavassalis	Review
0.6	31/05/2021	Valentini Paparrizou	Review
1.0	03/06/2021	ATOS	FINAL VERSION TO BE SUBMITTED

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable leader	Victoriano Giralt (UMA)	03/06/2021
Technical manager	UJI	03/06/2021
Quality manager	ATOS	03/06/2021
Project Manager	ATOS	03/06/2021

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs			Page:	2 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1
				Status:	Final

Table of Contents

Document Information	2
Table of Contents	3
List of Tables.....	5
List of Figures	6
List of Acronyms.....	7
1. Executive Summary	9
2. Introduction.....	10
2.1 Purpose of the document.....	10
2.2 Context of this deliverable	10
2.3 Structure of the document	11
3. User level information.....	12
3.1 SEAL Service Dashboard	12
3.1.1 Web Dashboard.....	12
3.1.2 Mobile Application	14
3.2 Credentials issuing	20
3.2.1 Federated Credentials	20
3.2.2 Verifiable Credentials	24
4. HEI as Identity Provider.....	28
4.1 “SAML” (Direct IdP Connection to SEAL)	28
4.2 OIDC.....	28
4.3 “SSI credential issuing”	29
4.4 “Developing a generic data source”.....	29
5. HEI as Identity consumer.....	34
5.1 “SAML”	34
5.1.1 Available software for integration	34
5.1.2 General Guidelines	35
5.1.3 Connection setting	36
5.2 “OIDC”	36
5.2.1 Available Software for integration.....	36
5.2.2 General Guidelines	37

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs				Page:	3 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final

5.2.3	Connection setting	38
5.3	“SSI credential consumption”	39
6.	Conclusions.....	43
7.	References.....	44

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs				Page:	4 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final

List of Tables

Table 1: List of supported attributes and claims _____ 37

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs				Page:	5 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final

List of Figures

Figure 1: Mobile Dashboard main page	15
Figure 2: PDS options	15
Figure 3: Load PDS options	16
Figure 4: Load local PDS	16
Figure 5: Load Cloud options	17
Figure 6: PDS selection process	17
Figure 7: Store PDS in the local mobile device	17
Figure 8: SSI authentication functionality on mobile SEAL application	18
Figure 9: Authentication sources	19
Figure 10: SEAL eMRTD reader mobile application	19
Figure 11: Request Manager Source Selector	22
Figure 12: eIDAS Country Selector	23
Figure 13: Spanish eIDAS IDP	23
Figure 14: eduGAIN Metadata Discovery Service	24
Figure 15: Credential Graph and the Proof Graph	25
Figure 16: Wallet Connection Request	26
Figure 17: Authoritative Source Selection	26
Figure 18: Verifiable Credential Issuance Notification	27
Figure 19: Verifiable Credential stored on user's mobile wallet	27
Figure 20: Microservice functional diagram	30

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs				Page:	6 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final

List of Acronyms

Abbreviation / acronym	Description
API	Application Programming Interface
DLT	Distributed Ledger Technology
Dx.y	Deliverable number y belonging to WP x
EC	European Commission
eduGAIN	EDUcation Global Authentication INfrastructure
eID	Electronic Identification
eIDAS	Electronic IDentification, Authentication and trust Services
eMRTD	Electronic Machine Readable Travel Document
ePassport	Electronic Passport
ESMO	eIDAS-enabled Student Mobility
EWP	Erasmus Without Paper
HEI	Higher Education Institution
HMAC	Hash based message authentication code
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IdMS	Identity Management Server
IdP	Identity Provider
JSON	JavaScript Object Notation
LoA	Level of Assurance
MRZ	Machine-Readable Zone
NFC	Near-Field Communication
OCR	Optical character recognition
OIDC	OpenID Connect
OTP	One-time password

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs			Page:	7 of 44	
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final

Abbreviation / acronym	Description
PDS	SEAL Personal Data Storage
PDS	Personal Data Store
QR code	Quick Response code
RFID	Radio-Frequency IDentification
RP	Relying Party
SAML	Security Assertion Markup Language
SCHAC	SCHema for ACademia
SEAL OP	SEAL OpenID Connect Provider
SP	Service Provider
SSI	Self-Sovereign Identity
UI	User Interface
uPort	Distributed Ledger Technology based identity(now serto)
URL	Uniform Resource Locator
VC	Verifiable Credential
VCs	Verifiable Credentials
WP	Work Package
XML	Extensible Markup Language

1. Executive Summary

The present document introduces the functionalities SEAL offers to Higher Education Institutions and how they can be integrated in the institution identity ecosystem. These services allow for user level interaction on a personal basis for combining identities from diverse sources, mainly government issued eIDs and academic identities, and having them under personal control, thanks to the SEAL dashboard service managed either via a web browser or a mobile app.

Institution identity service administrators will find that SEAL offers options for them to provide identities through eduGAIN, based on the institutional identity repository, thus allowing their members to incorporate academic attributes to the identities they manage via the SEAL dashboard.

Institutional application providers are able to authenticate and authorise users by consuming SEAL based identities via federation protocols such as SAML or OIDC, either by directly configuring the SEAL Identity Provider as a trusted source or, once it is incorporated to the metadata feeds, through eduGAIN.

These SEAL identities can provide rich attribute sets by combining diverse trusted and verified sources, thus allowing services to have a higher level of trust on the information they are receiving.

The document will also describe how the identity sources can be augmented by the institutional identity providers or other entities that decide to operate their own instance of services based on SEAL software developments.

Also, thanks to SEAL Verified Credentials derivation, it is possible for persons using the services to generate elements that can be used as Self-Sovereign Identity assertions with information elements extracted from their linked identities, increasing control on personal information and, therefore, privacy.

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs				Page:	9 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final

2. Introduction

2.1 Purpose of the document

This deliverable “Technical documentation on SEAL integration package for HEIs” falls under Milestone 17 “HEI integration package” related to Task 6.1 of project SEAL “Technical packaging of outcomes from technical activities”.

This document:

1. Will introduce to persons responsible for identity management in HEIs the tools that SEAL services offers for their users to manage their diverse identities.
2. Will introduce HEI identity administrators to the functions of the SEAL Identity Provider.
3. Will inform HEI identity administrators about how they can allow their constituency to obtain academic identity information to add to their SEAL linked identities.
4. Will introduce HEI application administrations and developers to the options that SEAL Identity Provider offers for obtaining identities and attributes about their users.

2.2 Context of this deliverable

The Erasmus+ and Connecting Europe Facility (CEF) programmes have put forward several projects that rely on trustworthy digital identities. SEAL ability to link identities from diverse sources and generate new ones derived from data gathered from the former, can provide an extra level of trust to identities and attributes used in projects, like ESMO, EWP, ESC, MyAcademicID or EDSSI. The eIDAS regulation and infrastructure, together with the eduGAIN federation are the two main sources of identity. The EBSI will, most probably, become the support for the SSI credentials that can be used in and derived from SEAL service. The new Erasmus guidelines clearly call for a full digitalisation of the processes that will need the kind of identities that SEAL can provide.

The Erasmus deadline states that all participating HEIs should be exchanging data electronically with the following schedule:

- All HEIs need to be connected to the EWP network as soon as possible before June 2021 for managing Learning Agreements and renewing Inter Institutional Agreements.
- 2021/2022 academic term is a transitional one, so the previous requirement will be compulsory for 2022/2023 academic term.
- All LAs for student mobilities will have to be managed on-line (OLA) over the EWP network.
- All IIAs should have been renewed electronically using the EWP network.

HEIs could lose their ECHE accreditation for 2021-2027 if they do not comply with the dates and processes described above.

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs				Page:	10 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final

Electronic identity is a foundational part of these processes because a clear and trustworthy identification of the person that is involved in the mobility processes is an absolute requirement. Having a combination of government issued eID and academic identities for said person increases the level of trust on the received information.

2.3 Structure of the document

This document is structured in three major chapters:

Chapter 3 presents the services offered to the users as individuals that need to link identities and collect identity information into their own personal storages.

Chapter 4 presents how HEIs can participate in the network as data providers, providing academic identities and academic information. It includes a section to develop custom identity modules for SEAL.

Chapter 5 presents how HEIs can consume the identities the individuals and the links, either through federated or through self-sovereign means.

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs				Page:	11 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final

3. User level information

This section covers the user (understood as institutional SP) aspects of SEAL integration. That is, understanding which are the functionalities SEAL does provide for end users and how they can be leveraged and integrated by institutions on their own services through the standard interfaces SEAL offers.

HEIs can also act as providers of identity and linking data for SEAL (Identity Providers), either through standard interfaces or through non-standard interfaces. How to create an adapter for SEAL to know how to deal with this specific interaction is covered in section 4, and the standard ways for HEIs to be providers are described in section 5.

It also covers the standard ways SEAL offers to interact with the academic institutions, as consumers of the identity and linking data (SP). Technical details for this interaction are covered in section 6.

3.1 SEAL Service Dashboard

3.1.1 Web Dashboard

This section describes the functionalities provided by the Web dashboard through SEAL Services that allows the user to have her/his identities under control. This gives end users the opportunity of presenting these identities to HEIs in order to use them as validated credentials, hence taking advantages of educational services within different institutions. In addition, these personal digital identities can be linked among them, which permits the creation of a brand-new identity that while referring to the very same individual also combines attributes from previous different identities. As these new identities are processed by the core linking module of SEAL service, they are validated and certified with an appropriate level of assurance.

The web dashboard main page allows the user to have access to the dashboard functionalities through either a federate access method (Personal Data Store, PDS) or a Self-Sovereign Identity (SSI) method, which uses a user's wallet along with Distributed Ledger Technology (DLT) for issuing credentials. The PDS method is based on the storage of a datastore file with the corresponding encryption in a place where the user can control its use. This place could be either a local user's device (personal computer or mobile), a cloud service infrastructure or even both of them, in case the end user wants to have more redundancy over the data. To perform the access through this method the user shall select one of the options available and continue with either the uploading of the local file or its retrieval from the cloud service. In case this is the very first time this user is accessing the service, it will be also the opportunity to create a new PDS which will not have any personal information already. On the other hand, the SSI access method uses the uPort external platform as the selected technology for communicating over the DLT. Further, the uPort app also provides a wallet for certain types of issued credentials such as the ones used in the user authentication in SEAL.

The web dashboard functionalities are the list of operations the end user can perform once a success access has been done, these are integrated in the Identity Manager dashboard page.

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs				Page:	12 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final

The key feature of the Web Dashboard functionalities is the Identity Reconciliation one, which combines two identities previously loaded in session into a brand-new linked one. This new identity sustains the individual benefits of the identities from which it comes but also extending the potential relation of their linked attributes. This reconciliation process can be performed either in an automatic or manual way, achieving, as a result, a mentioned new identity with a certain Level of Assurance (LoA) that will depend on the trust relation between its linked attributes and its providers.

The functionality from where the user can gather his/her identities is settled in the Retrieve Identity Data button, and it allows the fetching of attributes from eIDAS, eduGAIN and eMRTD providers after the previous authentication in the corresponding selected source platform. Besides, the Web Dashboard also has a function for credential issuing, it is called Issue Verifiable Credential and it gives the user the possibility of emitting Verifiable Credential (VC) through the DLT, using in the process the uPort mobile application as the user's controllable wallet to where these VCs will be sent. This functionality supports the issuing of credentials that came from the same source providers as the original identities, hence eIDAS, eduGAIN and eMRTD, as well as the eIDAS-eduGAIN linked identities generated by SEAL.

The Derive Identity functionality performs the creation of a new identity that refers to a previously owned one but with a brand-new identifier. This makes this new identity as polyvalent as the original one regarding its uses; however, their attributes will not be trackable due to its nature of being a new identity. It is important to notice that the previous identity must be available at the time of the derivation, as per derivate a new identifier it is necessary to have the identity from which to derivate first.

Along with all these features, the web dashboard provides a Persistence store option (Save PDS file) for clustering the identities loaded in session in a permanent way on a Personal Data Store (PDS) file, which permits the user to resume these identities, as well as the SEAL session state, using the firstly mentioned PDS access methods. Thus, this is a future-proof environment where identities can live together as long as the end user is performing actions, and saving could be done as soon as this tasks are finished, leaving the service without information but keeping a copy of your resources available locally or in a cloud service of preference.

Save PDS file

Local Storage

Cloud Storage

Manage Identity Data

eIDAS identity

eduGAIN identity

Linked Identity

Derived Identity

Configure Data Store (?)

Retrieve Identity Data

eIDAS source

eduGAIN source

eMRTD source

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs			Page:	13 of 44	
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final

Identity Reconciliation

Automatic process

Manual process

Identity Reconciliation Status (?)

Issue Verifiable Credentials

eIDAS source

eduGAIN

eMRTD

eIDAS - eduGAIN

Derive Identifier

3.1.2 Mobile Application

This section describes both the functionalities and their integration on the SEAL application, abbreviated SEAL app, which indeed is the mobile dashboard face of SEAL Services. This application implements the same functionalities as the Web dashboard, giving a straightforward user experience and adding extra value such as the eMRTD reading functionality. This is a new unique feature only present in the mobile version of the SEAL, which allows the user to add the identity stored in both an ePassport and a national eID to a SEAL session by using the certified SEAL eMRTD Reader. This last-mentioned app is an auxiliary one required to perform the reading of the NFC chip of the document, as well as sending this personal information to the SEAL app itself, so its main purpose is to act as a bridge between the mobile dashboard, and an authorized NFC document with valid credentials.

When the user opens the SEAL app an automatic registering process is made by the application on the SEAL servers, initiating a new session if possible. In case this process is interrupted or incomplete a proper message would appear, informing the user of the error. After the successful completion of it, the user is redirected to the application main window, which groups all the functionalities in different iconic buttons.

These buttons are either direct functionalities or deployable submenus for accessing them but in a neater presentation. The below picture represents this commented main window and the function buttons of the SEAL application:

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs				Page:	14 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final

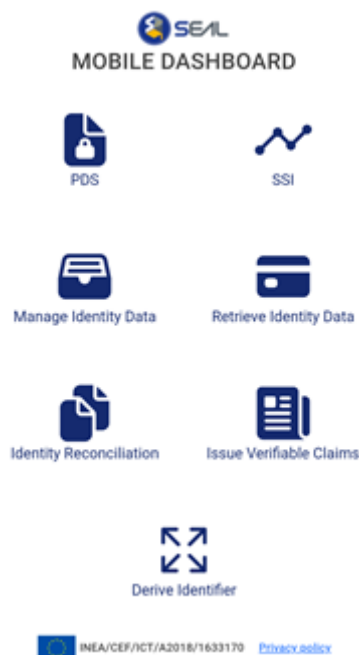


Figure 1: Mobile Dashboard main page

As it is shown in Figure 1., both the user experience and the user interface of the SEAL dashboard app has been enhanced but maintaining the very same functionalities than the web one, this is going to clearly result in a more instinctive use and control of the personal data.

The first button that we found when opening the application is the PDS one, which is indeed referring to the Personal Datastore management. With it the user can both retrieve a copy of the current state of the SEAL session in a PDS file (saved either locally or in a cloud service) or upload a previously available file in order to resume this particular state again into the active session. Both of them act as they do in the web dashboard, so users do not need to reorganize their behaviours when changing the platform; furthermore, all PDS files are usable in a cross-platform environment, which means that users can use their PC files in their mobile, or the ones generated using the SEAL app in their PC either.



Figure 2: PDS options

The Load PDS option presents a deployable submenu where users can select between the load of a local PDS or a cloud one. There is also the least feature called Scan QR code, which is for now under development, that will bring the option to upload a PDS file in a cross-platform environment. This has

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs				Page:	15 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final

been mentioned previously as a new powerful advance in the user experience changes of this new application, and it consists in the reading of a QR code that is presented in the Web Dashboard. This reading leads to the upload of the PDS from the SEAL app, allowing rapid actions in different platforms but without moving the critical files between devices, and transmitting just the very necessary ciphered information using SEAL secure channels. Besides all this, is important to notice that these same details apply for the Store PDS option, as its logic is the same as the load one but changing the receptor of the PDS file (downloading to the SEAL mobile app, instead of uploading from it).

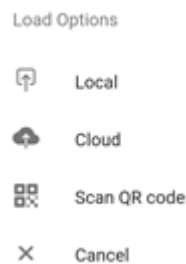


Figure 3: Load PDS options

The local PDS load option, the first of the three just above discussed, redirects the user flow to the persistence microservice page as it is described in the previous Web Dashboard section, either for loading an existing PDS or creating a new one:



Figure 4: Load local PDS

The cloud PDS load option allows the user to select one of the two cloud infrastructures, either Google Drive or Microsoft One Drive as in the Web Dashboard and facilitate the authentication in the selected platform of choice, loading the PDS in session as a result of a successful process. In addition to this, the

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs				Page:	16 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final

password for decrypting the selected PDS is needed to operate with it, as the file is encrypted in the cloud for extended security.

Load Options

-  Google Drive
-  One Drive
-  Cancel

Figure 5: Load Cloud options


datastore_20201223_cloud.seal
datastore_cloud_210129_data.seal
datastore_Google2OneD.seal
datastore_test_3.seal
datastore_UMA.seal
ds210420-c.seal

Password

Submit

Figure 6: PDS selection process

The Store PDS option gives the user, also as in the Web Dashboard, the chance of storing the PDS in either the local device or the cloud platform, with a mentioned needed password for encrypting and securing the identities saved in it:



Save Your Current Data

A File With Your Data will be added to your Local File Storage. ?

File Name

Password

Submit

Figure 7: Store PDS in the local mobile device

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs				Page:	17 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final

The SSI option in the main mobile dashboard page is related to this very same feature in the Web Dashboard, which purpose is connecting the current SEAL session to a valid DLT address of the user. With this bind through the uPort external application, it is possible to pair the user wallet through the DLT ecosystem, permitting Verifiable Credentials to be later issued using the VC Issuer of SEAL service. The Issue Verifiable Claims functionality, which is the one in charge of this task, allows the user to emit Verifiable Credentials (VCs) through the DLT using the uPort application wallet. As VC is concerning, is important to notice that they always have to come from previously loaded identities, so though the service that generates these new DLT credentials is SEAL, the in it contained attributes are the ones provided by approved sources. In this case, the VCs can be emitted out of eIDAS, eduGAIN or eMRTD identities, as they are the main identity providers for the SEAL service. Besides, it is also possible to emit eIDAS-eduGAIN credentials, which are VCs of linked identities of these two mentioned identity providers. As it is described, the actual behaviour of SEAL mobile application is identical to the Web Dashboard one and is usability and action recycling are in fact the key point of the user experience. Even so, the emission of VC has not been completely implemented yet, it will be in the last release of the application, added up to the already available functionalities.

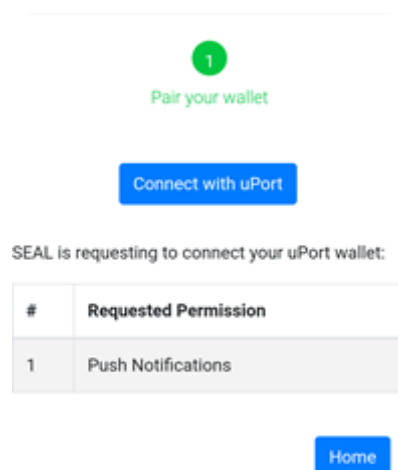


Figure 8: SSI authentication functionality on mobile SEAL application

The Manage Identity Data functionality provides a very important feature to the application, and as a mimic of the Web Dashboard. It presents the already loaded user's identities in the current session but notice that no identity will be displayed the first time that user opens the application. This functionality has a symbiotic relationship with the Retrieve Identity Data, which features the retrieval of the user identities from the available identity sources, in an identical flow as in the SEAL Web Dashboard., To improve the usability aspect, the identity sources have been arranged in a precise submenu, so the user does not need to navigate through different pages until redirected to the official Identity Provider one. These identity sources have also been commented in upper sections, as they are the very same package for all the functionalities of the SEAL service. As shown below they are eIDAS, eduGAIN and eMRTD:

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs				Page:	18 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final

Identity Sources





-  eIDAS
-  eduGAIN
-  eMRTD
-  Cancel

Figure 9: Authentication sources

The eMRTD identity source option connects the mobile SEAL dashboard application with the SEAL eMRTD Reader, which is the official SEAL external application (only Android available at the moment as a result of unreleased open APIs for iOS core NFC reading). It needs to be installed previously in the user device for its use. This external reader permits the user to fetch his/her identity from both ePassport and eID national card document, and its main page is the following. Also notice that preloaded example values are populated in the fields as a user guide on how to fill them.

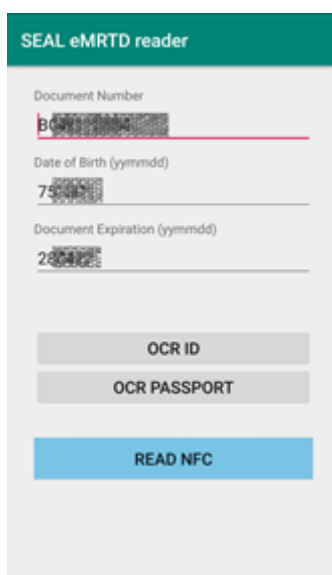


Figure 10: SEAL eMRTD reader mobile application

Continuing with the same user experience policy carried along with all the design patterns, the user can skip the manual population of the fields by using the camera device. For this, both an eID and ePassport OCR capability have been added, which leads to the document MRZ scanning, which allows the user to retrieve the values of the Document Number, Date of Birth and Document Expiration field. Nonetheless, if thorough action is preferred, instead of using the automatic OCR feature the user can insert these values manually.

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs				Page:	19 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final

Once the values are assigned either manually or automatically to the fields, the user can use the Near Field Communication (NFC) device integrated chip for reading inside the eID/ePassport. If confirmation of a successful process is displayed the extracted data is displayed and sent to the SEAL mobile application via a deeplink. This will trigger a dialogue window on the user device asking whether to open this URL with the official application (default option) or with another application instead. As this data is captured by SEAL app the process to send it to the SEAL service is executed, ciphering it up to SEAL servers. As a result of the processing of this data a new identity is creating in this very user session and presented to the user in the Manage Identity Data functionality, among all the other identities already loaded in it. Furthermore, if the process is either incomplete or erroneous all the personal information will be erased from within the SEAL core, and as neither the mobile application nor the reader saves any data on the user device, all the critical data is eradicated from the system. This result in an open and conscientiously treatment of user information in the entire procedure.

The Derive Identifier feature acts the same way as it has been explained openly in the Web Dashboard section, and it performs the process of generating a new identity that refers to a previously owned one but with a new identifier, enhancing anonymity of users and preventing the traceability when using these identities. The same occurs to the Identity Reconciliation core functionality, which has also been thoroughly discussed in upper sections, and whose main purpose is to link two identities from different data sources of the same individual creating a new bounded one as a result of it, however, it is not available at the moment on the SEAL app, but it will be in the complete final release.

3.2 Credentials issuing

This section describes the different standard interfaces SEAL offers to issue identity and linking information for the consuming services (SP). It can be divided in two main approaches:

- **Federated access**, requiring online connectivity between the service provider and SEAL at the moment of consumption, by issuing as a response a security assertion on any of the supported federation protocols.
- **self-sovereign access**, not requiring online connectivity between the service provider and SEAL at the moment of consumption, by issuing Verifiable Credentials to the user's wallet from SEAL, and then leaving the validation and consumption of the VC to the SP when the user accesses it.

SEAL does offer also the possibility of bridging both approaches: A SP willing to consume VCs but not willing to implement it, can use SEAL as a verifier, and consume the VC content through any of the federated interfaces. And vice-versa, a self-sovereign SP not willing to implement federated access, can point the user to SEAL, so the user accesses the federated data sources and gets a VC issued with their content, ready for consumption.

3.2.1 Federated Credentials

Two main interfaces are offered: a SAML and an OIDC interface. SAML is the most used and consolidated Internet standard for identity federation. It is a XML-based protocol for message exchange,

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs				Page:	20 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final

and the implementation covers both standard SAML and eIDAS extensions to SAML. OIDC is a light federation protocol based on JSON that is growing in usage thanks to the popularity of mobile devices. This federated access is used with two different purposes: authenticating the user and/or getting trusted data related to the user (for further authorisation checks after an authentication, or simply for data import/archiving). To this end, SEAL offers a different access endpoint for each purpose to the services. Both authentication and query purposes are invoked the same way, and, in fact, all authentication sources act as data sources as well (so authentication sources also implement the query API), as all of them transfer attributes, but they differ in a basic assumption: authentication sources guarantee that the identity of the user is validated through some mean (password, OTP, etc.) and the identity is returned along the data. Data sources do not need to guarantee it. This allows, for example, to gather data from self-sovereign sources, which can be trusted data as well (not self-asserted data).

From the user perspective, whenever a service is accessed, and it is connected to SEAL through a federated interface, at some point, the service will redirect the user to SEAL. There, the user will deal with the request manager user interface, where the user will need to choose the source of the data and follow the required steps to gather the data to be delivered to the SP.

Data sources can be live data sources (like eIDAS and eduGAIN, which will be explained below), or the user can choose to get the data from a personal storage, like a SSI wallet, or the SEAL Personal Data Storage (PDS).

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs				Page:	21 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final



Service Provider is requesting to query:

- ☒ PersonIdentifier
- ☒ FirstName
- ☒ GivenName
- ☒ FamilyName
- ☒ DateOfBirth
- ☒ SealLink
- ☒ schacHomeOrganization
- ☒ schacHomeOrganizationType

Select Data query source

eduGAIN

eIDAS

Personal Data Store

SSI Wallet

Mobile

Browser

Google Drive

One Drive

Accept

Reject

Click to consent to --> this request

Privacy Policy

Info

Figure 11: Request Manager Source Selector

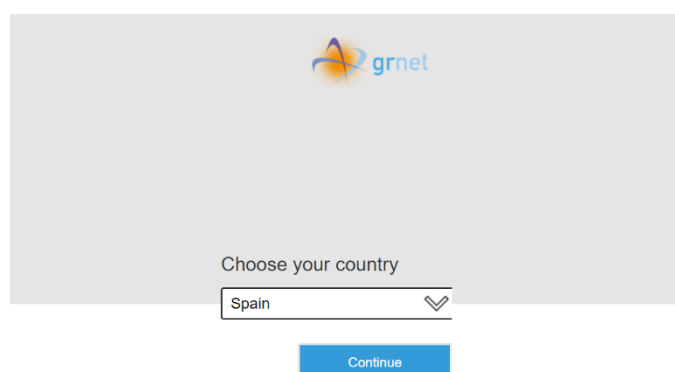
After selecting the source and gathering the data, the user will be prompted to consent the data transfer and will be allowed to select which data attributes are returned, getting always fine control over his data. The SPs have many configurable options, among which is pre-selecting the source of the data. In this case, the user interface to select the source will be bypassed, and for live sources, SEAL can be a fully transparent proxy if the SP wants to use it that way.

3.2.1.1 eIDAS

eIDAS is the main data and authentication source for SEAL. It gives access to national public administration issued e-IDs, notified by the member states under the eIDAS regulation.

It connects to the CEF eID infrastructure through a national proxy. Different modules, to adapt to the country-specific protocols of the eIDAS connectors can be plugged in. Currently, it is deployed with the adaptor for the Greek eIDAS Connector.

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs				Page:	22 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final



The image shows a web interface for selecting a country. At the top, there is a logo for 'grnet'. Below it, the text 'Choose your country' is displayed. Underneath, there is a dropdown menu with 'Spain' selected. At the bottom, there is a blue button labeled 'Continue'.

Figure 12: eIDAS Country Selector

The user will be redirected to the country selector of the Greek eIDAS connector, and through it to any eIDAS node. The user will be authenticated with her own country notified credential.



The image shows a web interface for the Spanish eIDAS IDP. At the top, there are logos for the Spanish Government (GOBIERNO DE ESPAÑA), Cl@ve (IDENTIDAD ELECTRONICA PARA LAS ADMINISTRACIONES), and eID connecting europe. Below these logos, there is a language selector dropdown set to 'Español'. The main heading is 'Identificación europea con eID extranjero'. Below this, there is a message: 'Por favor, confirme que desea enviar la siguiente información al proveedor de servicios europeo pulsando el botón Enviar.' followed by four input fields: 'Apellidos:', 'Identificador Nacional:', 'Nombre:', and 'Fecha de Nacimiento:'. At the bottom, there are two buttons: 'Enviar' (orange) and 'Cancelar' (grey).

Figure 13: Spanish eIDAS IDP

The control will go back to the request manager through several secure redirections, along with the security assertion and the user will be asked to consent the transfer of the attributes to the SP.

3.2.1.2 eduGAIN

The second data and authentication source for SEAL, and the main source of academic data. Gives access to thousands of identity providers from academic and research institutions globally, but mainly in Europe.

SEAL connects as a standard SP to the eduGAIN federation but does so through a proxy deployed at the Greek NREN. It could as well, be deployed, but this setting allows for a better control of the trust management.

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs				Page:	23 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final

The user will be redirected to the metadata discovery service of the Greek federation, and through it to any eduGAIN IDP in any of the sub-federations.



Figure 14: eduGAIN Metadata Discovery Service

The user will authenticate with its own's institution credentials and a security assertion will be returned to SEAL. The user will be asked to consent the transfer of the attributes to the SP.

3.2.2 Verifiable Credentials

Verifiable Credentials [1] (VCs) can be defined as digital identification cards securely stored in a user's "wallet" app in his or her controlled device. They can be thought of as the digital equivalent of physical identity cards which are stored in a physical wallet. Users can present these VCs to Relying Parties (i.e., Service Providers) that validate them and use the contained information to provide access to their

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs					Page:	24 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status:	Final

services (the same way we present physical identity cards in everyday life to access physical services). Just like physical identity cards, VCs originate and explicitly reference a VC Issuer, i.e. the entity Issuing and guaranteeing the accuracy of the contained information. The key difference is that Verifiable Credentials are tamper evident, and are bound to their holder (i.e. entity to whom they were issued to) via cryptographic means, making them impossible to counterfeit.

Specifically, a Verifiable Credential consists of a (set of) claim(s) regarding an entity made by an Issuer. Usually, a credential will include metadata about the Issuer, the expiry date, and means of verifying its integrity. Additionally, it might include metadata about revocation mechanisms used and so on. A Verifiable Credential (VC) is defined as a set of tamper-evident set of claims and metadata about those claims such that ownership and authorship can be cryptographically proved. In more details, a typical Verifiable Credential apart from the actual claims includes metadata about:

- The time of issuance;
- The identity of the Issuer;
- The subject of the credential;
- The type of credential;
- The verification means (proof).

These concepts are usually referred to as the Credential Graph and the Proof Graph (the proof graph contains the details of the verification means) (Figure 15).

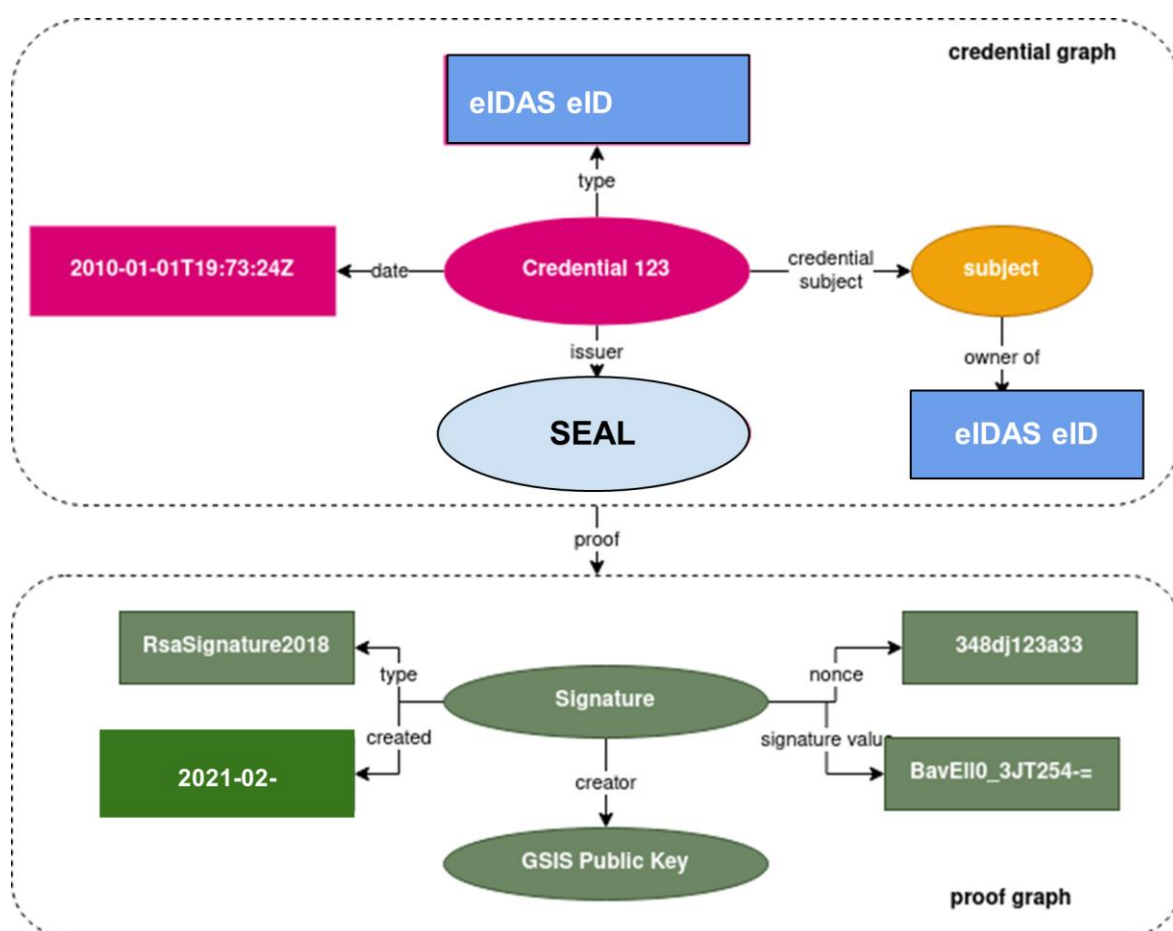


Figure 15: Credential Graph and the Proof Graph

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs					Page:	25 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status:	Final

The SEAL platform fully supports the issuance of Verifiable Credentials using the connected identities sources (eIDAS,eduGAIN, eMRTD¹), as well as the issuance of Verifiable Credentials form the linking of the aforementioned sources.

Specifically, the SEAL VC service, using specialized interfaces, allows the users to:

1. Pair their wallet app (currently uPort is supported [2]) with the platform (Figure 16);
2. Import their identification information from the connected authoritative sources (Figure 17);
3. Use these as assertions to issue Verifiable Credentials to the user's wallet attesting to the imported attributes and their links (Figure 18 and Figure 19).

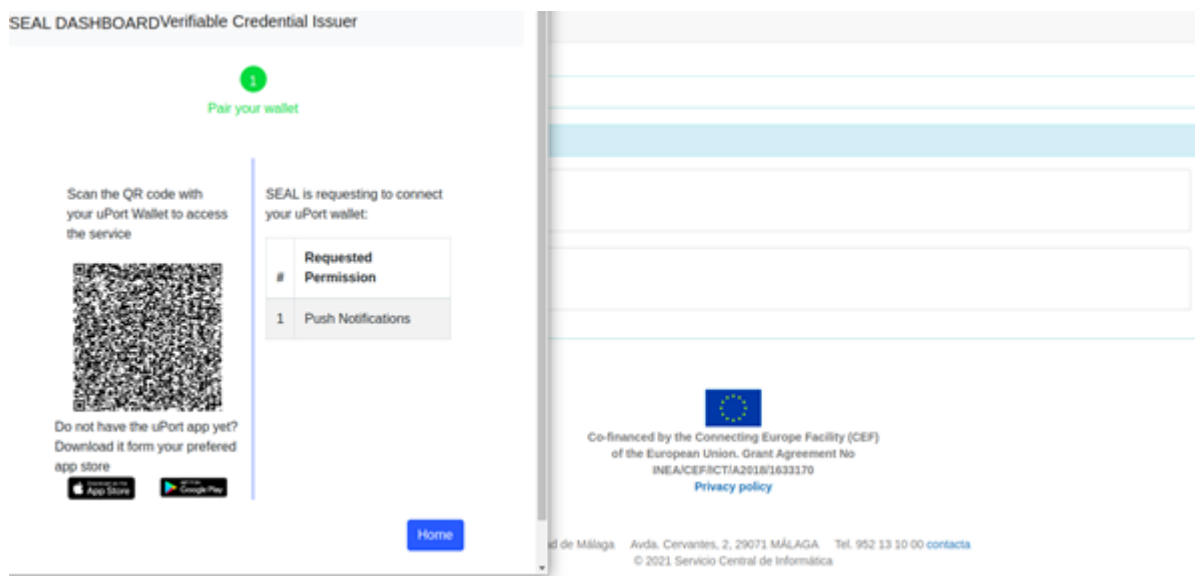


Figure 16: Wallet Connection Request

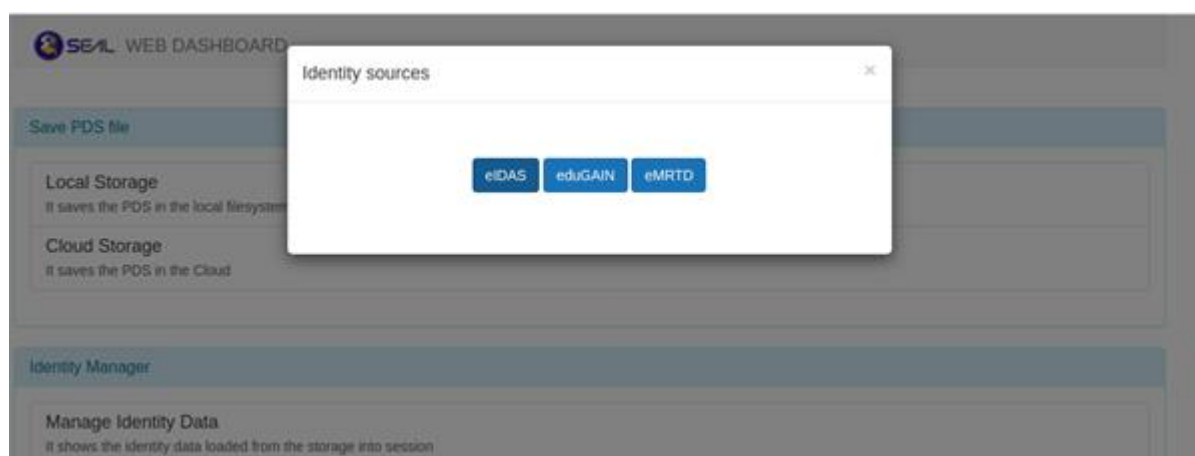


Figure 17: Authoritative Source Selection

¹ Eventually via the SEAL mobile dashboard app.

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs					Page:	26 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status:	Final

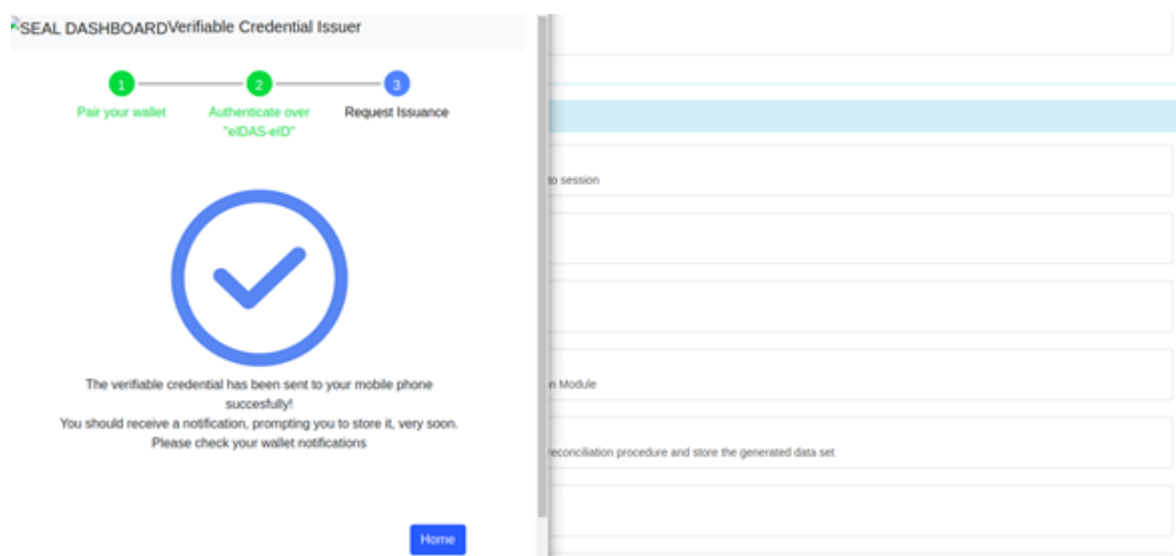


Figure 18: Verifiable Credential Issuance Notification

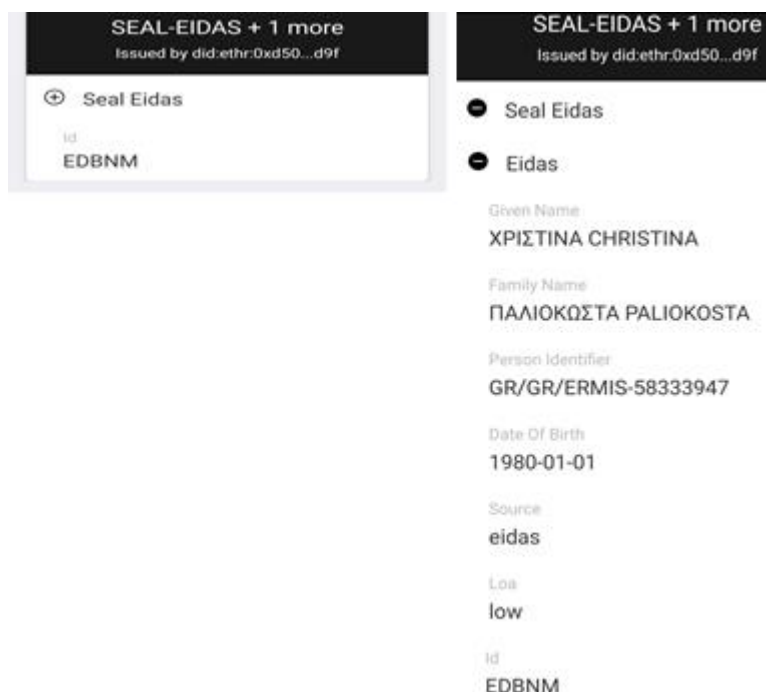


Figure 19: Verifiable Credential stored on user's mobile wallet

As a result, a SEAL enabled academic online service connected to SEAL Platform (VC enabled service) allows users to: a) issue a Verifiable Credential once. b) use that VC to authenticate and access the service “many times” (no need to connect back to authoritative resources each time a user requests service access)

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs					Page:	27 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status:	Final

4. HEI as Identity Provider

As described in section 3.2.1, HEIs can act as Identity Providers through the eduGAIN inter-federation, if their corporate IdP is connected to their national federation and it's metadata is exported to eduGAIN. Depending on the federation policy, the institution must actively accept or reject to be included, so some might be there for their own will and some will be there by default.

Despite most HEIs will have access through eduGAIN, or if they do not yet, will probably prefer to do so, other needs might appear. Data sources could be connected directly to SEAL using the common federation protocols - those SEAL has an adapter for - or even custom modules could be developed for other protocols or very specific systems access. This section covers all of these possibilities.

4.1 “SAML” (Direct IdP Connection to SEAL)

The SAML module of SEAL implements both the functionality of IdP, to serve data consumers to SEAL, and the SP functionality, to consume data from an IdP.

Despite the same instance used for SP-connector purposes can be used, it is recommended to deploy a new instance, specific for this IdP access purpose.

Internally, the module includes two adapted deployments of Simple SAML PHP software. For this purpose, a single one can be used.

On the module documentation (on the code repository, link below), configuration and deployment settings are described, for the SEAL specific part of the connection. The configuration for the SP part is as described on the Simple SAML PHP documentation:

<https://simplesamlphp.org/docs/stable/simplesamlphp-sp>

<https://simplesamlphp.org/docs/stable/saml:sp>

The code for the module can be found here:

https://github.com/EC-SEAL/SP_SAML

And the already built Docker image can be found here:

<https://hub.docker.com/r/faragom/sealsaml>

4.2 OIDC

The OIDC microservice of SEAL enables the consumption of identity assertions offered from Identity Providers implementing the OpenID Connect (OIDC) specification. Specifically, this module can be configured to connect to any OIDC compatible IdP and request the authentication of the user, retrieve the identity assertions and import them in the SEAL session (essentially allowing its use in the rest of the SEAL flows).

It is implemented using Spring boot and Spring security and its source code is available at: <https://github.com/EC-SEAL/eidas-idp-ms>. Additionally, this module is built as a docker image, available on docker hub (<https://hub.docker.com/repository/docker/endimion13/seal-eidas-idp>) for easier deployment. Configuration instructions are available at the github repo and together with a deployment example: <https://github.com/EC-SEAL/eidas-idp-ms/blob/development/docker-compose.yml>.

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs			Page:	28 of 44	
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final

Currently there are two versions of this microservice instantiated on the SEAL platform:

1. The first version of this microservice is configured to request and retrieve the personal identification information of the users from the eIDAS network via an eIDAS proxy service maintained in Greece (exposing OIDC interfaces).

The second version of this microservice is configured to be a client over the Athens ESMO GW². As a result, this version allows the consumption of SEAL issued Verifiable Credentials over OIDC from the SEAL portal. Essentially, allowing the SEAL service to act as a VC verifier as well as an issuer and eventually proxy this functionality to connected HEI services (please visit section 5 for more details).

4.3 “SSI credential issuing”

The SEAL platform contains an SSI Credential microservice. This microservice is capable of integrating with the rest of the SEAL microservices in order to:

- retrieve from the session previously imported user attributes,
- request the retrieval of user attributes,
- request the linking of the user’s identity attributes, and finally
- issue to the users mobile wallet³ Verifiable Credentials attesting to the aforementioned attribute sets.

Specifically, the SEAL Issuer microservice implements a full Verifiable Credential Issuer (as that was described in section 3.2.2) complete with the required user interfaces and back channel functionality. This module is implemented using express and next.js (backend and front end javascript frameworks respectively) and internally re-uses the sdks available by uPort [3] [4].

The source code of this microservice is available on github: <https://github.com/EC-SEAL/vc-issuer> together with configuration and deployment instructions. Additionally, this microservice is build as a docker image and it is available over docker hub: <https://hub.docker.com/repository/docker/endimion13/seal-issuer>

4.4 “Developing a generic data source”

As indicated above, some cases will require to build a new module implementation to support other protocols or even specific access to information systems. We will introduce the microservice communication and security architecture of SEAL, with examples and references to existing implementations, to facilitate building the internal part of the module.

SEAL follows a microservice architecture. That is, independent services implementing small parts of the business and support logic of the application are deployed independently and interconnected through a network protocol to collaborate on fulfilling the overall purpose of the application. It is an evolution of the modular application design, but detaching the modules on the full stack, producing isolated and resilient components that can operate on their own and can be easily managed to provide a dynamic

² A identification portal maintained by the Greek Ministry of Edugain and UAegean offering eIDAS proxying services and Verifiable Credential verification capabilities to registered Service Providers.

³ In the current version the uPort mobile wallet is supported.

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs				Page:	29 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final

service capability and availability in response to varying needs (hot-plugging of new modules and of replicated instances).

Below can be seen a model of the functional modules SEAL supports and their interactions:

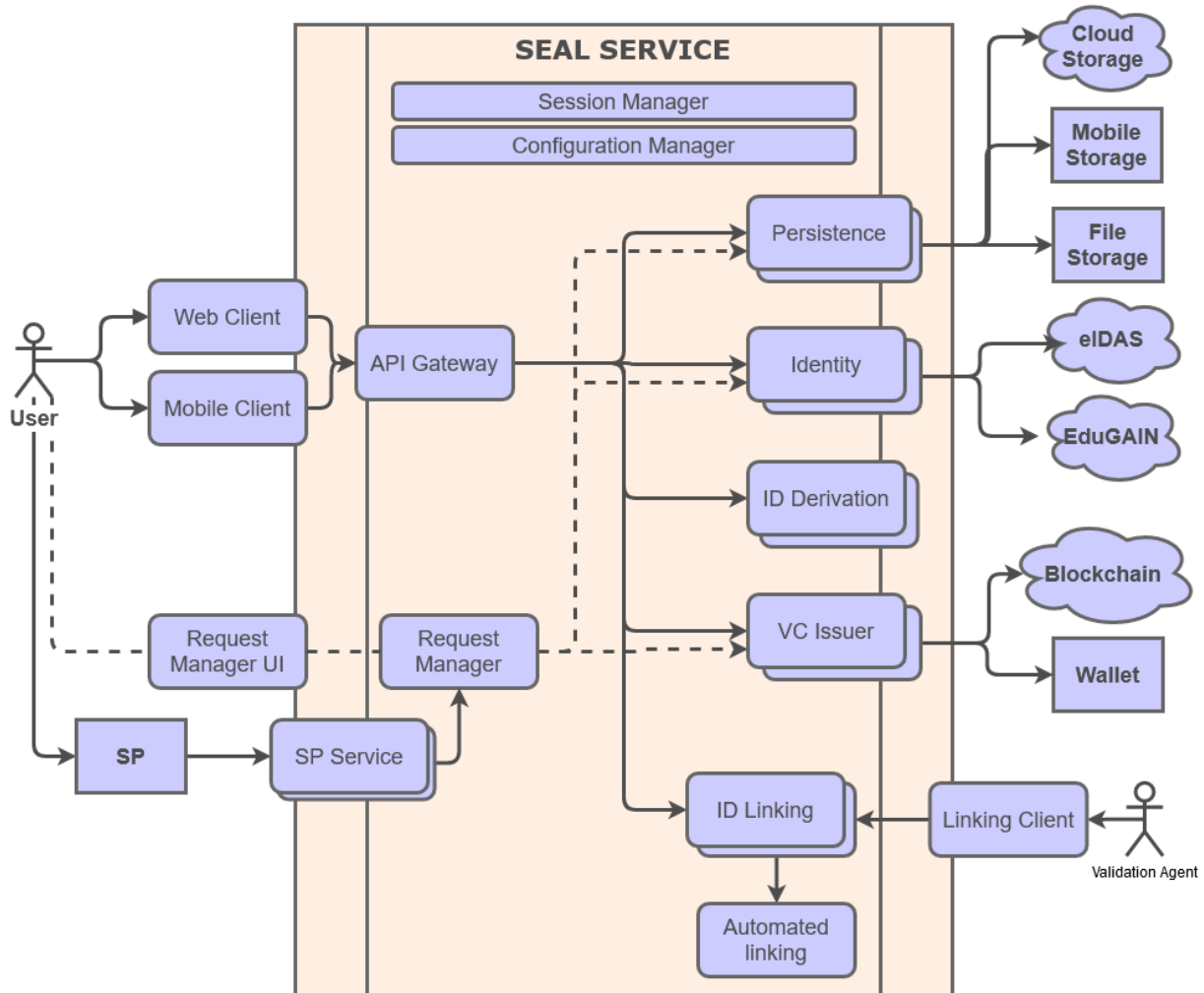


Figure 20: Microservice functional diagram

This section focuses on explaining how to create an implementation of a “Identity” microservice, but instructions can be used to build any of the other extendable microservices on the right column, just by changing the API to implement.

As can be seen, SEAL includes two microservice that are transversal to the rest of them:

- The **Config Manager** is a centralised repository for the high-level shared configuration of the microservices (microservices can still use local configuration for the specific things). It implements a common back-channel (see next paragraph) API to access said configuration data. The most important one is the list of registered microservices and their metadata.
- The **Session Manager** is a centralised Session Management object. It implements 3 core functionalities through back-channel APIs: the session management API, the security token issuance and validation API, and the Data Store API. All three will be described later

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs					Page:	30 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status:	Final

SEAL implements a custom model to secure the API calls. Microservices can communicate through two means:

- back-channel API calls: direct HTTP connections between microservices, without involving the user browser (so, no UI is shown)
- front-channel API calls: HTTPS redirections between microservices through the user browser (allowing the invoked microservice to display specific UI)

Back-channel calls are secured through HTTP Signatures⁴. SEAL provides implementations in Java (inherited from EWP), PHP and Python⁵. Caller microservice will insert a signature on the HTTPS API request, which the invoked microservice will validate (to ascertain the identity of the caller and the integrity of the request) and will check the signing key against a list of authorised keys (to check if the caller is authorised). Invoked microservice is authenticated and secured through HTTPS, so no need to do HTTP Signature on the response. The calls will exchange at least a session identifier, which is the element that coordinates the action of all the microservices to fulfil the business logic of the user.

Front channel calls are secured by the exchange of a security token along the redirection. The invoked microservice will check the signature of the token (to ascertain the identity of the caller and the integrity of the request) and will check the signing key against a list of authorised keys (to check if the caller is authorised to call). The token includes the session identifier, so the invoked microservice can retake the application flow.

As described above Session Manager is the most important microservice, as it holds together the whole application and implements the support for the security model described. Beside the session API, which is a classic start/end/add/remove/update/get interface⁶, and the Datastore API, which will be described later, Session Manager implements the security token API⁷. This API allows a microservice to request a token to be issued for a front-channel call between a microservice of origin, and a microservice of destination. And then, it has another call to allow validating the signature and audience of a received token. This way, the microservices do not need to implement the token generation or validation, lightening the development effort and moving this security management to a centralised component.

The API calls have parameters. For back-channel calls, they are passed in the query string or the POST body, depending on the API, but for redirections, to limit the size of the exchanged token, the parameters are written on the common session as fixed variables. The process of an API call is as follows:

Back-channel:

1. The caller checks the microservice registry, to get the metadata of the microservice to invoke.
2. The caller guesses the API URL by searching the signature of the API call on the metadata.
3. The caller prepares the parameters of the call (the POST body and/or the GET query string)
4. The caller calculates and includes the HTTP Signature on the HTTP call headers.
5. The caller does the HTTPS call.

⁴ Still a draft, but highly consolidated and growing in usage. Currently it is used by the Erasmus Without Papers infrastructure. <https://tools.ietf.org/id/draft-cavage-http-signatures-01.html>

⁵ See the repositories from <https://github.com/EC-SEAL>

⁶ Interface specification in Open API: https://github.com/EC-SEAL/interface-specs/blob/a2d0618c9a803375ca156b4155ec3df34492c84b/SEAL_Interfaces.yaml#L41

⁷ Interface specification in Open API: https://github.com/EC-SEAL/interface-specs/blob/a2d0618c9a803375ca156b4155ec3df34492c84b/SEAL_Interfaces.yaml#L205

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs			Page:	31 of 44	
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final

6. The receiver gets the HTTPS call.
7. Following the algorithm, validates the integrity of the HTTP Signature.
8. Check the identity of the caller towards the authorised callers list.
9. If something failed, return error.
10. If all went right, perform the call logic and return the proper response status.

Front-channel:

1. The caller checks the microservice registry, to get the metadata of the microservice to invoke.
2. The caller guesses the API URL by searching the signature of the API call on the metadata.
3. The caller prepares the parameters of the call and writes them to the session manager microservice through the Session Manager on the variables expected by the API call.
4. The caller does the same with the callback URL to the clientCallbackAddr variable (the url the microservice will invoke when it's done, to return the result)
5. The caller gets a security token passing the sessionID and the ID of the destination microservice.
6. The caller does a POST redirect to the API URL, passing the security token as a POST parameter named msToken.
7. The receiver gets the token and passes it to the API for token validation on the Session Manager.
8. If the response is correct, the payload, including at least the Session ID will be returned.
9. The receiver gets the expected parameters from the fixed variables on the Session Manager.
10. If all request parameters are OK, perform the call logic.
11. The receiver writes the response object on the proper Session variable.
12. The receiver requests a security token.
13. Redirect back to the callback URL passing the token along and the status.

The Data Store is the internal SEAL representation of the Personal Data Store. Every time a user opens a session and imports any data, it is stored in the session Data Store. The user can then choose to export it to a PDS or simply let it disappear with the session. The load of a PDS, for security reasons, will erase any data existing on the Data Store in Session prior to the load.

The PDS allows managing a set of container objects, that can carry a variety of objects inside (currently data sets and links) and supports a reset/add/delete/replace/get/search interface⁸

Now that the common components are described, we will focus on describing the API for the Identity Module and how it needs to use the common components to integrate with SEAL.

Two APIs exist, which are compatible but semantically different: The Authentication Source API and the Identity Source API.

The Authentication Source API, which only those identity sources that guarantee that the user will be properly authenticated before getting the data, describes a single call, as/authenticate⁹, a front-channel call which expects a standard request object and a standard data source metadata object (because a module can handle the access to different data sources, and they can be discovered by the caller). The response object (exactly the same object as the standard request) is expected on the dsResponse session variable.

⁸ Here, the proposal and the argumentation for this interface can be found <https://github.com/EC-SEAL/interface-specs/issues/3>

⁹ API definition https://github.com/EC-SEAL/interface-specs/blob/a2d0618c9a803375ca156b4155ec3df34492c84b/SEAL_Interfaces.yaml#L1757

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs				Page:	32 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final

The Identity Source API is implemented by all modules, also those implementing as/authenticate (for those, the /is/query call is exactly the same as as/authenticate), but some modules will not implement as/authenticate and only is/query. Functionally, it is the same API call, but with a different name and context.

Code examples in different programming languages can be found on the SEAL repository, in Java¹⁰, PHP¹¹ and Python¹².

¹⁰ <https://github.com/EC-SEAL/edugain-idp> and <https://github.com/EC-SEAL/eidas-idp-ms>

¹¹ https://github.com/EC-SEAL/SP_SAML

¹² <https://github.com/EC-SEAL/reconciliation>

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs				Page:	33 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final

5. HEI as Identity consumer

This section will guide potential Service Providers into the best strategies to consume data from SEAL, focusing in suggesting the best support software option based on the requirements of the applications the organisation wants to connect.

5.1 “SAML”

SAML is the most reliable option for web services. Many well-tested and used solutions exist and has been widely used for years, integrated at application level, and at application server level. But is a heavy option, and eventually, the usual drawbacks of XML language appear. In the last years, some security issues with XML and SAML have appeared, so a good maintenance of the integration, with frequent updates and using well-maintained and tested software is highly advisable (see list below).

SEAL supports an interface for Requests based on the eIDAS extensions. These allow for a better and more flexible control of the requested attributes, but as a general advice, if the requirements can be covered with a Standard SAML2 solution, it is advised to use it, as eIDAS-supported solutions are still scarce and not well established. SEAL includes in its code such a solution for PHP language, in case it is required by the integrator.

5.1.1 Available software for integration

These are the main options for integration in different frameworks and settings. All of them are solvent solutions and currently being maintained.

Solution	Setting	Comments
Simple SAML	PHP, application	Full IdP/SP suite. Needs to be fully deployed to integrate with external applications [5]
Mod_auth_mellon	Any, app-server.	Apache 2.0 application server module. Will serve any application served by this server, on any supported language ¹³
Pysaml2	Pyhon, library	Popular development. Base library of the SATOSA proxy ¹⁴
Python-saml	Python, library	Another implementation of the SAML2 protocol, by the OneLogin corporation, open source [6]
Shibboleth	Java, app-server	The original Idp/SP suite. It also has a library, but the developers themselves recommend NOT to integrate the lib, but the app, as many controls and good security practices are coded on the app and the lib is not designed for the general public [7]

¹³ https://github.com/Uninett/mod_auth_mellon

¹⁴ <https://github.com/IdentityPython/pysaml2>

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs			Page:	34 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1
				Status:	Final

Keycloak	Java, application or app-server level integration.	Full IdM suite that offers modular adapters for application integration. Needs to be deployed to integrate. Documentation also lists 3 rd party solutions for other languages [8]
Java-saml	Java, library	Another implementation of the SAML2 protocol, by the OneLogin corporation, open source ¹⁵
Spring-security-saml	Java, framework	Spring Framework module to support application authentication with SAML2. Only for Spring or Spring-boot applications [9]

5.1.2 General Guidelines

SP will issue SAM2 AuthnRequests and will receive SAML2 Responses through the HTTP-POST Binding.

Requests and responses must be signed with an X509 certificate. A certificate is used for convenience, but the trust on the certificate does not depend on its issuer, but on the exchange process.

A set of attributes will be requested, from the eIDAS, SCHAC, EduPerson or SEAL schemas, and those available at the source will be returned, listed with the short name. Specific filters can be applied per SP to transform the attributes as needed. The list of available attributes can be found in Table 1: List of supported attributes and claims.

As described, SEAL presents two parallel endpoints for SAML SPs. The query and the auth endpoints. Each one of them is a separate IdP that can be queried normally (and also access control can be established separately if needed). They can be found in these basic paths:

```
https://HOST:PORT/auth/
https://HOST:PORT/query/
```

The metadata for each IdP can be found in the following sub-path:

```
/saml2/idp/metadata.php
```

The SP can pre-select the source for the data-. In SAML it can be done as follows. The SAML Standard accepts to define the list of acceptable IdPs in the AuthnRequest. SEAL accepts just one value, from the available list (eIDAS, eduGAIN, PDS, SSI basically, but the proper list must be provided by the SEAL administrator, as new modules might have been added).

```
<saml2p:IDPList>
  <saml2p:IDPEntry ProviderID="eIDAS" />
</saml2p:IDPList>
```

The Level of Assurance of the request can be established as a SP parameter on the SEAL-side metadata.

¹⁵ <https://github.com/onelogin/java-saml>

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs			Page:	35 of 44	
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final

5.1.3 Connection setting

It is necessary to deliver the metadata of the SP to the SEAL administrator through a trusted channel (the deployer of the service will establish which one).

After that, the SEAL administrator will deliver the SEAL service metadata for the SP to trust and accept responses from. The SP operator must choose to be accepted in the auth or query interface (or, by default, to both).

If any of the fixed parameters described above is required, it must be provided along the metadata.

From this point on, SP can issue AuthnRequests to SEAL and accept Responses from SEAL.

5.2 “OIDC”

OIDC is a light and growing option for federated access, especially on mobile environments. The authentication process requires, besides the redirection for user interaction, a direct interaction between the app and the OIDC provider

5.2.1 Available Software for integration

These are the main options for integration in different frameworks and settings. All of them are solvent solutions and currently being maintained.

Solution	Setting	Comments
Keycloak	Java, application or app-server level integration.	Full IdM suite that offers modular adapters for application integration. Needs to be deployed to integrate. Documentation also lists 3 rd party solutions for other languages and for many Java Frameworks [8]
Mod_auth_openidc	Any, app-server.	Apache 2.0 application server module. Will serve any application served by this server, on any supported language ¹⁶
Oidcphp	PHP, application	OIDC RP implementation. Can be integrated into app ¹⁷
phpOIDC	PHP, application	Certified implementation [10]
Simple OIDC client	PHP, application	Minimal implementation. Can be easily integrated with any app ¹⁸
Pyoidc	Python, library	SATOSA RP integrates this one ¹⁹
Oic	Python, library	Another implementation in python [11]

¹⁶ https://github.com/zmartzone/mod_auth_openidc

¹⁷ <https://github.com/oidcphp/core>

¹⁸ <https://github.com/rciam/simple-oidc-client-php>

¹⁹ <https://github.com/rohe/pyoidc/>

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs				Page:	36 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final

App-auth-iOS	iOS, library	Certified implementation for iOS ²⁰
App-auth-android	Android, library	Certified implementation for Android ²¹

5.2.2 General Guidelines

Clients are individually authorised to query the OIDC Provider. Each one will be provided with a pre-shared key that will be used to authenticate the requests as the OIDC protocol allows to.

Requests and responses are not signed, as all the critical steps are done through back-channel connections. Both SEAL OP and the client RP must be deployed in a HTTPS environment so the remote side URL can be authenticated.

A set of attributes will be requested, from the eIDAS, SCHAC, EduPerson or SEAL schemas. They can be specified as scopes of the OIDC request. A list of the available claims and their matching to SAML attributes can be seen below:

Table 1: List of supported attributes and claims

Full Name	Friendly Name	Claim
http://eidas.europa.eu/attributes/naturalperson/CurrentFamilyName	FamilyName	efln
http://eidas.europa.eu/attributes/naturalperson/CurrentGivenName	FirstName	efin
http://eidas.europa.eu/attributes/naturalperson/DateOfBirth	DateOfBirth	edob
http://eidas.europa.eu/attributes/naturalperson/PersonIdentifier	PersonIdentifier	epi
http://eidas.europa.eu/attributes/naturalperson/BirthName	BirthName	ebn
http://eidas.europa.eu/attributes/naturalperson/PlaceOfBirth	PlaceOfBirth	epob
http://eidas.europa.eu/attributes/naturalperson/CurrentAddress	CurrentAddress	eca
http://eidas.europa.eu/attributes/naturalperson/Gender	Gender	eg
2.5.4.3	cn	eocn
eduOrgHomePageURI	eduOrgHomePageURI	eohipu
eduOrgLegalName	eduOrgLegalName	eoln
eduOrgPostalAddress	eduOrgPostalAddress	eopa
2.5.4.7	l	eol
1.3.6.1.4.1.25178.1.2.17	schacExpiryDate	sed
1.3.6.1.4.1.25178.1.2.9	schacHomeOrganization	sho
1.3.6.1.4.1.5923.1.1.1.1	eduPersonAffiliation	epaf
1.3.6.1.4.1.5923.1.1.1.5	eduPersonPrimaryAffiliation	eppaf
1.3.6.1.4.1.5923.1.1.1.6	eduPersonPrincipalName	eppn
1.3.6.1.4.1.5923.1.1.1.12	eduPersonPrincipalNamePrior	eppnp
1.3.6.1.4.1.5923.1.1.1.8	eduPersonOrgUnitDN	epoudn
1.3.6.1.4.1.5923.1.1.1.13	eduPersonUniqueid	epui
2.16.840.1.113730.3.1.241	displayName	epdn

²⁰ <https://github.com/openid/AppAuth-iOS>

²¹ <https://github.com/openid/AppAuth-Android>

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs				Page:	37 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final

2.5.4.42	givenName	epgn
0.9.2342.19200300.100.1.3	mail	epma
0.9.2342.19200300.100.1.41	mobile	epmo
2.5.4.10	o	epo
2.5.4.4	sn	epsn
1.3.6.1.4.1.25178.1.2.14	schacPersonalUniqueCode	spuc
1.3.6.1.4.1.25178.1.2.15	schacPersonalUniqueID	spuid
1.3.6.1.4.1.5923.1.1.1.10	eduPersonTargetedID	epti
SealLink	SealLink	seallink

As described, SEAL presents two data access modes, the query and the auth. For the OIDC endpoint, the access mode is described as a scope. Possible values are:

```
sealprojectep:auth
sealprojectep:query
```

The metadata for the OP can be found in a standard well-known URL:

```
https://HOST:PORT/auth/realms/seal/.well-known/openid-configuration
```

The SP can pre-select the source for the data. As above, it can be defined with a specific scope, as listed below. SEAL has the basic list (eIDAS, eduGAIN, PDS, SSI) but the proper list must be provided by the SEAL administrator, as new modules might have been added).

```
sealprojectsrc:eIDAS
sealprojectsrc:EduGAIN
sealprojectsrc:PDS
sealprojectsrc:SSI
```

5.2.3 Connection setting

It is necessary to deliver the basic data of the RP to the SEAL administrator through a trusted channel (the deployer of the service will establish which one):

- **The Client ID:** A fixed and unique string that identifies the RP towards the SEAL OP.
- **The base URL of the RP:** SEAL will use it to deduce the Well-Known URL to retrieve all the OIDC RP parameters.

After that, the SEAL administrator will deliver the shared secret, unique for the client to trust and accept responses from.

From this point on, RP can issue OIDC Requests to SEAL and will then receive an access token from SEAL to fetch the claims through a secure back-channel connection.

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs			Page:	38 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1
				Status:	Final

5.3 “SSI credential consumption”

The SEAL service enables its users to issue Verifiable Credentials (VCs) containing their (linked) personal identification information. These VCs are securely stored in the user’s mobile device (using a secure open source app called uPort) in such a way that at any time the user can prove, to any requester:

- the VC was issued by the SEAL service,
- the VC was issued to the device of the presenter ,
- the VC has not expired,
- the VC has not been revoked.

In other words, the identity of the issuer (SEAL service) and of the user (VC holder) are bound together in such a way that they are always verifiable in any VC disclosure (presentation).

Once securely stored, in the user's mobile app, the user can present these credentials to any consumer (Verifier) in order to prove ownership over the contained claims (asserted by the SEAL Issuer service).

In order for the user to present the VCs, the Verifier must generate an appropriate Credential Disclosure Request [12] and push it to the user’s mobile wallet app. As a result, the Verifier must consume the necessary open source sdk to generate and propagate this request.

As a result, consumption of SEAL identities can take place with no third-party involvement required and without integration with the SEAL service. The only requirement is for the SP to implement the necessary infrastructure to:

1. request from the user’s wallet the SEAL issued Verifiable Credentials.
2. verify their authenticity,
3. verify the user’s/presenter’s ownership.
4. verify that they have not been revoked (by accessing the SEAL maintained revocation list on a publicly available Besu blockchain, maintained by the university of the Aegean which is available at <http://I4mlab-besu.westeurope.cloudapp.azure.com:8545>).

However, to minimize the barriers and costs to entry, using this new technology stack, for the Service Providers (SPs) an SSI Verification service was implemented exposing a simple OIDC. Using this interface Service Providers can request the authentication of the users via SEAL issued Verifiable Credentials and retrieve the contained claims as OIDC responses. The aforementioned service implements all the required validations and simply returns to the SP the claims contained in the VC over OIDC.

In other words, this service:

- On the one hand it acts as a full OpenID Connect (OIDC) server capable of implementing all of the OIDC flows defined in the specification of the protocol and
- On the other hand it acts as a Self Sovereign Identity (SSI) Agent/Verifier capable of generating requests for the presentation of the Verifiable Credentials issued by the SEAL service.

Specifically, the Verifier has been built as a Keycloak plugin. Keycloak [13] is an open source software product that allows single sign-on with Identity and Access Management. Specifically, Keycloak supports both OIDC and SAML and is capable of acting as a fully functional industrial ready Identity Management Server (IdMs). Keycloak supports the creation and instalment of various plugins that intervene in the normal authentication flow of a user. Through the use of these plugins (referred to as Service Provider Interfaces or SPIs) various means of user authentication mechanisms can be deployed.

The code for the Keycloak plugin that implements the Verifiable Credential Disclosure Requests and translates the response to OIDC assertions is available over GitHub at: <https://github.com/endumion/ssi-keycloak> and the service is deployed at the Athens ESMO GW (<https://esmo-gateway.eu/auth/>) and maintained by the University of the Aegean.

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs			Page:	39 of 44	
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final

In order for a SP to consume SEAL issued Verifiable Credentials over OIDC, the SP first needs to register as a client on the aforementioned Athens ESMO GW. Next, they need to implement the necessary OIDC flows (either manually or via a library depending on the technological stack the SP uses) and add the type of the VC that is to be requested from the user on the OIDC scope parameter. Specifically, SEAL issues the following Verifiable Credentials:

Verifiable Credential Name	Description
SEAL-EIDAS	Verifiable Credential containing the Personal Identification information of the holder retrieved over the eIDAS network
SEAL-EDUGAIN	Verifiable Credential containing the Academic attributes of the holder retrieved over the eduGAIN network
SEAL-EIDAS-EDUGAIN	Verifiable Credential containing the linked Personal Identification information of the holder retrieve over the eIDAS network and the Academic attributes of the holder retrieved over the eduGAIN network

For example, an SP might initiate a credential disclosure request by making a call as follows:

```
https://esmo-gateway.eu/auth/realms/SSI/protocol/openid-connect/auth?client_id=test&state=0d9c5cb6-efdb-4089-87de-8fc7970d542c&redirect_uri=http%3A%2F%2Fdss1.aegean.gr%2FisErasmus%2FSSI%2Fverify%3FspId%3D051083yyyyza3%26auth_callback%3D1&scope=openid%20SEAL-EIDAS&response_type=code
```

Depending on the requested VC (via the scope parameter), after the VC presentation and verification the SP will receive the VC contained assertions as OIDC claims following the mapping of the table below:

Verifiable Credential Requested	OIDC attributes
SEAL-EIDAS	<ul style="list-style-type: none"> • eidas-familyName • eidas-firstName • eidas-dateOfBirth • eidas-loa • eidas-credential-id
SEAL-EDUGAIN	<ul style="list-style-type: none"> • edugain-cn • edugain-eduOrgHomePageURI • edugain-eduOrgLegalName

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs					Page:	40 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status:	Final

	<ul style="list-style-type: none"> • edugain-eduOrgPostalAddress • edugain-schacExpiryDate • edugain-schacHomeOrganization • edugain-eduPersonAffiliation • edugain-eduPersonPrimaryAffiliation • edugain-eduPersonPrincipalName • edugain-eduPersonPrincipalNamePrior • edugain-eduPersonOrgUnitDN • edugain-eduPersonUniqueId • edugain-displayName • edugain-givenName • edugain-mail • edugain-mobile • edugain-o • edugain-sn • edugain-schacPersonalUniqueCode • edugain-schacPersonalUniqueId • edugain-eduPersonTargetedID • edugain-loa
SEAL-EIDAS-EDUGAIN	<ul style="list-style-type: none"> • eidas-familyName • eidas-firstName • eidas-dateOfBirth • eidas-loa • eidas-credential-id • edugain-cn • edugain-eduOrgHomePageURI • edugain-eduOrgLegalName • edugain-eduOrgPostalAddress • edugain-schacExpiryDate • edugain-schacHomeOrganization • edugain-eduPersonAffiliation • edugain-eduPersonPrimaryAffiliation • edugain-eduPersonPrincipalName • edugain-eduPersonPrincipalNamePrior • edugain-eduPersonOrgUnitDN • edugain-eduPersonUniqueId • edugain-displayName • edugain-givenName • edugain-mail • edugain-mobile • edugain-o • edugain-sn • edugain-schacPersonalUniqueCode • edugain-schacPersonalUniqueId

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs					Page:	41 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status:	Final

	<ul style="list-style-type: none">• edugain-eduPersonTargetedID• edugain-loa• link-loa
--	--

Closing, as mentioned in section 4.2 the SEAL OIDC module is registered as a Service Provider on the Athens ESMO GW. This feature enables the SEAL service to proxy VC verification capabilities to any registered HEI of the SEAL service either over SAML or OIDC interfaces.

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs				Page:	42 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final

6. Conclusions

Services developed as an outcome of the SEAL project can provide a solid foundation for creating electronic identities that combine government issued eID attributes with academic ones in a single, so to say, integrated package and doing so with three useful characteristics both for the services consuming such identities and the individuals making use of them: trustworthiness, privacy preservation and personal control.

The capability of retrieving information from electronic identity documents, such as ePassport, opens the service to non-EU citizens, with the added benefit of being able to add academic information obtained from a service such as eduGAIN with global reach., This opens opportunities for participants in Erasmus+ from EHEA countries and also for participants in other mobility programmes all over the world.

The identity derivation module for generating Verified Credentials future proofs the services for what seems to be the near future of electronic identity. Also, the ability to create derived identities with only the parts needed for using a given service, increases the level of control individuals can have over their personal data and privacy.

The fact that the information can be stored in user devices or personal cloud services, allows for those small HEIs that lack the resources needed for running an Identity Management System to have a distributed one of sorts. This would require the development of a module for off-line verification of student identity that was outside the scope of the current project, but all needed services for such module are already in place.

In summary, the services developed by the SEAL project provide HEIs participating in Erasmus+ with a solid foundation for their users to share identity information in a secure, privacy preserving way.

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs				Page:	43 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status: Final

7. References

- [1] W3C, *Verifiable Credentials Data Model 1.0*, <https://www.w3.org/TR/vc-data-model/> retrieved 2021-05-28
- [2] UPORT, *Build User-Centric Ethereum Apps*, <https://developer.uport.me> retrieved 2021-05-28
- [3] UPORT, *uPort Transports*, <https://www.npmjs.com/package/uport-transports> retrieved 2021-05-28
- [4] UPORT, *uPort Credentials Library*, <https://www.npmjs.com/package/uport-credentials> retrieved 2021-05-28
- [5] SIMPLE SALM PHP, *SP API Reference*, <https://simplesamlphp.org/docs/stable/simplesamlphp-sp-api> retrieved 2021-05-25
- [6] ONE LOGIN DEVELOPERS, *Phyton Toolkit*, <https://developers.onelogin.com/saml/python> retrieved 2021-05-25
- [7] WIKI SHIBBOLETH, *ApplicationIntegration*, <https://wiki.shibboleth.net/confluence/display/SP3/ApplicationIntegration> retrieved 2021-05-25
- [8] KEYCLOAK, *Securing Apps*, https://www.keycloak.org/docs/latest/securing_apps/ retrieved 2021-05-25
- [9] SPRING, *Spring Security SALM*, <https://spring.io/projects/spring-security-saml> retrieved 2021-05-25
- [10] BITBICKET, *PhP OIDC*, <https://bitbucket.org/PEOFIAMP/phpoidc/src/master/> retrieved 2021-05-25
- [11] PYPI, *A Python OpenID Connect implementation*, <https://pypi.org/project/oic/> retrieved 2021-05-25
- [12] UPORT, *Selective Disclosure Request*, <https://developer.uport.me/messages/sharereq> retrieved 2021-05-28
- [13] KEYCLOACK, *Open Source Identity and Access Management*, <https://www.keycloak.org/> 2021-05-28

Document name:	D6.1 Technical documentation on SEAL integration package for HEIs					Page:	44 of 44
Reference:	D6.1	Dissemination:	PU	Version:	0.1	Status:	Final