Digital Security: Cybersecurity, Privacy and Trust

DS-02-2014 Access Control

European Commission

Horizon 2020
European Union funding
for Research & Innovation

From Real-world Identities to Privacy-preserving and Attribute-based CREDentials for Device-centric Access Control

ReCRED

WP4 – Identity Consolidation and Real-to-Online Identity Mapping

Deliverable D4.2 "Full Identity Consolidator and Attributes Acquisition"

Editor(s):	CUT
Author(s):	Kostantinos Papadamou, Savvas Zannettou, Michael Sirivianos, George Fakas, Charalambos Chrysostomou (CUT), Patricia Callejo (IMDEA), Ruben Cuevas (UC3M), Alberto Caponi, Claudio Pisa (CNIT), George Gugulea (CSGN), Bilgili Ozlem, Joao Gomez, Mark Fidell (VERIZON), Vangelis Bagiatis (UPCOM), Antonis Hatzikonstantinou, Yannis Katsaros (EXUS), Evangelos Kotsifakos, Michalis Pramateutakis, Michalis Koulinas (WEDIA), Claudio Soriente, Carmelo Acosta (TID)
Dissemination Level:	Public
Nature:	Demonstrator
Version:	2.1

ReCRED Project Profile

Contract Number	653417
Acronym	ReCRED
Title	From Real-world Identities to Privacy-preserving and Attribute-based CREDentials for Device-centric Access Control
Start Date	May 1st, 2015
Duration	36 Months

Partners

University of Piraeus	University of Piraeus research center	Greece
Telefónica Investigación y Desarrollo	Telefonica Investigacion Y Desarrollo Sa	Spain
verizon	Verizon Nederland B.V.	The Netherlands
certSIGN	Certsign SA	Romania
wedia	Wedia Limited	Greece
EXUS	EXUS Software Ltd	U.K.
upcom Bringing business and IT together	Upcom Bvba (sme)	Belgium
THE PRODUCTIZERS	De Productizers B.V.	The Netherlands
2004	Cyprus University of Technology	Cyprus
UNIVERSIDAD CARLOS III DE MADRID	Universidad Carlos III de Madrid	Spain
cnit	Consorzio Nazionale Interuniversitario per le Telecomunicazioni	Italy
BAKER & McKENZIE	Studio Professionale Associato a Baker & Mckenzie	Italy

Document History

AUTHORS

CUT	Kostantinos Papadamou, Savvas Zannettou, Michael Sirivianos, George Fakas, Charalambos Chrysostomou
UC3M	Ruben Cuevas
IMDEA	Patricia Callejo
CNIT	Alberto Caponi, Claudio Pisa
CSGN	Geroge Gugulea
VERIZON	Bilgili Ozlem, Joao Gomez, Mark Fidell
UPCOM	Vangelis Bagiatis
EXUS	Antonis Hatzikonstantinou, Yannis Katsaros
WEDIA	Evangelos Kotsifakos, Michalis Pramateutakis, Michalis Koulinas
TID	Claudio Soriente, Carmelo Acosta

VERSIONS

Version	Date	Author	Remarks
0.1	04-01-2017	CUT	Initial Table of Contents and distribution of first draft to partners
0.2	10-01-2017	CUT, UC3M, IMDEA, CNIT, VERIZON, EXUS, UPCOM, WEDIA, TID, UPRC	Revised Table of Contents
0.3	14-01-2017	CUT	Added the Executive summary and the Introduction
0.4	19-01-2017	CUT	Added the description for the Identity Consolidator platform
0.5	21-01-2017	CUT	Added Identity Consolidator architecture diagram
0.6	25-01-2017	CUT	Added an initial description for the Physical Identity Acquisition module
0.7	29-01-2017	IMDEA	Added Storage API description
0.8	04-02-2017	CUT	Updated the description for the Physical Identity Acquisition module
0.9	06-02-2017	IMDEA	Added the Identity Repository schema description
1.0	09-02-2017	UC3M/IMDEA	Added the description for the Online Identity Acquisition module
1.1	11-02-2017	CUT	Added 3rd Party API description
1.2	12-02-2017	IMDEA	Minor changes in the Identity Repository description
1.3	13-02-2017	VERIZON	Added Authentication Management module description
1.4	15-02-2017	UPCOM	Added the description for the Identity Profile Management mobile application
1.5	17-02-2017	UPCOM	Added the description for the Consent Management mobile application
1.6	19-02-2017	EXUS/UPCOM	Finalized the description for the

			Identity Management module
1.7	21-02-2017	CNIT	Added Credential Management module description
1.8	22-02-2017	IMDEA	Added Identity Integration module description
1.9	24-02-2017	WEDIA	Added the description for the Account Management module section
2.0	26-02-2017	CUT	Comments and corrections
2.1	28-02-2017	CUT	Proof reading – final version
2.2	01-08-2017	CUT	Updated Physical Identity Acquisition module functionalities description

Executive Summary

The Identity Consolidator Platform is one of the key components of the ReCRED architecture. It enables the seamless integration of the multiple identity attributes of a user, both physical and online and provides access to this information to third parties taking into consideration relevant security, privacy, authorization and authentication aspects.

This deliverable, “Full Identity Consolidator and Attributes Acquisition”, is based on D4.1, which was an initial design of the Identity Consolidator platform. It describes the entire design and the implementation of the Identity Consolidator Platform while providing detailed descriptions of the supported functionalities for each module that consist the Identity Consolidator. The description of the supported functionalities is classified according to the 10 modules that together form the Identity Consolidator platform. These are:

1. Authentication Management module
2. Credential Management module
3. Account Management module
4. Identity Management module
5. Online Identity Acquisition module
6. Physical Identity Acquisition module
7. Identity Integration module
8. Storage API
9. Identity Repository
10. 3rd Party API

Table of Contents

Executive Summary 5
List of Figures 11
1 Introduction 16
1.1 Purpose of the document 16
1.2 General overview 16
1.3 Structure of the document 16
2 Identity Consolidator Platform 17
2.1 Identity Consolidator Web Application 19
3 Authentication Management Module 22
3.1 OpenAM 22
3.2 User Authentication 22
3.2.1 Means of Authentication and LoAs 22
3.2.2 Tentative Access and Access promoting with Behavioral Authentication Authorities 24
3.2.3 Failover Authentication Mechanisms 25
3.3 User Enrollment (Registration) 26
3.3.1 Means of Registration 26
3.4 User Account Deletion 26
3.5 Identity Attribute value transfer among different value transfer 27
4 Credential Management Module 27
4.1 PABAC API Server 28
4.1.1 Idemix API Server 28
4.1.2 U-prove API Server 29
4.2 Credential Management Application 31
4.2.1 Credential Management Module Application Frontend 32
4.3 Credentials Backup and Restore 35
4.3.1 Main menu 36
4.3.2 Cryptographic Credentials stored on the device 36
4.3.3 Cryptographic Credentials backup in the Identity Consolidator server 38
5 Account Management Module 39
5.1 Behavioral Authentication Authority registration 40
5.2 Identity Provider registration 42
5.3 Service Provider registration 45
5.4 Account Locking mechanism 47

5.4.1 Account locking and status 47
5.4.2 Account Locking policies 52
5.5 Account Recovery mechanism 54
5.6 Mobile Connect 55
5.6.1 APIGEE OneAPI Exchange Discovery Service 55
5.6.2 Identity Consolidator as a Discovery Service 55
5.7 Degree of Privacy Selection and Enforcement 56
5.7.1 Least Degree of Privacy 56
5.7.2 Highest Degree of Privacy 56
6 Identity Management Module 56
6.1 Identity Profile Management 56
6.1.1 Identity Profile Management Web Front-end 57
6.1.2 Identity Profile Management Mobile Application 66
6.1.3 De-Anonymization Risk Assessment 75
6.2 Consent Management 79
6.2.1 Consent Management Back-end 79
6.2.2 Consent Management Web Front-end 80
6.2.3 Consent Management Mobile Application 80
7 Online Identity Acquisition Module 87
7.1 User assisted online identity binding process 88
7.1.1 Identity Information Acquisition process 88
7.1.2 Interaction with the Identity Repository 90
7.1.3 Access to the Identity Providers Information 90
7.1.4 Authorization and Privacy Management 90
7.2 Non-user assisted Online Identity binding process 91
7.2.1 Acquisition process 91
7.2.2 Interaction with the Identity Repository 92
7.2.3 Access to the Identity Providers Information 92
7.2.4 Authorization and Privacy management 93
7.3 Supported Functionalities 93
8 Physical Identity Acquisition Module 94
8.1 Physical Identity Acquisition 96
8.1.1 Initiate Physical Identity Acquisition process 97
8.1.2 Capture Physical Identity Document and Face photos 98

8.1.3 View/Manage identity information and uploaded pictures 102
8.1.4 Address acquisition 103
8.1.5 Real-time video presentation of Identity documents using WebRTC 105
8.1.6 RFID Identity document scanning using NFC-enabled devices 106
8.2 Physical Identity Verification 112
8.2.1 Peer-to-Peer Verification using crowdsourcing techniques 112
8.2.2 Automated verification methods 115
8.2.3 View verification results 116
8.2.4 Address Verification 117
8.2.5 Face-to-Face Identity Document verification by trained auditors 119
8.3 Physical Identity Acquisition mobile application 120
8.4 Physical Identity Acquisition Internal REST API 122
8.4.1 User’s general identity verification status 123
8.4.2 Physical Identity Document verification status 123
8.4.3 Identity Attribute verification status 124
8.4.4 Identity attribute peer-to-peer verification results 125
9 Storage API 126
9.1 REST Services 127
9.2 Header Fields 127
9.3 CREATE, UPDATE, and DELETE operations 128
9.3.1 Creation of information 128
9.3.2 Conditional Requests 128
9.3.3 Full Vs Partial Modification 129
9.3.4 Delete information 129
9.4 Query Invocations 129
9.4.1 Authentication and Authorization 129
9.4.2 Additional Path Information 130
9.4.3 Query Parameters 130
9.4.4 Filtering 131
9.5 Response Format 134
9.6 Error Codes 134
9.7 Structure 135
9.7.1 entry Element 135
9.7.2 Singular Fields 135

9.7.3 Plural Fields 136
9.7.4 Name Element 138
9.7.5 address Element 138
9.7.6 account Element 139
9.7.7 verificationInfo Element 139
9.7.8 mediaItem Element 140
9.7.9 urls Element 140
9.8 Example 140
9.9 Compatibility with Standards 142
10 Identity Repository 142
10.1 Notation and Conventions 143
10.2 Definitions 143
10.3 Database design 143
10.4 User Account Table 145
10.5 User Physical Identities Info Table 147
10.6 Identity Providers data Table 148
10.7 User Names Info Table 154
10.8 User Phones Info Table 156
10.9 User Emails Info Table 158
10.10 User Addresses Info Table 159
10.11 User Acquired Locations Table 163
10.12 Urls Info Table 164
10.13 MediaItem Table 165
10.14 Verification Table 167
10.15 Audits Info Table 168
10.16 Financial Information Table 170
10.17 Cryptographic Credentials Table 171
10.18 Behavioral Authentication Auths Table 173
10.19 Behavioral Authentication Authorities Reference Table 173
10.20 Service Providers Users Table 174
10.21 Identity Providers Users Table 175
10.22 Account Locking Policies Table 176
10.23 Providers Table 179
10.24 User Notifications Info Table 179

10.25 User Physical Documents Cropped Regions Table 180
10.26 User Relationships Info Table 181
11 Identity Integration Module 181
11.1 Consolidation of identity information from multiple Identity Providers 182
12 3rd Party REST API 183
12.1 High Level Operations 183
12.2 Transfer trust between Service Providers 183
12.3 User Account status in a particular Service Provider 184
12.4 Behavioral Authentication Authorities that collected behavioral profiles for a user 185
12.5 Inform the Identity Consolidator for signs of unusual behavior 186
12.6 Inform the Identity Consolidator for BAAs authentication outcomes 187
12.7 Inform the Identity Consolidator for the behavioral profiles stored by a BAA 188
12.8 Cryptographic Credential issuance to the User’s Device 189
12.9 Physical Identity Acquisition Embed API 191
13 Conclusion 191
14 References 192

List of Figures

Figure 1. Identity Consolidator Platform architecture 17
Figure 2. Identity Consolidator web application landing page 20
Figure 3. Identity Consolidator platform login page 21
Figure 4. Identity Consolidator platform create account page 21
Figure 5. Identity Consolidator platform main page 22
Figure 6. Authentication method 1: Username (email address) and Password 23
Figure 7. Authentication method 2: FIDO UAF 23
Figure 8. Authentication method 3: Mobile Connect 24
Figure 9. IRMA Architecture 28
Figure 10. IRMA Issuance Protocol 29
Figure 11. U-Prove client-server architecture 30
Figure 12. Credential Management module submodules and interactions 31
Figure 13. List of available credential ready to be issued 32
Figure 14. Credential Issuing: Link of the User Device to the API server by means of QR Code 33
Figure 15. ReCRED Wallet Application 34
Figure 16. ReCRED Wallet Application: issuance consent display 34
Figure 17. ReCRED Wallet application: Credential list and details 35
Figure 18. Credential Backup & Restore mobile application: Main page 36
Figure 19. List with the Cryptographic Credentials stored in the mobile device 37
Figure 20. Page that shows to the user the details of a cryptographic credential 37
Figure 21. Details of an encrypted cryptographic credential 38
Figure 22. List with the Cryptographic credentials that are backed up in the Identity Consolidator 38
Figure 23. Process of fetching a cryptographic credential from the Identity Consolidator server 39
Figure 24. Register a Behavioral Authentication Authority with the Identity Consolidator 40
Figure 25. BAA management page of the Account Management module web front-end 41
Figure 26. Enable / disable BAAs to be used 41
Figure 27. Administrator screen for Identity Provider registration 43
Figure 28. Management of registered Identity Providers - IDC Administrator 43
Figure 29. Identity Provider registration by the user 43
Figure 30. User screen for the management of his registered Identity Providers 44
Figure 31. Service Provider registration screen 46
Figure 32. Registered Service Providers management screen 46
Figure 33. Lock/unlock API call implementation 48
Figure 34. Lock/Unlock API call implementation within Storage API Manager 48
Figure 35. Account locking policies management screen 53
Figure 36. Account locking policies for Identity providers 53
Figure 37. Account locking policies for Service Providers 53
Figure 38. Identity Profile Management web application main page 58
Figure 39. Identity Profile Management web application. Identity Providers users' identity attributes 58
Figure 40. Edit the value of a user’s identity attributes acquired from his Identity Providers 59
Figure 41. Edit dialog fort identity attributes acquired from an Identity Provider 59

Figure 42. Identity Profile Management web application. Edit identity attribute values or transfer identity attributes among IdPs functionalities 60
Figure 43. Identity Profile Management web application. User's financial information page 61
Figure 44. User's names information 61
Figure 45. User's acquired names details and verification information 62
Figure 46. User's acquired locations information page 63
Figure 47. User's declared websites information 63
Figure 48. User's acquired media items information 64
Figure 49. User's Physical Identity Documents information 64
Figure 50. User's phones information 65
Figure 51. User's declared email addresses information 65
Figure 52. User's declared addresses information 66
Figure 53. Identity Profile Management mobile application landing page 67
Figure 54. Identity Providers that know a selected identity attribute 68
Figure 55. List of Identity attributes 68
Figure 56. Identity attributes known to an identity provider 68
Figure 57. List of Identity Providers 68
Figure 58. List of submitted Physical Identity documents 69
Figure 59. List of acquired identity attributes for a specific Physical Identity document 70
Figure 60. List of identity attributes 71
Figure 61. List of Service Providers that know an identity attribute 71
Figure 62. List of the Service Providers that a user has shared identity attributes with 72
Figure 63. List of identity attributes known to a Service Provider 72
Figure 64. List of created partial verifiable profiles 73
Figure 65. Partial verifiable profile share dialog 74
Figure 66. Create partial verifiable profile page 74
Figure 67. De-anonymization risk calculation page for Identity Providers 75
Figure 68. De-anonymization risk calculation example for Facebook 76
Figure 69. Financial information identity attributes de-anonymization risk 77
Figure 70. Financial information identity attributes de-anonymization risk calculation 77
Figure 71. Create new consent policy page 81
Figure 72. Selection of specific identity attribute 81
Figure 73. Selection of identity attributes below a certain LoA 82
Figure 74. Selection of specific Identity Provider 82
Figure 75. Selection of Identity Providers below a certain LoA 82
Figure 76. Selection of specific Service Provider 82
Figure 77. Selection of Service Providers below a certain LoA 83
Figure 78. List of a user's identity attributes 83
Figure 79. View created policies per identity attribute 84
Figure 80. List of Identity Providers 84
Figure 81. List of created policies per Identity Provider 85
Figure 82. List of Service Providers 86
Figure 83. List of created consent policies per Service Provider 86
Figure 84. Online Identity Acquisition module main page 88
Figure 85. Online Identity Acquisition Facebook Login 89

Figure 86. Online Identity Acquisition Google Login....89
Figure 87. Facebook Login permissions authorization dialogs....90
Figure 88. Google Login permissions authorization dialog....91
Figure 89. Online Identity Acquisition Facebook crawler....92
Figure 90. Online Identity Acquisition Google crawler....92
Figure 91. List connected Online Accounts page....94
Figure 92. Physical Identity Acquisition and Verification processes....95
Figure 93. Physical Identity Acquisition web application home page....97
Figure 94. User chooses the Physical Identity document to verify screen....98
Figure 95. Declare identity document information screen....98
Figure 96. Notification to capture and upload a photo of the identity document....99
Figure 97. User captures a photo of his whole identity document....99
Figure 98. List of required parts to crop from the captured identity document photo....100
Figure 99. The user is cropping the part of the identity document photo that includes his identity number....100
Figure 100. User is requested to capture face photos for verification purposes....101
Figure 101. User captures a photo holding his identity document....102
Figure 102. User captures a photo of his front face (selfie)....102
Figure 103. View/manage identity uploaded document photos screen....103
Figure 104. Methods that the user has to declare his addresses....104
Figure 105. Address acquisition by clicking on a map....104
Figure 106. Address acquisition by declaring its details....105
Figure 107. Screen where the user declares the email address of the real-time video presentation verifier....106
Figure 108. Real-time identity presentation using WebRTC....106
Figure 109. Identity number audit example....114
Figure 110. Name and surname audit example....115
Figure 111. Holding identity document audit example....115
Figure 112. Peer-to-peer verification results page....117
Figure 113. Physical Identity Acquisition. Address verification methods offered to the users....117
Figure 114. User captures a photo of his utility bill for address verification....118
Figure 115. User is cropping the address details from his uploaded utility bill photo....119
Figure 116. Peer-to-peer audit that compares the address details on the utility bill with the declared address details....119
Figure 117. Face-to-face identity verification console for f2f auditors....120
Figure 118. Physical Identity Acquisition mobile application mane page....121
Figure 119. Physical Identity Acquisition mobile application: user declares the details of the identity document that he wants to verify....122
Figure 120. Storage model....144

Table of Abbreviations

ABAC	Attribute-Based Access Control
AES	Advanced Encryption System
AMM	Account Management Module
API	Application Programming Interface
BAA	Behavioral Authentication Authority
BAC	Basic Access Control
CC	Cryptographic Credential
CM	Consent Management
CMM	Credential Management Module
F2F	Face-to-Face
HTTP	Hypertext Transfer Protocol
ICAO	International Civil Aviation Organization
ID	Identity
IDC	Identity Consolidator
Idemix	Identity Mixer
IdP	Identity Provider
IIM	Identity Integration Module
IMM	Identity Management Module
IRMA	I Reveal My Attributes
JAAS	Java Authentication and Authorization Service
LoA	Level of Assurance
NFC	Near Field Communication
OAuth	Open Authentication
OCR	Optical Character Recognition
OIDC	OpenID Connect
OTP	One Time Password

PABAC	Privacy-Preserving Attribute Based Access Control
PVP	Partial Verifiable Profiles
QR	Quick Response
REST	Representational State Transfer
RFID	Radio Frequency Identification
SSL	Secure Sockets Layer
SP	Service Provider
TEE	Trusted Execution Environment
TPM	Trusted Platform Module
UI	User Interface
UX	User Experience
WebRTC	Web Real-Time Communication

1 Introduction

1.1 Purpose of the document

The main purpose of this document is to provide a detailed description of the design and the implementation of the various modules that form the Identity Consolidator platform. First, a general description of the final architectural design of the Identity Consolidator that was implemented is provided. Next, following the defined architecture, a detailed description of the implementation of the functionalities supported by each module is provided as well as representative screenshots from the applications that have been developed to offer those functionalities to the end-users. Last, a description of the Identity Repository schema, which holds all the acquired identity attributes and identity information of the users, is included.

1.2 General overview

The Identity Consolidator is among the major components of the ReCRED architecture. This component plays a major role in most of the use cases of the ReCRED platform and takes place in most of the piloting activities. It facilitates the seamless integration of the multiple identity attributes of a user, both physical and online. Specifically, it is responsible for horizontally binding the online identities of a user and vertically binding the real-word identities of a user to independent verifiable identity attributes.

In general, the Identity Consolidator offers to the users an Identity Management service where they can voluntarily submit identity attributes and proofs of online account ownership in order to achieve the consolidation of their physical and online identities and many other functionalities related to their submitted identity attributes. This identity attributes acquisition and verification part of this service is consisted of both the Physical Identity Acquisition and the Online Identity Acquisition modules. The Physical Identity Acquisition module is responsible for the acquisition and verification of the identity attributes included in all the real-world identities of a user in a secure and privacy-preserving manner. The Online Identity Acquisition module is responsible for the collection of all the identity attributes included in the online accounts of a user. All the collected identity information are securely stored and protected in the Identity Repository. All the modules of the Identity Consolidator are interacting with the Identity Repository through the Storage API.

Moreover, the Identity Consolidator exchanges verified identity attributes with Identity Providers (IdPs) using federated login protocols (such as OpenID Connect [1], OAuth [2][3]). The Identity Consolidator acts as a Relying party when it collects identity attributes from IdPs. It can also act as an Identity Provider when it proves identity attributes to verifiers upon a user’s request.

Moreover, the Identity Consolidator is consisted of the Authentication Management, the Credential Management, the Account Management and the Identity Management modules. They are all described in detail in the rest of the document. All the modules interact with each other through well-defined REST APIs. All the other ReCRED entities like Service Provider’s and Identity Providers can communicate with the Identity Consolidator Platform through the 3rd Party API that it offers.

1.3 Structure of the document

The document is structured as follows. Section 2 defines the final architecture of the Identity Consolidator platform. Section 3 describes the various authentication schemes that Authentication Management module offers to the users. Depending on the level of assurance that the user is willing to have, the Identity Consolidator requires different authentication method. Section 4 describes the

implementation of the Credential Management module. Section 5 provides a description on the implementation of the Account Management module. Section 6 describes the Identity Management module, which is consisted of the Identity Profile Management and the Consent Management sub-modules. Section 7 and 8 provide the implementation details of the Online Identity Acquisition and the Physical Identity Acquisition modules accordingly. Section 9 provides the implementation details of the Storage REST API while Section 10 provides a description of the schema of the Identity Repository. Section 11 describes the Identity Integration module and Section 12 describes the implementation details of the 3rd Party REST API that the ReCRED entities use to communicate with the Identity Consolidator. We conclude in Section 13.

2 Identity Consolidator Platform

The Identity Consolidator Platform is an integral part of the ReCRED ecosystem. As mentioned above, the Identity Consolidator is responsible for horizontally binding the online identities of a user and vertically binding the real-world identities of the user to independent verifiable identity attributes. It allows the users to voluntarily submit their identity attributes and proofs of account ownership through an identity management service. The user interacts with this service using an identity management web and mobile application.

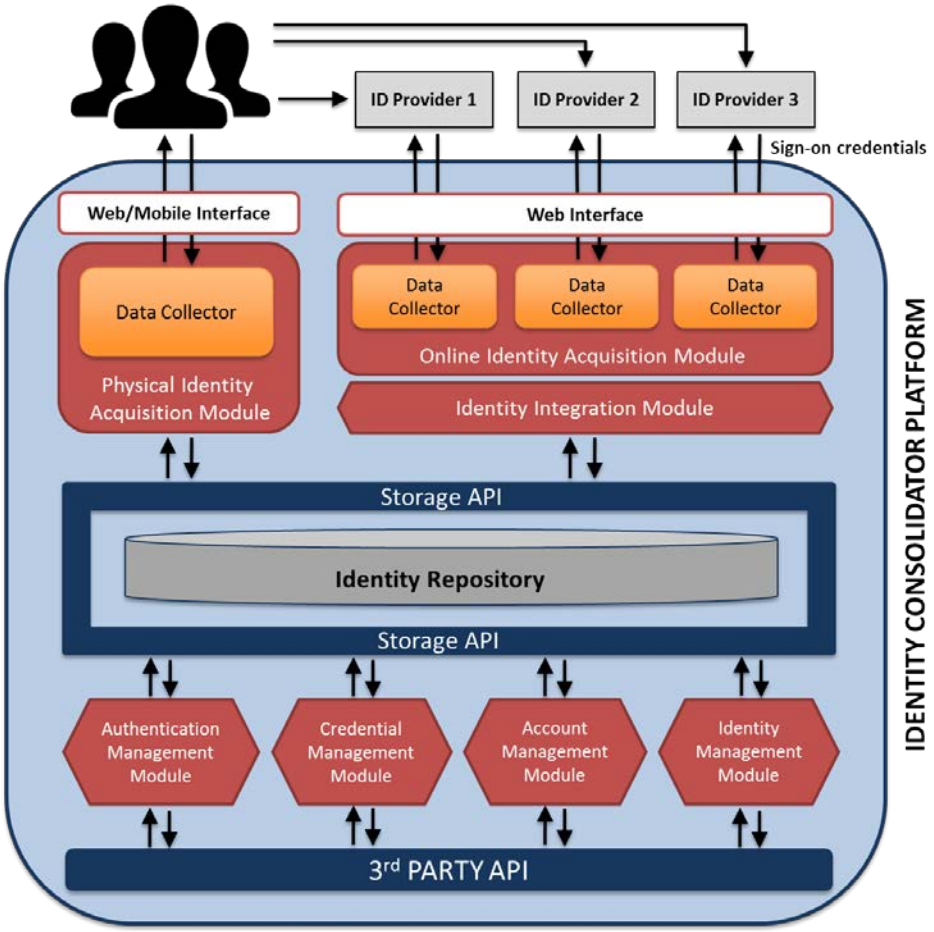

Figure 1. Identity Consolidator Platform architecture

Figure 1 depicts the architecture of the Identity Consolidator platform, which is composed by various modules that are describe below.

Initially, this service contains the **Physical Identity Acquisition and Verification module**, which is responsible for securely acquiring and verifying all the personal identity information of a user. The identity acquisition part of this module uses smart trusted-computing-enabled devices (e.g., mobile devices) which will acquire all the physical characteristics and the information included in the real-world identities of the users. Additionally, the mobile devices use trusted software paths in order to securely capture, through the device's camera, pictures of the user and of his real-world identity documents. The physical characteristics of the users (such as location) are extracted exploiting the existing technology on the devices. After their collection, all this information passes through the identity verification part of the Physical Identity Acquisition module, which runs verification algorithms and is responsible for the verification of the collected identity information of the user. The identity verification process uses existing techniques (such as OCR) and/or peer-to-peer (crowdsourcing) techniques. Automated verification is established on the acquired photos of the users using face detection, face recognition and OCR. Peer-to-peer verification using crowdsourcing techniques takes place to verify that the information on the acquired photos match the declared identity information and physical characteristics of the users. Additionally, identity verification supports verification of the collected identity information with face to face verification by trained auditors. All the verified information is stored in the **Identity Repository** as independent identity attributes. The Identity Consolidator is also in place to infer identity attributes by aggregating the collected identity information of a user.

Horizontal binding of the various online identities (e.g., the user's validated account in Facebook, Twitter, etc.) of a user is performed by the **Online Identity Acquisition module**. For each Identity Provider an online authentication process is required. The user should also give explicit authorization to each service to access the user's personal information. Following the authentication process, the attributes acquisition takes place and the collected attributes are stored in the **Identity Repository**.

Next, is the **Identity Integration module** which is responsible for standardizing and normalizing the collected users' identity information. In general the Identity Integration module is responsible for aggregating and connecting the acquired online and physical user attributes, as well as for inferring the veracity of the claimed identity attributes via means of statistical data analysis techniques. This module is also responsible for assigning confidence scores for the veracity of the attributes and for labelling identity attributes based on their origin.

Furthermore, the Identity Consolidator enables ABAC with the **Credential Management module**. The Credential Management module allows users to issue cryptographic credentials from their verified identity attributes, depending on their access needs, directly to their mobile device and use them whenever they want to prove those attributes and access a Service Provider. In the case when a user fully trusts the Identity Consolidator, is able to back-up the issued cryptographic credentials to the Identity Consolidator for recovery in case of a failure or device loss. Backing up the cryptographic credentials is also useful because it enables the Identity Consolidator to prove an identity attribute of a user while the user is offline. The cryptographic credentials issuance will be realized through the use of FiWARE [4] interface, which enables the seamless use if the Idemix and U-Prove cryptographic stacks. In addition, the Identity Consolidator runs secure protocols directly with the Identity Providers, through the use of OpenID Connect to associate a user's device with identity attribute for which it issues a cryptographic credential.

In order a user to be able to access his Identity Consolidator account, the Identity Consolidator offers various diverse authentication mechanisms for authenticating its users. Those mechanisms are offered by the **Authentication Management module**, which encapsulates a FIDO-enhanced OpenAM [5] (acts as the OpenID Connect Provider) for undertaking FIDO authentication. The Authentication Management module offers to the users various ways to authenticate such as FIDO UAF, Mobile Connect with FIDO and master password. The authentication mechanisms are described in more detail in the next Section.

The Identity Consolidator encapsulates the **Account Management module**, which is responsible to manage the status of the online accounts of a user and to expose this information to authorized parties within the ReCRED framework. It offers LATCH [6] functionality, which can be activated upon notification that a behavioral authentication has failed and this may indicates that the device has been stolen and is no longer held by its legitimate user. In this case, LATCH locks the user out of all his logged in Online Services in his mobile device. LATCH can also be activated by the user manually though the ID Consolidator. Besides LATCH, the Account Management module enables users to register a Behavioral Authentication Authority (BAA) or an Identity Provider with the Identity Consolidator, and BAAs and Identity Provider administrators to register their entities with the Identity Consolidator. In addition to this, the Account Management module is responsible to act as BAA discovery service for verifiers that require second-factor device-to-service authentication. Lastly, the Account Management module is considered as the main recovery mechanism for restoring user's accounts.

The latter module of the Identity Consolidator is the **Identity Management module** where users can view and manage various aspects of their identity. Identity Management module is subdivided in two sub-modules, the **Identity Profile Management** and the **Consent Management**. In general, it provides an identity matrix containing the different type and range of identifiers, and unique identity attributes a user can have. Apart from that, this module offers a protocol to transfer identity attributes between ID providers and at the same time guarantees the security in the transfer of such sensitive information. Also, it gives users the option to create partial verifiable profiles, which consist of selected identity attributes of a user, to be presented to verifiers depending on the context and the access control requirements. Furthermore, it allows users to define their consent for the management of their various identity attributes. In this context the users are able to create fine-grained consent management policies for their identity attributes and also view risks indicators for de-anonymization.

Besides the aforementioned modules, the Identity Consolidator also includes the Storage API and the 3rd Party API. The **Storage API** allows the previously described modules to interact (read, write, update information) with the Identity Repository. The **3rd Party API** allows all the ReCRED external entities and third parties (e.g., BAAs) to interact with the Identity Consolidator platform.

2.1 Identity Consolidator Web Application

According to the ReCRED architecture, the Identity Consolidator platform integrates the following modules under one entity:

1. Physical Identity Acquisition module
2. Online Identity Acquisition and Identity Integration modules

3. Authentication Management module
4. Credential Management module
5. Account Management module
6. Identity Management module

All the above mentioned modules are communicating with the Identity Repository through the Storage API.

In this context we have developed a web application that integrates in one common platform all the primary modules of the Identity Consolidator. This web application has been developed in PHP [8] and is running on an Apache HTTP Server [7] at CUT premises and can be accessed at https://consolidator.recred.eu. All the communications between the clients and our servers are encrypted and secured as a result of an SSL certificate that we have installed on our Server. Figure 2 below shows the landing page of the Identity Consolidator web application.

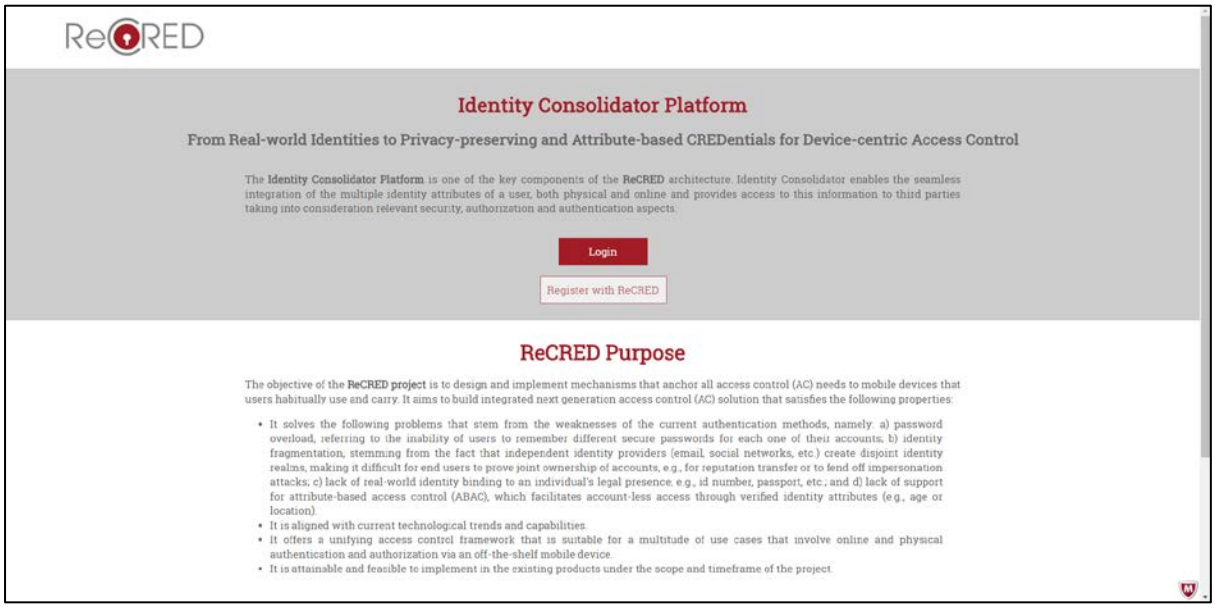

Figure 2. Identity Consolidator web application landing page

Authentication Management module is integrated with this platform so that the users are able to authenticate with the Identity Consolidator and access their Identity Consolidator account or create on by registering with the Identity Consolidator platform. Figure 3 below shows the login page that Authentication Management module offers while Figure 4 shows the register page.

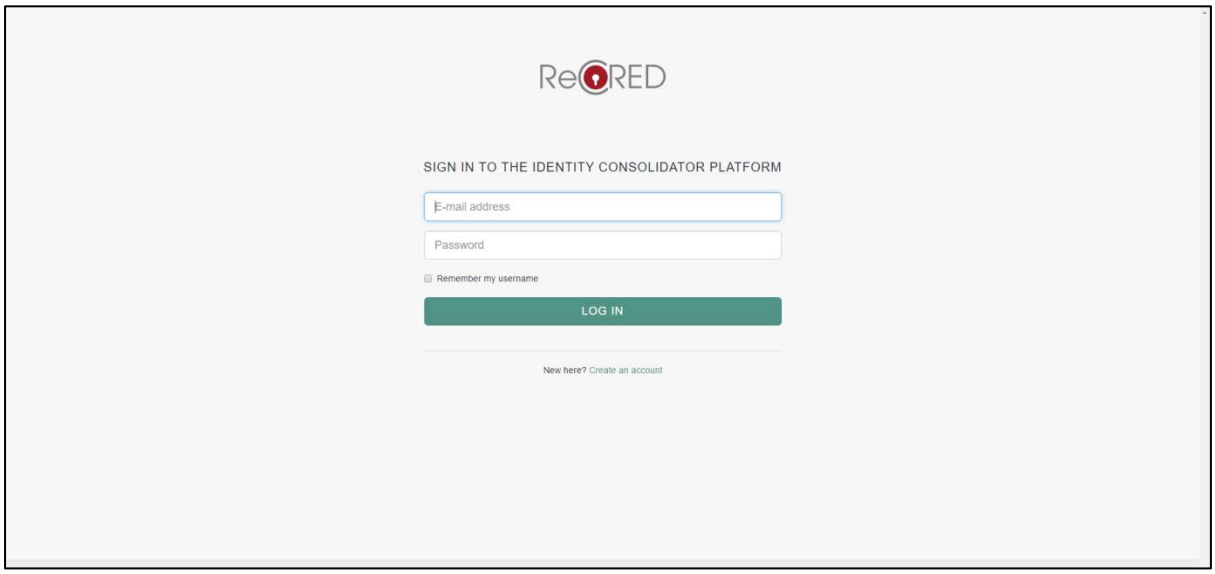

Figure 3. Identity Consolidator platform login page

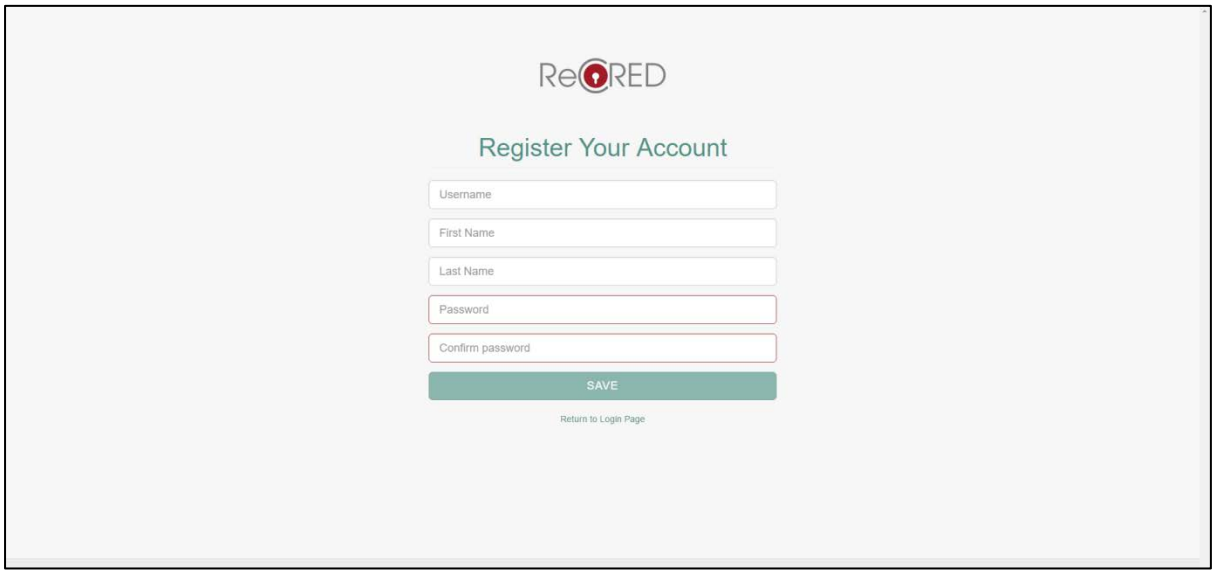

Figure 4. Identity Consolidator platform create account page

From within the Identity Consolidator main page the user is able to access all the modules that the Identity Consolidator component offers. The user is able to access this page only if he has already authenticated against the Identity Consolidator and a secure session exists. This page can be seen in Figure 5 below.

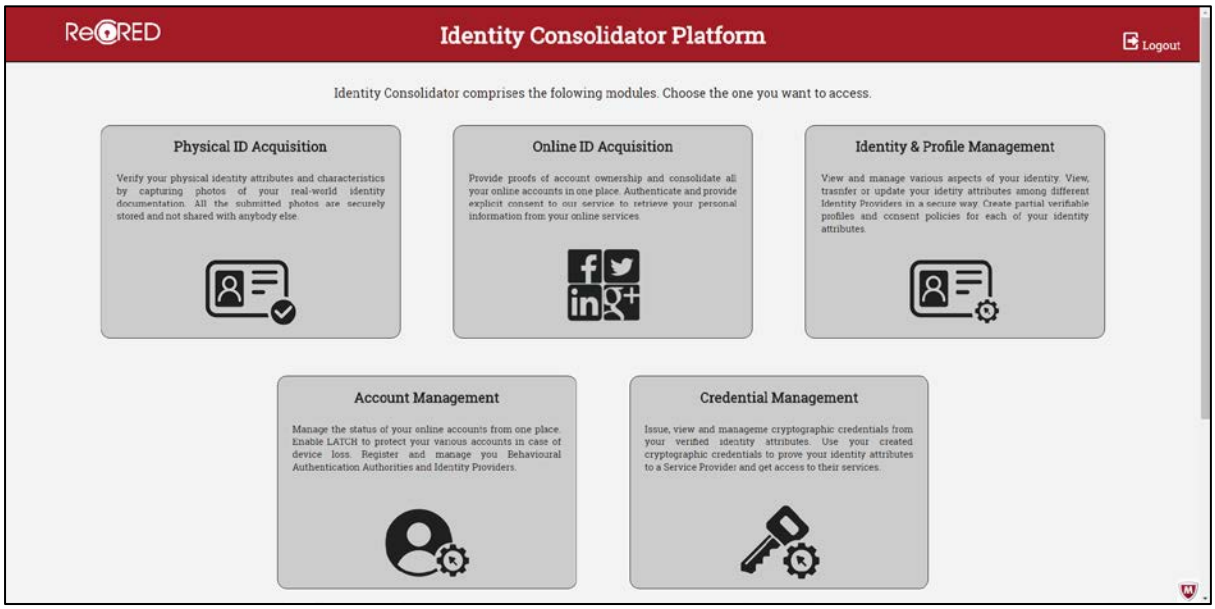

Figure 5. Identity Consolidator platform main page

3 Authentication Management Module

3.1 OpenAM

The core of the solution for implementing the Authentication Management module is OpenAM [5]. This is a web-based open-source solution that provides authentication, authorization, entitlement, and federation services. OpenAM out-of-the-box supports 25 authentication methods and offers the flexibility to create custom authentication modules based on the JAAS (Java Authentication and Authorization Service) open standard. In addition to that, OpenAM’s federation services allow federated members to securely share identity information with each other.

This platform can be extended to allow authentication using custom modules. One such module has been chosen, build upon the OpenID Connect protocol, for the ReCRED project, it authenticates a user against the ID consolidator and then, if successful, returns a success status to the OpenAM server to continue with the authentication process, providing at the same time user attributes in the http header in order to share that information with the other involved modules.

3.2 User Authentication

3.2.1 Means of Authentication and LoAs

The authentication management module supports three methods for identifying a user:

- Username (In the form of Email address) and Password
- Username (In the form of Email address), Password and TOTP through email, SMS or mobile OTP generating application
- FIDO UAF
- Mobile Connect

These correspond to the following LoA's:

- *LoA 1:* Username (In the form of Email address) and Password
- *LoA 2:* Username (In the form of Email address), Password and OTP

- *LoA 3:* FIDO UAF
- *LoA 4:* Mobile connect (when used with a TEE inside the mobile device)

The figures below are some screenshots of the implemented authentication methods captured from the Identity Consolidator web application.

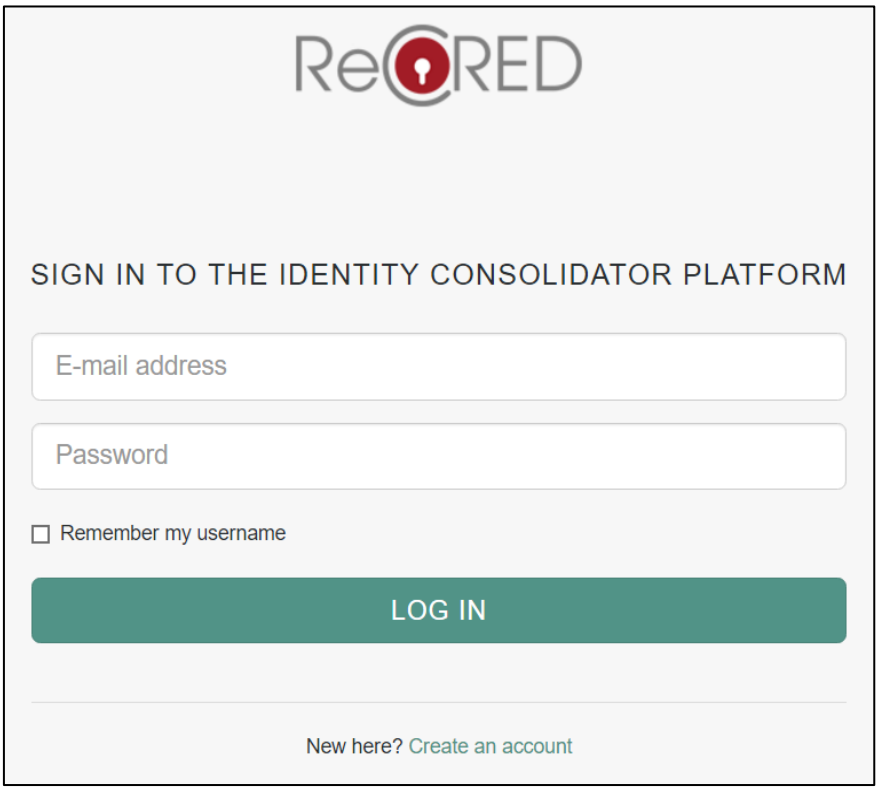

Figure 6. Authentication method 1: Username (email address) and Password

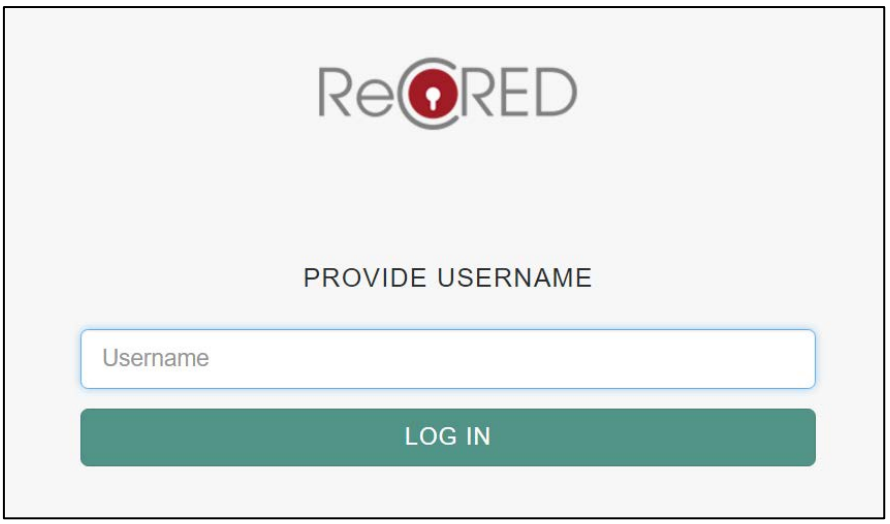

Figure 7. Authentication method 2: FIDO UAF

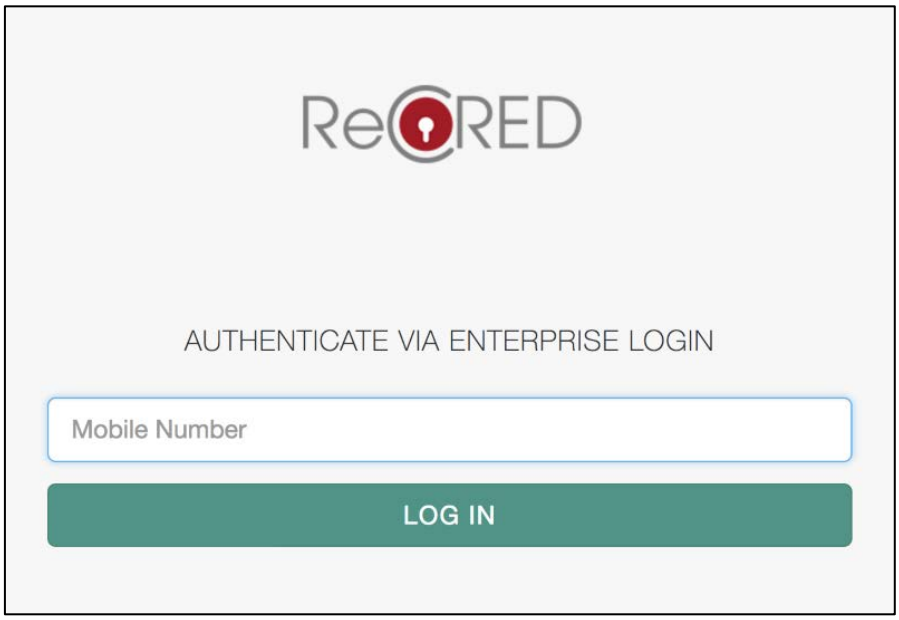

Figure 8. Authentication method 3: Mobile Connect

3.2.2 Tentative Access and Access promoting with Behavioral Authentication Authorities

Tentative Access Mode:

If the user is unable to access the IDC due to the loss or theft of his device, the system provides him with an option to login using his master password. If the user is incapable to provide the master password, the user can initialize an alternative login procedure based upon a series of security questions. By successfully login to IDC the user has tentative access to the list of the ID provider only, for a limited time. The system contacts all the BAA authorities to verify the identity of the user.

User Authentication on Tentative Access to Behavioral Authentication Authorities:

The IDC, running on the tentative mode contacts all the Behavioral Authentication Authorities available for the user through OpenID Connect. After a BAA is contacted using a tentative mode, the BAA will runs in tentative mode and will request the backup credentials, in order to provide full access. If the user is incapable to provide the backup credentials, the BAA initializes the authentication procedure based on the behavioral signatures of the specific user, for a minimum time. If the behavioral signatures match the user, the BAA will authenticate the user to IDC. A notification is then also sent to the user.

Access Promoting Based on BAAs Response:

After the Identity Consolidator collects the responses from the BAA of the user, it decides if the user is in tentative mode, if it is the real user and provides full access or else we are in the presence of an imposter and then locks the account.

The responses come from all BAAs OpenID Connect, based on those responses, the probability that the user is the legitimate one is calculated.

According to the defined Identity Consolidator Policies for account management, if the IDC is highly confident that the user is the legitimate one then it will be able to provide full access to the user

3.2.3 Failover Authentication Mechanisms

The failover authentication mechanisms ensure that in cases where the user loses access to the device and wants to restore his credentials on a new device, utilizing:

- Mobile Connect
- Behaviour Authentication Authorities

Normally a user can access the ID Consolidator platform just by proving the possession of the FIDO cryptographic key, which is bound to his device. In the case of device loss or theft, which we treat similarly because loss may actually be theft, the following two failover authentication options are offered in order to recover access to the Identity Consolidator:

1. If the user has another device with the credentials available, he can login to the IDC and disable the access for the stolen device. No more steps are needed.
2. If the user has no other device, he logs in to the IDC with his secure master password or security questions, and selects whether his device is lost/stolen or damaged, and whether he still has possession of his old SIM card. In doing so, he is granted only temporary and tentative access, which provides limited functionality. Tentative access is the Level of Assurance (LOA) LOA-1. In addition, the IDC invokes Latch to lock all the IDP accounts managed by the IDC. In particular, he cannot view or restore credentials, and he cannot view the attributes. He can only see who his IDPs and BAAs are. Importantly, he also has the ability to lock the account so that the thief cannot access the IDP via the stolen device. Subsequently, the IDC acts as Service Provider authenticating the user through a Telco Identity Provider via Mobile Connect. Because the user no longer has his device, he cannot use FIDO UAF to authenticate to Mobile Connect IDP, he has to authenticate to the IDP via SMS using his SIM card. In the case of a stolen or lost device, to ensure, the access attempt is performed by the legitimate user the IDC needs to confirm with the Mobile Connect IDP that the given Phone and his phone was reported lost and a new SIM was issued, via OpenID connect. In the case where the user declares the mobile damaged and still has possession of his old SIM card the IDC does not need to communicate with the Mobile Connect IDP. By providing the IDC Master password and proving his identity to the Mobile Connect IDP, the user gains full access to the IDC to restore his credentials on a new device and reset the credentials to the compromised accounts.

3.2.3.1 Tentative access mode with Backup credentials

A user can recover access to their Identity Consolidator account using a preset Master password that they set when they first create an account to the ID Consolidator.

3.2.3.2 Tentative access mode without Backup Credentials

IF a user does not remember or does have their Master password, the user is also able to gain access to their Identity Consolidator account by entering some challenge response security questions.

The platform randomly selects three security questions from the ones stored against that user's profile. The user enters their response to these questions, which is processed according the rules below. The three user entered responses are then checked server side against the ones held on file against the user. When all three security questions are correctly validated, the user is able to access the Identity Consolidator in failover mode.

Security answers are processed as follows:

- All spaces are removed
- Answers are converted to uppercase
- Punctuation, such as Full stops, commas, exclamation marks are removed from the answers. (Only valid characters are 0-9 A-Z)
- The answers are stored in the Identity Repository of the Identity Consolidator platform as hashed values using the Storage API

3.3 User Enrollment (Registration)

3.3.1 Means of Registration

Currently, the Authentication Management module supports two methods for enrollment:

1. When authenticating through the ReCRED Identity Consolidator against an existing Identity Provider
2. By explicitly creating an account in the Identity Consolidator

A user can create an account in the ReCRED’s ID consolidation service, which acts as a primary IdP so that he can make use of all the features that ReCRED offers (such as Physical Identity Acquisition, binding of multiple online accounts, etc.).

The Identity consolidator checks that the user that wants to register is unique and there is no registered user with the same personal information.

The service verifies that an email confirmation has been sent to the email address that the user declared and that all the declared identity information of the included passport and user’s photos are valid. In order to do that it will use the acquisition and verification features of the Physical Identity Acquisition module.

The user is able to set up some basic configuration for his consolidator account (e.g., account lock policies, whether he trusts the consolidator, etc.) and makes use of OpenAM's backend infrastructure because the Authentication Management Module is based on OpenAM. The IDC acts as the Identity Provider with which the user explicitly registers via this functionality.

The user's registration involves the registration with the FIDO server component that runs on the Identity Consolidator.

During registration, the user has to create a complex, of at least 15 characters password, which we refer to as the Master Password. Additionally, the user will be given security questions and will provide answers.

The User can also register to the ID consolidator by authenticating against Mobile Connect. The Identity Consolidator contacts the GSMA gateway to identify the Telco provider for that specific phone number and verifies the credentials of the user using their own Mobile Connect approved method.

After a successful verification from the Telco provider(s) the user will be considered authenticated at LoA 4 If the MNO has verified the user using a hardware token.

3.4 User Account Deletion

Authentication Management module also offers the ability to the users to delete their ReCRED account. A user’s account will be marked with the highest level of assurance that the user has

verified themselves against the platform too. A user that has been verified to LoA 1 (Level of Assurance 1 such as using just Username and Password) will be marked in the system as only ever having achieved LoA 1. A user, who has verified themselves to LoA 4 will have their account identified in the platform as having been verified all the way up to LoA 4.

When a user wants to remove their account from the platform, they must first verify themselves to the same level that their account has been flagged as having achieved.

Once a user has verified themselves to this level, they will then be presented with an option (Disabled / Greyed out until the appropriate level is achieved), to delete their account.

This operation can be performed by accessing the ReCRED Identity Consolidator User Account Management Module.

When a user selects to remove their account, a confirmation email will be sent to that user’s registered account. This email will contain a link that the user can click on to take them to a final confirmation page.

Upon confirmation all user attributes for that user will be removed from the storage API. The users identity will be marked as removed (But it will be retained to enable correlation between logs / audit information and a user’s identity).

Authentication management module supports two levels of user account deletion:

1. User self de-registration
2. Request for full purge

The difference between these two processes is that with self de-register the user information and their access logs / auditing will persist in the platform, whereas with a full purge, all auditing and logging related to the user will also be removed from the platform.

3.5 Identity Attribute value transfer among different value transfer

The identity attributes are transferred to the Identity consolidator via the OpenID Connect protocol. The claims supported by the MNO will be queried and all claims will be requested upon initial authentication of the user. If a user has enabled lowest degree of privacy within their profile settings, all the attributes that a user shares, are stored within the IDC and flagged with the LoA that the user achieved during authentication time. This is, if a user authenticates to the Mobile Connect Provider with a TEE enabled device, the users LoA would be classed as LoA 4 and therefore the attributes returned will be marked as LoA assured to Level 4 within the Identity Consolidator’s Identity Repository.

If a user chooses not the share all attributes with the Identity Consolidator, the claims that are shared will be stored in the case of least degree of privacy.

4 Credential Management Module

One of the most important features provided by the ReCRED framework is the implementation of a user-friendly Attribute Based Access Control (ABAC) architecture, integrated in the system, allowing the user to preserve the privacy of its attributes. The Identity Consolidator is in charge to collect, validate and store users’ identity attributes (e.g. physical identity attributes) in order to make them manageable by the user and available to be used by service providers. In order to make such identity attributes usable for access-control purposes, an additional module is required. Indeed, the

Credential Management Module is designed to “translate” identity attributes acquired by the Identity Consolidator to valid credentials for the two Privacy-Preserving ABAC (PABAC) cryptographic technologies adopted by ReCRED, i.e. Idemix and U-Prove.

The Credential Management Module of the ReCRED Identity Consolidator is a special instance of the Issuer component of the ABAC architecture: it is in charge of issuing credentials to the user, compiling the credential attribute values from verified identity attributes stored by the Identity Consolidator. These values are retrieved by the Credential Management Module by using the Storage Module API.

The design of the Credential Management Module is based on the architecture in figure 9, proposed by the IRMA project [15], in which the following components are provided:

- **IRMA API Server**: this is a server that sits between the user device and service or identity providers on the other hand. It handles all specific cryptographic details of issuing credentials and verifying disclosure proofs on behalf of the service or identity provider.
- **Application Server:** this is the service or identity provider server implementing the interface between the user and the IRMA API Server (cryptographic API)**.**
- **User Device:** this is the user device used to access services using IRMA application and enforcing ABAC by means of Idemix credentials.

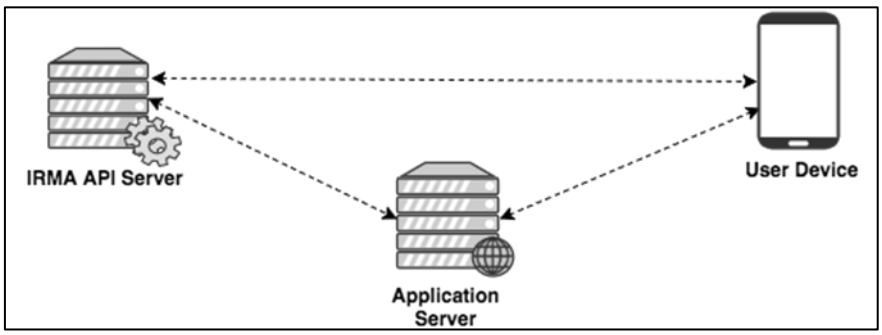

Figure 9. IRMA Architecture

The design of the IRMA architecture allows to have a cryptographic implementation clearly separated with respect to the service/identity-provider specific application level.

4.1 PABAC API Server

The ReCRED PABAC API Server is the server demanded to run the cryptographic functionalities for Idemix and U-prove. The following sections provide an overview of the issuance protocol performed when issuing Idemix and U-prove credentials.

4.1.1 Idemix API Server

The issuing protocol is triggered when a user requires the issuance of Idemix credentials from an identity provider.

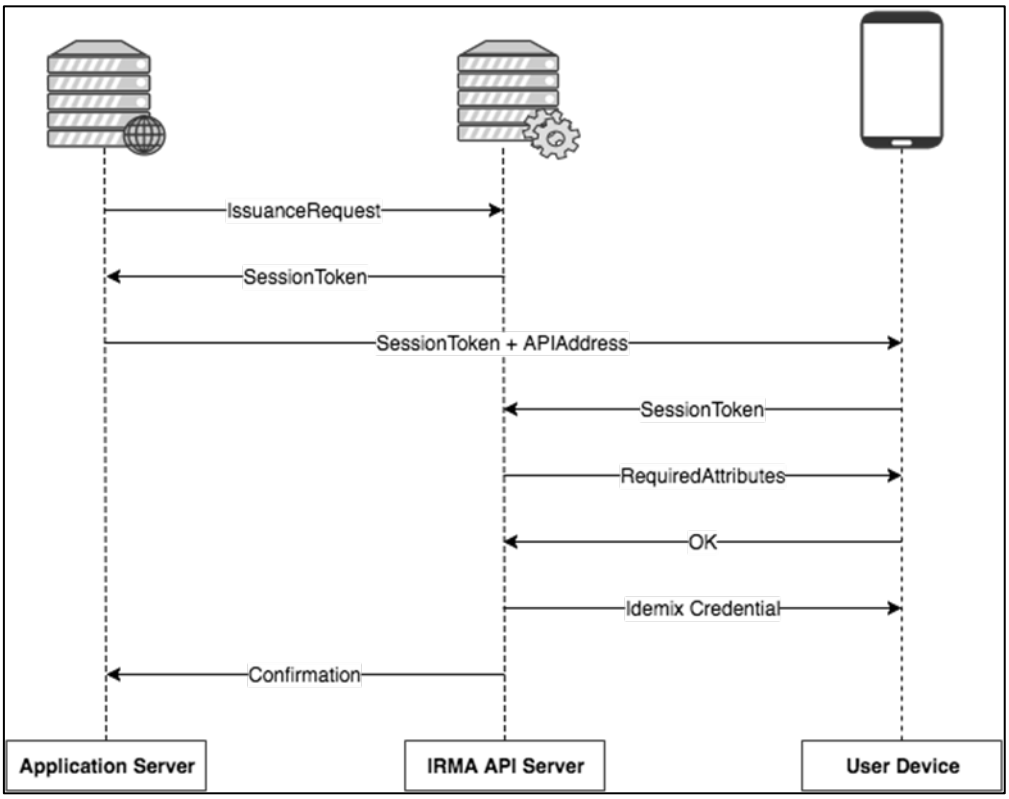

Figure 10. IRMA Issuance Protocol

The issuance protocol’s phases reported in Figure 10 are the following:

1. The Application Server submits the **issuance request** (triggered by the user) to the IRMA API Server;
2. The IRMA API Server provides to the Application Server a **session token** to be provided to the User Device together with the end-point that the user should contact to require the **credential**;
3. The User Device accesses the end-point provided by the Application Server by using the **session token**;
4. The User Device receives the Idemix credential issued by the IRMA API Server.

At the end of the protocol the User Device stores the received credential in a secure storage on the device. Note that, also in this case, the Application Server is not required to run any cryptographic operations.

4.1.2 U-prove API Server

The U-Prove API service contains a U-Prove engine which performs all the cryptographic operations during the issuance protocol. For the communication between client and server the U-Prove objects are serialized using the JSON format. The following image illustrates how the data is being sent over the network from the client to the server.

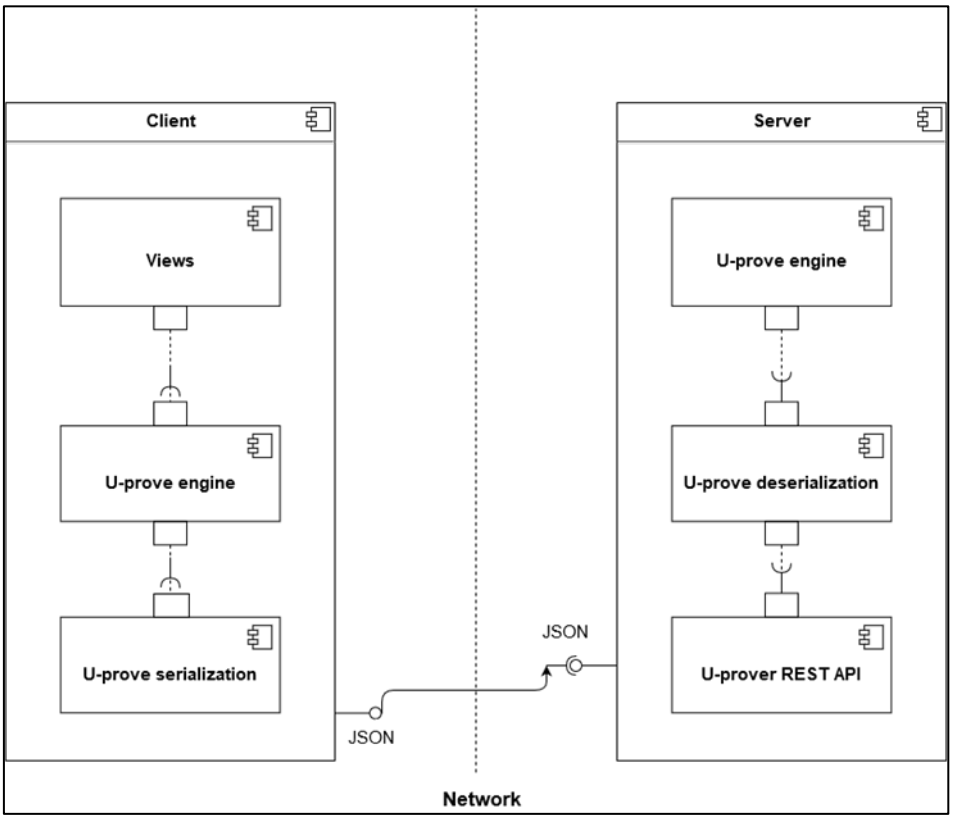

Figure 11. U-Prove client-server architecture

The output of the issuance process is a U-Prove token containing the following information:

1. An application-specific unique identifier for the Issuer parameters under which this token was issued;
2. the *Token Information* field is used to encode token-specific information which is always disclosed to Verifiers, such as token usage restrictions, a validity period, or any other metadata. If this information is significant then the implementation should provide the particular implementation for the decoding process in the verification process;
3. the *Prover Information* field is used to encode token-specific information which is always disclosed to Verifiers, such as token usage restrictions, a validity period, or any other metadata. This information is also disclosed to the Verifier;
4. The signature provided by the Issuer.

For keeping track of all the U-Prove clients that are accessing the U-Prove server, the API generates a session key for each client that tries to obtain a U-Prove credential. When a user tries to generate multiple U-Prove credentials (tokens) all the following is being verified:

- the U-Prove session key which the user presents exists in the database and the session key that the user provided is not expired;
- the U-Prove client together with the valid session key previously validated makes a valid request to the U-Prove server. This aspect must be checked because the issuance process in U-Prove takes more than one step;

- the identity of the issuer doesn’t exist in different states at the same time during the issuance process. More exactly the same issuer should not generate a U-Prove credential for more than one user at the same time;
- the number of credentials the user wants to obtain is smaller than the maximum number of credentials the server can provider at once.

4.2 Credential Management Application

One of the aims of ReCRED is to support an Attribute Based Access Control Infrastructure that enables the use of anonymous credentials, based on technology at the state of the art such as Idemix and U-prove. These technologies provide the core cryptographic functionalities of the PABAC architecture but have different requirements for what concerns their interfaces. The FIWARE project provides a common interface between Idemix and U-prove by means of the Privacy Generic Enabler [4]. In the ReCRED ABAC architecture we aim at joining together the ReCRED results with the outputs from the IRMA and FIWARE projects to provide an integrated PABAC architecture, providing both Idemix and U-prove credentials support.

The Credential Management (CM) module aims at supporting the issuance of anonymous, PABAC credentials derived from the information acquired by the Identity Consolidator. The PABAC credentials are issued to the User Device and received and stored by the ReCRED Wallet app.

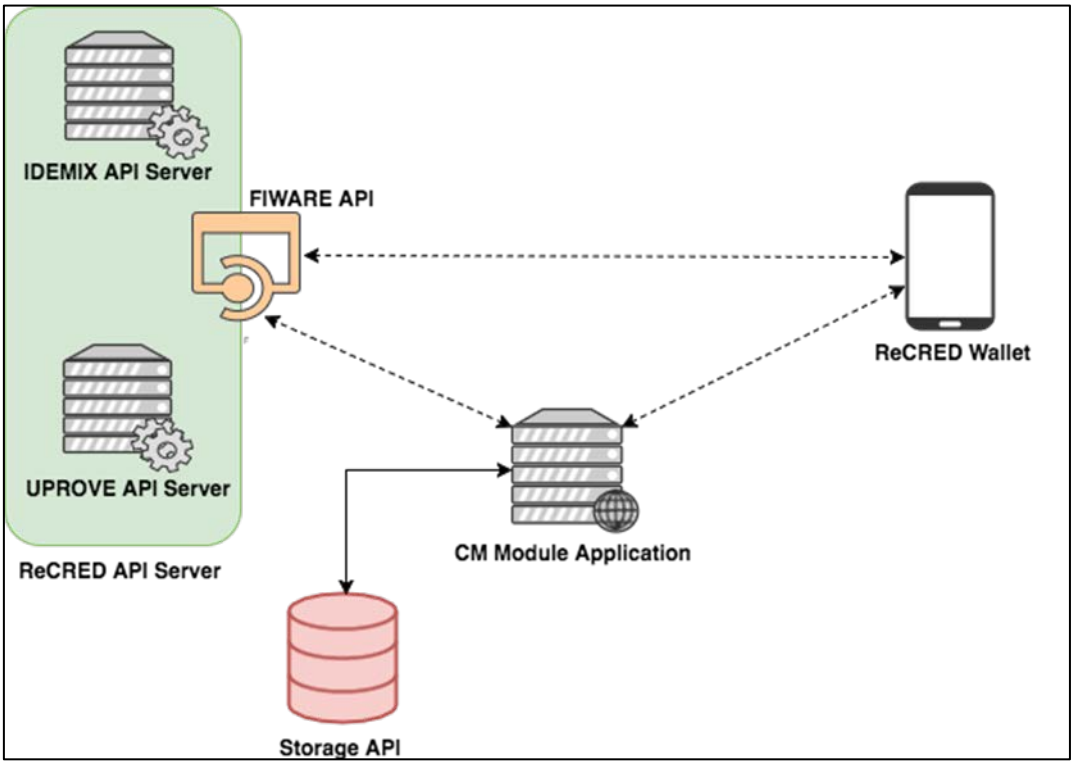

Figure 12. Credential Management module submodules and interactions

As shown in figure 12, the Credential Management Module is composed by two main sub-components reflecting the Issuer component structure:

- **ReCRED API Server**
- **Credential Management Module Application**

4.2.1 Credential Management Module Application Frontend

The Credential Management Module frontend is a web application that can be used by the user to list the credentials that can be issued to her by the Identity Consolidator. The application is linked to the Storage API back-end, in order to be able to retrieve identity attributes to be used in the credential generation, and to the ReCRED API Server, for the actual credentials issuance.

We provide below a list of operations which can be performed by the user on the Credential Management Module frontend.

4.2.1.1 List Compiled Credentials

Once the user logs into the CM Module Application Frontend, the application retrieves through the Storage API all attribute values that match the fields of a set of pre-defined credentials. These attribute values are used to fill the fields of these credential structures, i.e. to compile the credentials to be issued. These credentials are listed to the user as shown in figure 13. For each credential the following information is presented:

- **Credential Name:** is the name of the credential represented in the following dotted format:
  - **<Domain>.<Issuer>.<CredentialName>** where**:**
    - **Domain:** is the application domain, which in the ReCRED project case will be **recred;**
    - **Issuer:** is the issuer name, which in the case of the CM Module is **RecredIdentityConsolidator**, since the issuer is the Identity Consolidator;
    - **CredentialName:** is the name that identifies the credential (e.g. usernames, userphone)
- **List of attributes:** is the list of attributes defined in the credential structure and for which the value has been retrieved through the Storage API.

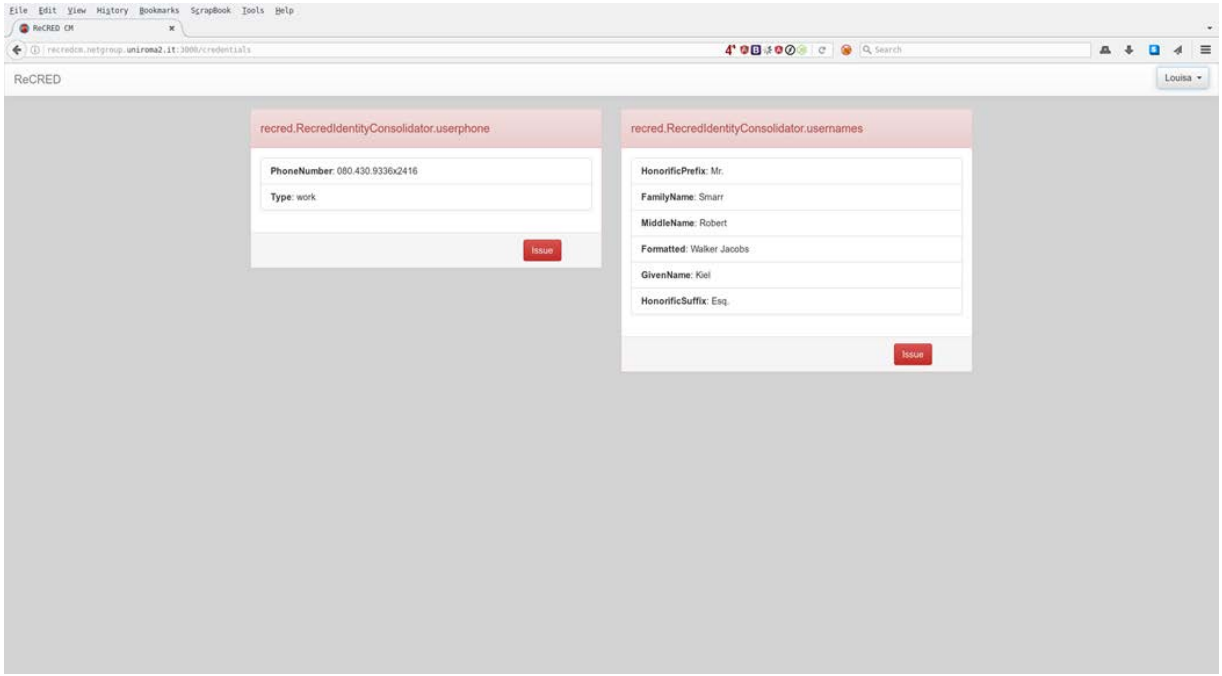

Figure 13. List of available credential ready to be issued

4.2.1.2 Issue Credentials

From the credentials list, the user is able to select the credential that wants to be issued to her own device. Once it selects the “Issue” button for a specific credential, the CM Module Application contacts the ReCRED API Server to obtain a session token for the issuance of the selected credential. Once the Credential Management Module application receives the session token from the API Server, it provides to the user a QR Code, as shown in figure 14, that contains a JSON text with the following information:

- **Session Token**, e.g., eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9
- **URL of the API Server**: e.g., https://api.recred.eu/abac
- **Protocol Version:** e.g., v1

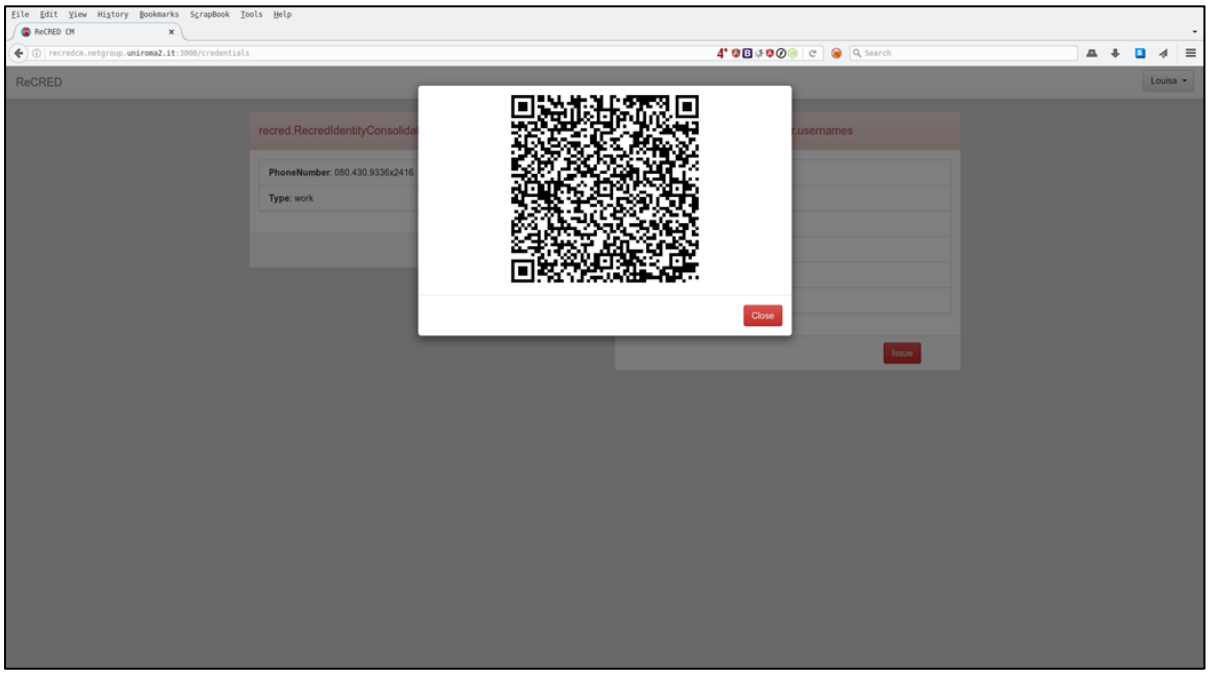

Figure 14. Credential Issuing: Link of the User Device to the API server by means of QR Code

The QR code allows the user to receive the information about the session with the API Server in order to execute the Issuance protocol directly with the API Server itself. By using the ReCRED Wallet Application shown in Figure 15, developed for Android devices, the user can scan the QR code and receive such information.

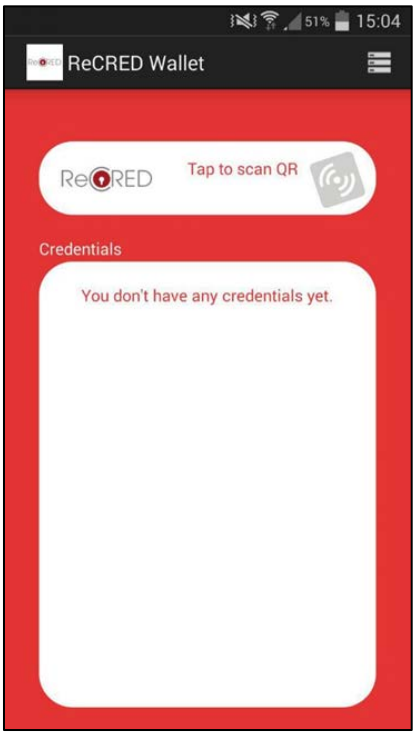

Figure 15. ReCRED Wallet Application

At this point the ReCRED Wallet Application and the API Server will run the Issuance Protocol in order to perform the issuance of the previously selected credential. As shown in figure 16, the user will be asked to provide the consent to receive the credential and the attributes, together with the value of each attribute, displayed to the user by means of a dialog. The user can deny the consent, and the issuance of the credential will be stopped. Otherwise the credential will be issued to the user and will be stored in the user device itself.

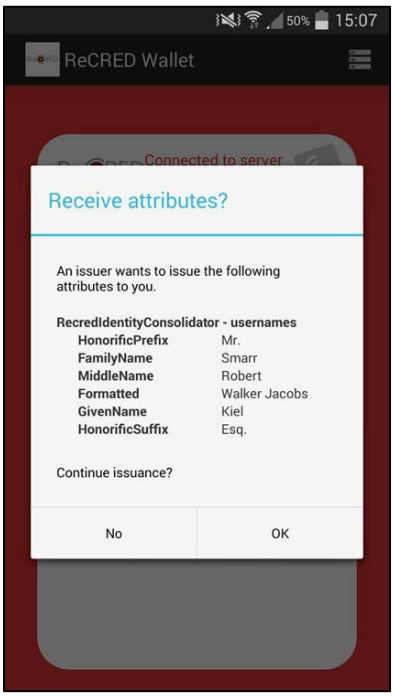

Figure 16. ReCRED Wallet Application: issuance consent display

Once the issuance of the credential is completed, it is added to the list of credentials owned by the user and issued to her device (figure 17).

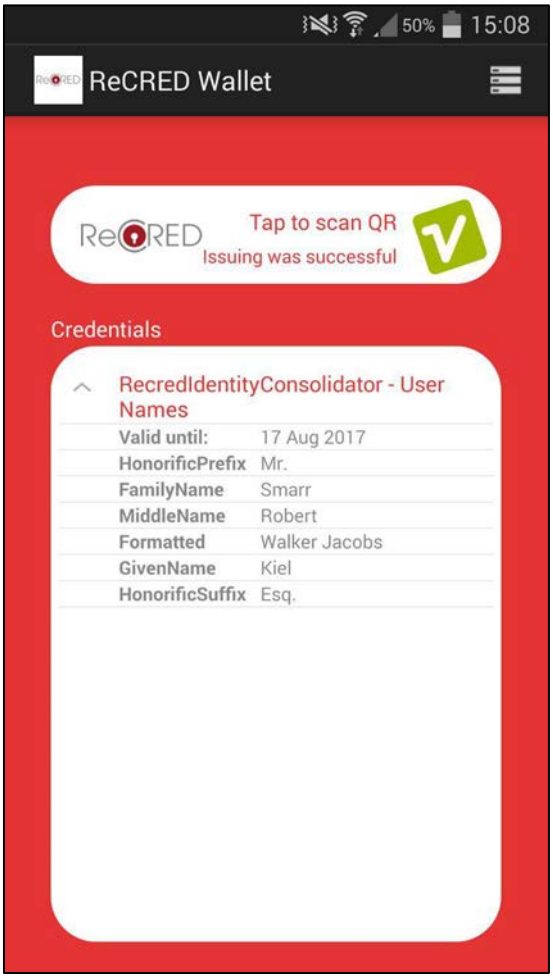

Figure 17. ReCRED Wallet application: Credential list and details

4.3 Credentials Backup and Restore

According to the ReCRED architecture, Identity Providers may issue credentials that ReCRED users store in their mobile devices. These credentials are stored in the device and are encrypted using the TEE if it exists in the mobile device.

The ReCRED suite of mobile applications includes Credentials Backup & Restore Android application dedicated to the handling of ReCRED credentials. The ReCRED user authenticates to the mobile device using FIDO and a biometric key such as a fingerprint.

The following functionalities are currently offered:

- The app is offered as a library to be incorporated in the general ReCRED mobile application.
- The application shows all the credentials available in the device and can be encrypted by TEE if it exists.
- The application can connect to the Identity Consolidator server for fetching backed up credentials using the 3rd Party API.
- The credentials in the device can be backed up in the Identity Consolidator server.

4.3.1 Main menu

In the main menu of the Credentials Backup & Restore application a user can see all the local cryptographic credentials stored in the device as well as all the remote credentials of the user that are available for backup.

Yet the way for authenticating the user of the device to the remote server for having access to the backup data has to be resolved. Currently the remote backup service is OAuth v0.2 secured.

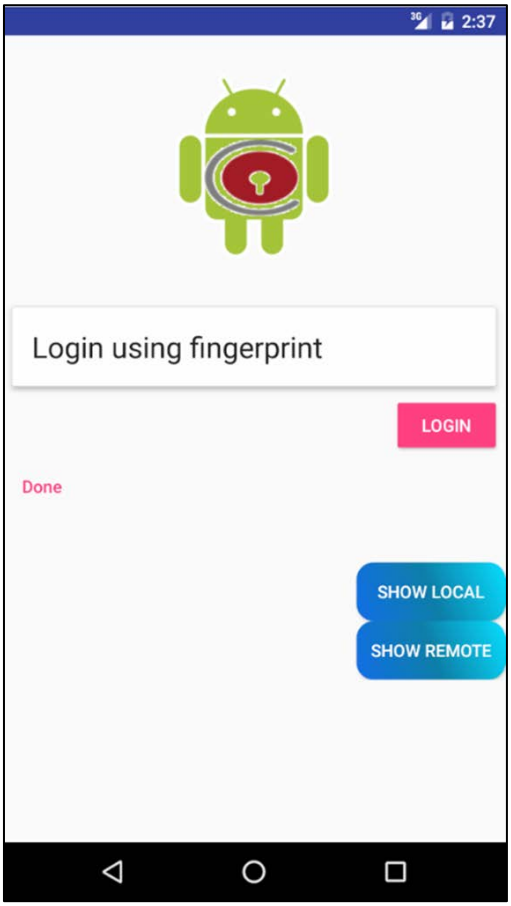

Figure 18. Credential Backup & Restore mobile application: Main page

4.3.2 Cryptographic Credentials stored on the device

The cryptographic credentials that are locally stored in the device are presented to the user in a list. Currently, they are fetched from a local SQLite database of the application. This SQLite database is also encrypted. Apart from the credentials data some metadata are also stored. The SQLite database file is located in the folder of the device which is specific for this application’s identifier. Therefore the way which credentials will be stored at the device has to be defined so that we can proceed with accessing them properly as ReCRED defines so. This has also to be unique and common between all ReCRED mobile sub-applications in order to have a common basis. For example, in order to use a cryptographic credential to prove an identity attribute to a Service Provider.

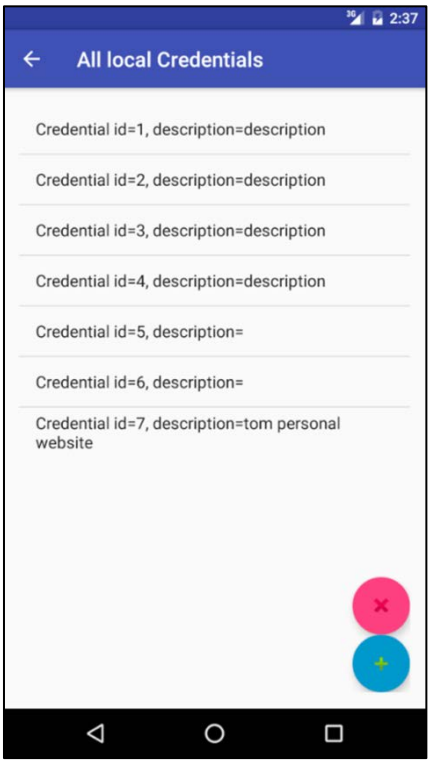

Figure 19. List with the Cryptographic Credentials stored in the mobile device

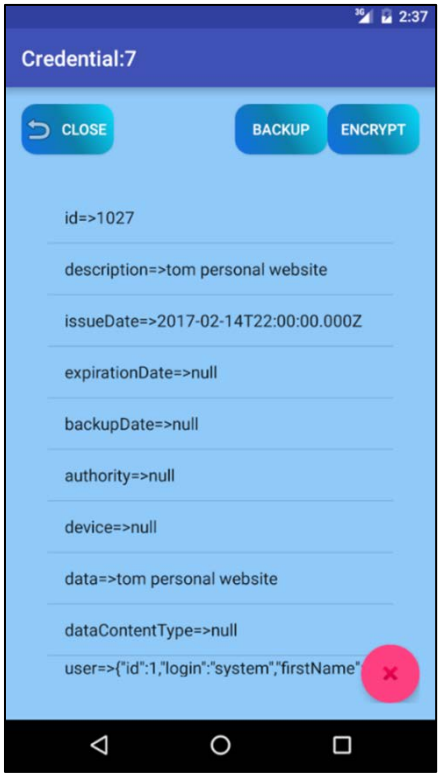

Figure 20. Page that shows to the user the details of a cryptographic credential

In the details of a locally stored cryptographic credential a user can see all the data and metadata of this credential. Additionally, the user can choose to encrypt the data and backup the credential to the Identity Consolidator server.

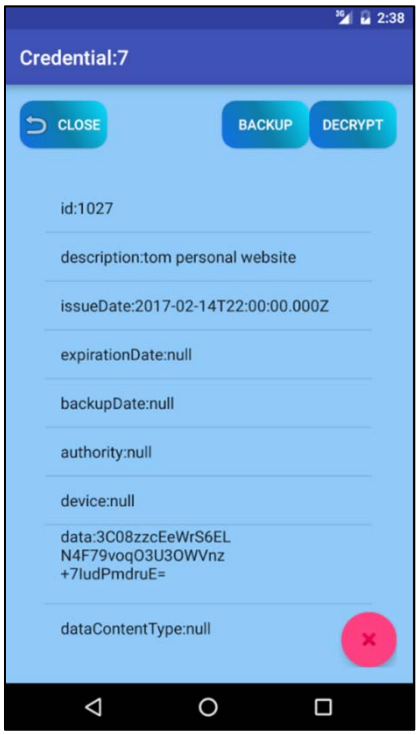

Figure 21. Details of an encrypted cryptographic credential

4.3.3 Cryptographic Credentials backup in the Identity Consolidator server

Currently, the details of the cryptographic credentials that are backed up in the Identity Consolidator server are presented to the user in a list. The user can choose which credentials to load and/or download and store to his mobile device.

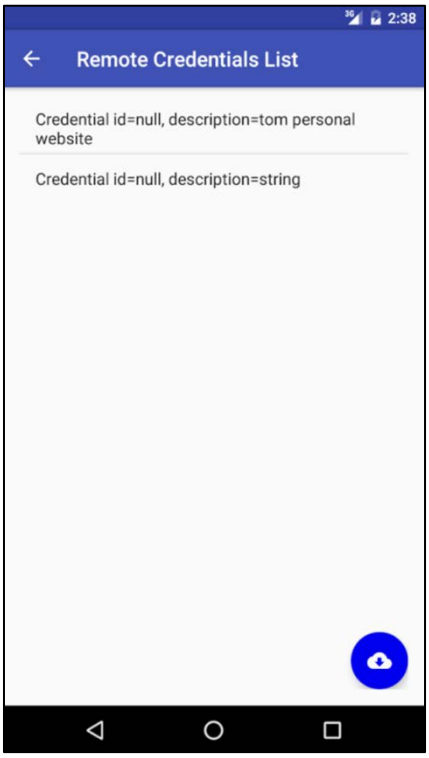

Figure 22. List with the Cryptographic credentials that are backed up in the Identity Consolidator

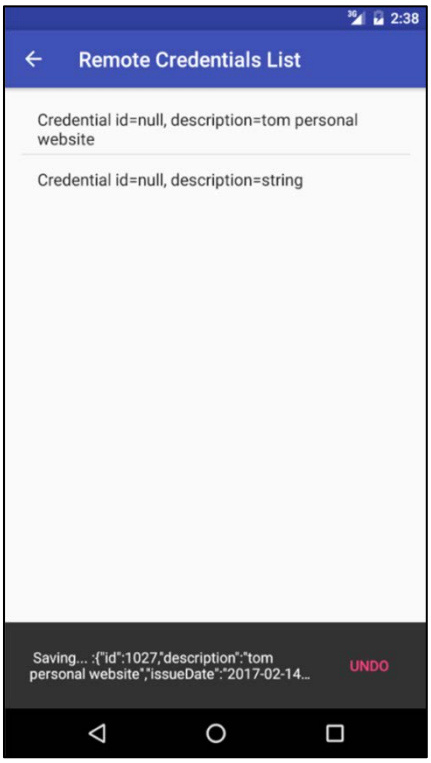

Figure 23. Process of fetching a cryptographic credential from the Identity Consolidator server

5 Account Management Module

The Account Management Module (AMM) within the Identity Consolidator is responsible for managing the status of online accounts and to expose this information to authorized parties within the ReCRED framework.

Account Management module is divided into several tasks that reflect the functionality that it provides. In this section a description of the various tasks and the business logic behind them is presented.

Furthermore, Account Management module comprises the following functionalities:

1. Behavioral Authentication Authority registration
2. Identity provider registration
3. Service provider registration
4. Account locking mechanism
    a. Latch
    b. Policies
5. Account recovery mechanism
6. Mobile Connect
7. Degree of Privacy Selection and Enforcement

Note that the Account Management module User Interface is at an early development stage and will be professionally implemented by UX and UI experts after all the functionality has been developed. We have implemented the web Front-end and the back-end of the Account Management module using the following technologies:

- Material UI [34]
- Moment.js [35]
- Java Web Service and Apache Olingo [36]

5.1 Behavioral Authentication Authority registration

Behavioral Authentication Authority (BAA) is a trusted party for ReCRED and plays the important role of providing behavioral second factor authentication to the ReCRED users. A number of BAAs can be registered to ID Consolidator (IDC) and IDC users are able to select them for providing them with the 2nd factor authentication. The AMM offers the API for storing to the IDC repository the information of the BAA and also offers the users the ability to view a list of the available BAAs and choose the ones that they will be able to authenticate them.

The process for registering a BAA (Behavioral Authentication Authority) to the ReCRED platform (ID Consolidator) is the following:

After the BAA has established a trusted relationship with the IDC (by following an offline procedure) the BAA uses the provided API to insert their data to the IDC. A web form is also available for this task. The information available for each BAA is following:

- *BAA info:* a name or description for the BAA
- *Behavioral Profile Type:* the behavioral profile type that the BAA supports (e.g., Gait, typing, browsing etc.).
- *Active:* this indicates that the BAA is active and the ReCRED users can use them. This field will be updated by the IDC admin
- *Updated:* the date that the registration request was submitted

After the BAA has inserted its data to the IDC, the IDC admin is able to activate or deactivate a BAA, add a new one or remove an existing one (Figures 24 and 25).

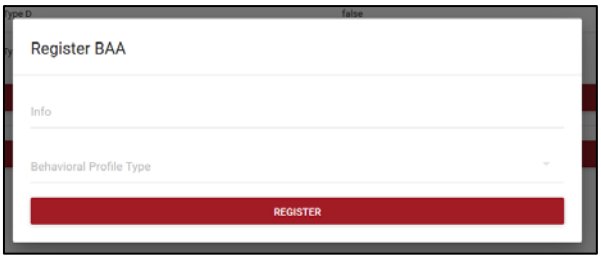

Figure 24. Register a Behavioral Authentication Authority with the Identity Consolidator

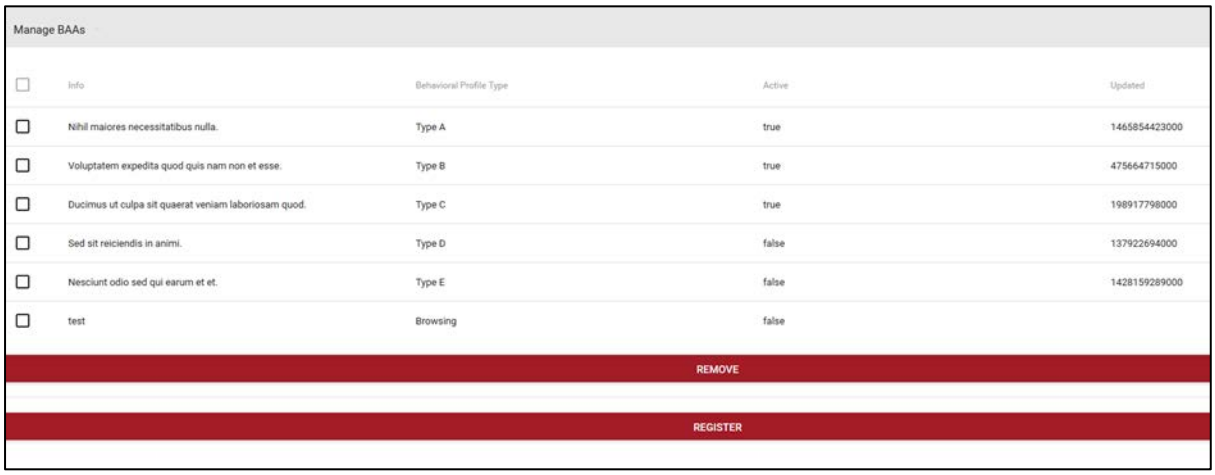

Figure 25. BAA management page of the Account Management module web front-end

The user is able to view the available BAAs that are registered in the Identity Consolidator and can enable or disable them from tracking him as shown in figure 26. User can select the BAAs that he wants to enable or disable and save the changes.

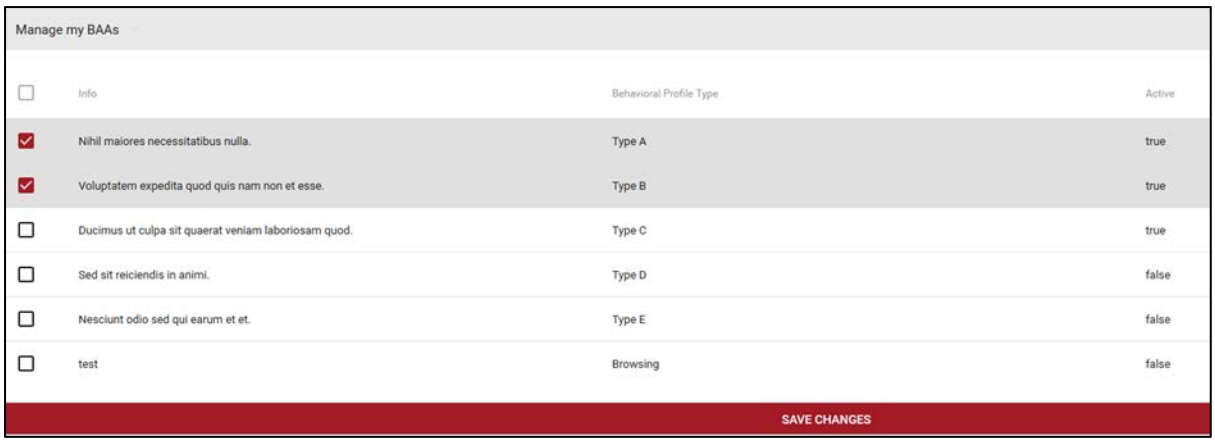

Figure 26. Enable / disable BAAs to be used

Related API calls:

***Get all BAAs**

@Path("/api/v1/baa")
@GET

***Create a BAA**

@Path("/api/v1/baa")
@POST
@Consumes(JSON: {BaaInfo, BehavioralProfileType})

***Delete a BAA**

```
@Path("/api/v1/baa/{id}")
@DELETE
```

***Get all BAAs for a user account**

```
@Path("/api/v1/user_account/{id}/baa")
@GET
```

***Register a BAA to a user account**

```
@Path("/api/v1/user_account/{user_account_id}/baa/{baa_id}")
@POST
```

***Delete a BAA from a user account based on the ID of their join**

```
@Path("/api/v1/user_account/baa/{id}"
@DELETE
```

5.2 Identity Provider registration

The IDC offers a process to statically register Service Providers and Identity Providers. Anybody may make a registration request, however approvals are done manually by the IDC administrator who has the option to reject or approve the registration request.

To submit a request for Service Providers and/or Identity Providers onboarding, a validated email address and ownership of the requested domain must be shown. This is done via a randomly generated string that must be placed on the root of the requested domain in the file that the IDC will query before releasing the request to the IDC Admin for approval.

IDC admins will provide to the Identity Provider the way that will securely connect to IDC and the IDC will send its data to be stored in the IDC repository. A user can then register his Identity Providers’ account to the Identity Consolidator and transfer their attributes to the IDC.

The process for registering an IDP to the ReCRED platform (ID Consolidator) is the following:

1. Identity Provider follows an offline procedure to setup a trusted relationship with IDC and the IDC admins provide the IDP the secure way to connect to IDC. The IDP use the AMM API to insert their data to the Identity Repository while an admin screen is also available as shown in figure 27. The data stored are the following:
    - *Name:* the IDP name
    - *Description:* a brief description of the IDP.
    - *Level of Assurance:* Levels of Assurance defined by the National Institute of Standards and Technology (NIST), in terms of the likely consequences of an authentication error.
    - *Supported Attributes:* includes a JSON object that will contain all the attributes that the IDP supports
    - *Endpoints:* includes a JSON object that will contain the endpoints that the IDP provides.

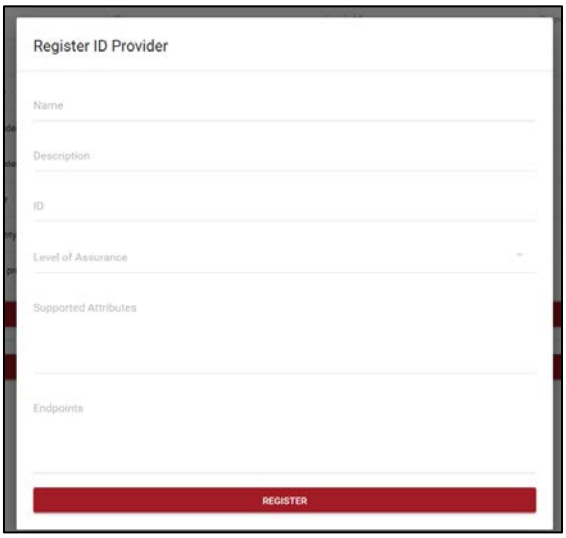

Figure 27. Administrator screen for Identity Provider registration

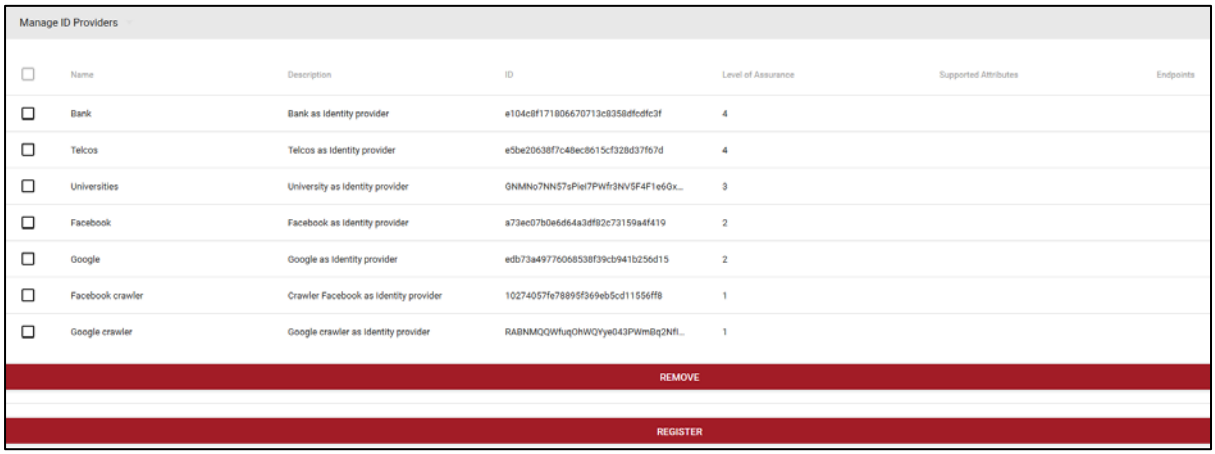

Figure 28. Management of registered Identity Providers - IDC Administrator

2. User is able to register the account that he has at the Identity Provider (shown in figure 29). The user registration will be done using the OpenID connect protocol.

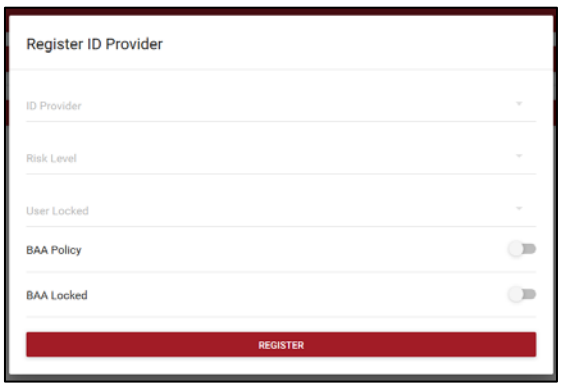

Figure 29. Identity Provider registration by the user

3. User selects from a list of the available ID providers and also provides the following information to be stored in IDC repository:

- *Risk Level:* User sets a risk level for this account in order to be able to set account locking policies. Possible values are low, medium, high.
- *User Locked:* Using this field the user can explicitly lock or unlock the specific account. The available options are 'POL' user defines that this account will be locked if there is a related policy about it, 'LOCK': user set this account locked regardless of any related policy, 'UNLOCK': user set this account unlocked regardless of any related policy.
- *BAA Policy:* User can define whether this account can be locked if a BAA send an alert about unusual user behavior.
- *BAA locked:* This field will be used to indicate that a BAA has raised an alert for unusual behavior and the specific account is locked. This will be possible if the BAA Policy field is set to "true" (user wants this account to be locked on BAA alerts).

The user is also able to see a list with his IDPs and select them to update information or remove his accounts at the IDPs (Figure 30).

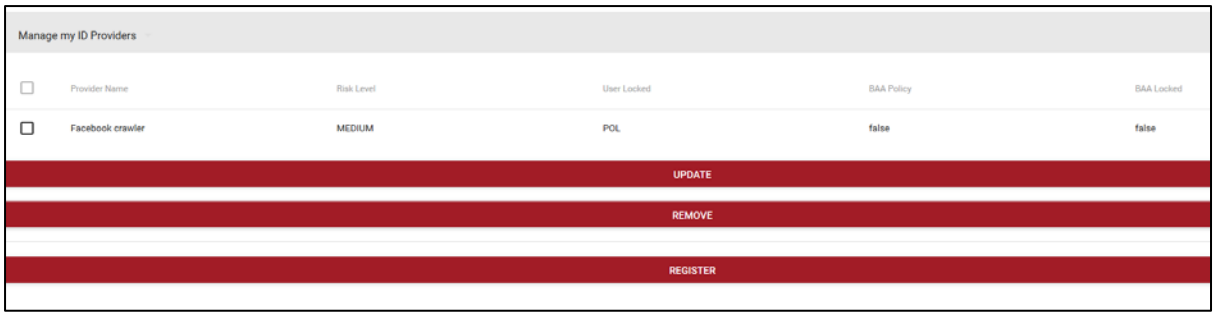

Figure 30. User screen for the management of his registered Identity Providers

4. User is presented with the list of the supported IDPs and chooses the ones that he has an account. Then, a window pops up (from the IDP - OpenID Connect) and user is requested his user name and password. After the authentication, all user attributes maintained by this Identity Provider are transferred to the Identity Consolidator if he wishes to.

Related API calls:

***Get all ID Providers**

@Path("/api/v1/id_provider")
@GET

***Create an ID Provider**

@Path("/api/v1/id_provider")
@POST
@Consumes(JSON: {ProviderName, Description, ProviderID, LevelAssurance, SupportedAttributes, Endpoints})

***Delete an ID Provider**

```
@Path("/api/v1/id_provider/{id}")
@DELETE
```

***Get all ID Providers for a user account**

```
@Path("/api/v1/user_account/{id}/id_provider")
@GET
```

***Register an ID Provider to a user account**

```
@Path("/api/v1/user_account/{user_account_id}/id_provider/{id_provider_id}")
@POST
@Consumes(JSON: {RiskLevel, UserLocked, BaaPolicy, BaaLocked})
```

***Update an ID Provider of a user account based on the ID of their join**

```
@Path("/api/v1/user_account/id_provider/{id}")
@PUT
@Consumes(JSON: {RiskLevel, UserLocked, BaaPolicy, BaaLocked})
```

***Delete an ID Provider from a user account based on the ID of their join**

```
@Path("/api/v1/user_account/id_provider/{id}")
@DELETE
```

5.3 Service Provider registration

A user can register a Service Provider (SP) to the IDC associating it with his/hers account in order for this Service Provider to be able to use the ReCRED functionality (obtaining information about specific user attributes or ask for a second factor authentication for the user using the registered BAAs etc).

A user is able to define the following fields for the Service Provider registration (figure 31) and also he can also manage their Identity providers (figure 32):

- *Service Provider URL:* the user provides the basis URL of the service provider
- *Username:* the username of the user account to the Service Provider
- *Description:* A description that user gives for the service provider in order to have related information about it, be able to search for it and distinguish its services.
- *Risk Level:* User sets a risk level for this account in order to be able to set account locking policies. Possible values are low, medium, high.
- *User Locked:* Using this field the user can explicitly lock or unlock the specific account. The available options are 'POL' user defines that this account will be locked if there is a related policy about it, 'LOCK': user set this account locked regardless of any related policy, 'UNLOCK': user set this account unlocked regardless of any related policy.

- *BAA Policy:* User can define whether this account can be locked if a BAA send an alert about unusual user behavior.
- *BAA locked:* This field will be used to indicate that a BAA has raised an alert for unusual behavior and the specific account is locked. This will be possible if the BAA Policy field is set to "true" (user wants this account to be locked on BAA alerts).

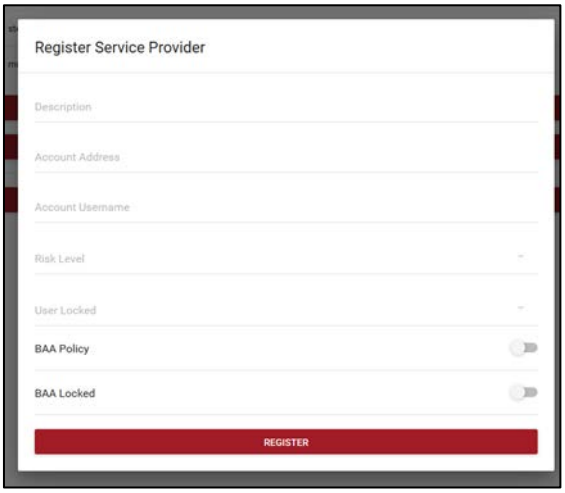

Figure 31. Service Provider registration screen

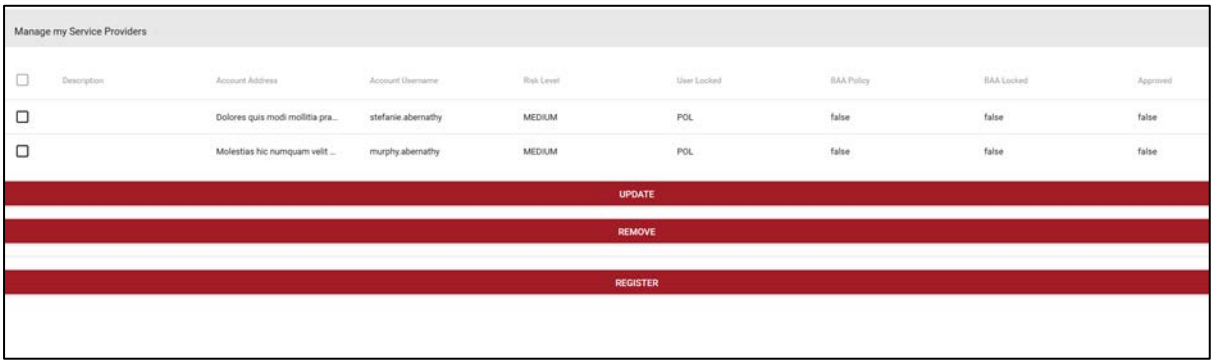

Figure 32. Registered Service Providers management screen

The new user account on the SP cannot be used until the IDC approves it. The IDC admin will check if the SP exists and might need to contact the SP in order to get related information (if it supports OpenID Connect, if it can implement the ReCRED software stack etc.). IDC admin will then approve the user account on the Service Provider and they will be able to interact through the ReCRED platform.

Related API calls:

***Get all Service Providers for a user account**

@Path("/api/v1/user_account/{id}/service_provider")
@GET

***Register a Service Provider to a user account**

@Path(
"/api/v1/user_account/{id}/service_provider")
@POST@Consumes(JSON: {Description, AccountAddress, AccountUsername, RiskLevel, UserLocked, BaaPolicy, BaaLocked})

***Update a Service Provider of a user account based on the ID of their join**

@Path("/api/v1/user_account/service_provider/{id}")
@PUT
@Consumes(JSON: {Description, RiskLevel, UserLocked, BaaPolicy, BaaLocked})

***Delete a Service Provider from a user account based on the ID of their join**
@Path("/api/v1/user_account/service_provider/{id}")
@DELETE

5.4 Account Locking mechanism

5.4.1 Account locking and status

The ReCRED Account Management Module enhances the security of online account with a unique form of second-factor authentication. In particular, online accounts carry a global status label that defines whether the service provider accepts logging attempts for that particular account. That is, upon a login attempt for a given account, the service provider queries the Account Management Module for the status of that particular account. If the status is "locked", the service provider does not grant login to the account even if presented with the correct credentials. If the status is "unlocked", the service provider grants login to the account if presented with the correct credentials. The Account Management Module within the ID Consolidator is responsible to manage the status of online accounts and to expose this information to authorized parties within the ReCRED framework.

Changing the status of an account can be either triggered by the legitimate account owner or by the ID consolidator. Alternatively, a user can define arbitrary policies to automatically lock or unlock an account.

Compared to deliverable D4.1 the design of the account management module has been refactored to take advantage of the functionalities developed in the Identity Consolidator, resulting in a learner software component. In particular, the current version of the account management module leverages the user-to-account binding of the identity consolidator as well as the storage API of the identity consolidator to store the required information to handle account locking/unlocking functionalities.

The AMM is implemented in Python as a Flask application. It implements its locking/unlocking functionality as requests to the Storage API, which holds the status of each account. The communication with the Storage API is managed by a Storage API Manager (a python class). As shown in figure 33, the main API calls lock & unlock are implemented by a public method from the Storage API Manager class, keeping AMM logic easy. Figure 34 shows how Storage API deals with each implementation, isolating the real details of the communication with the Storage API module.

```
@app.route('/lock/<string:account_id>', methods=['POST'])
def lock_account(account_id):
    storage_api_manager.lock_service_provider(account_id)
    return flask.jsonify()
@app.route('/unlock/<string:account_id>', methods=['POST'])
def lock_account(account_id):
    storage_api_manager.unlock_service_provider(account_id)
```

Figure 33. Lock/unlock API call implementation

```
    def glock_service_provider(self, provider_id, lock):
        """
            Locks/Unlocks the Service Provider with the specified provider_id (i.e., the field PrimaryKey of the Service_Provider table.
        """
        data = {"Locked": lock}
        req = requests.put(self._get_url(self, self.path_service_providers, provider_id), data=json.dumps(data),headers=self.headers)
        return req.status_code == 204
    def lock_service_provider(self, provider_id):
        """
            Locks the Service Provider with the specified provider_id (i.e., the field PrimaryKey of the
            Service_Provider table.
        """
        return self.glock_service_provider(self, provider_id, True)
    def unlock_service_provider(self, provider_id):
        """
            Unlocks the Service Provider with the specified provider_id (i.e., the field PrimaryKey of the
            Service_Provider table.
```

Figure 34. Lock/Unlock API call implementation within Storage API Manager

Related REST API calls:

Operation: GET /list

Description: Returns a list of all accounts.

Request:

```
GET /list
Accept: application/json
```

Response:

```
200
Content-Type: application/json
{
 "data": [
                {
                        "account_address": "www.facebook.com",
                        "account_id": "9Mk",
                        "account_username": "recred",
                        "status": "unlocked"
                }
 ]
}
```

Operation: GET /pair

Description: Gets a valid token to create a pairing with a third party account.

Request:

```
GET /list
Accept: application/json
```

Response:

```
200
Content-Type: application/json
```

```
{
 "data": {
                                "token": "6MTwvnkk"
 }
}
```

Operation: POST /pair/{token}

Description: Creates a new pairing with a third party account.

Request:

```
POST /pair/6MTwvnkk
Accept: application/json
Content-Type: application/json
{
        "account_address": "www.facebook.com",
        "account_username": "recred"
}
```

Response:

```
200
Content-Type: application/json
{
 "data": {
                                "accountID": "9Mk"
 }
}
```

Operation: GET /status/{account_id}

Description: Gets the current status of the specified user account (account_id).

Request:

```
GET /status/9Mk

Accept: application/json
```

Response:

```
200

Content-Type: application/json

{

 "data": {

                    "status": "unlocked"

 }

}
```

Operation: POST /lock/{account_id}

Description: Locks the pairing associated with the user account (account_id).

Request:

```
POST /lock/9Mk

Accept: application/json
```

Response:

```
200

Content-Type: application/json

{}
```

Operation: GET /unlock/{account_id}

Description: Unlocks the pairing related to the user account (account_id).

Request:

```
GET /list

Accept: application/json
```

Response:

```
200

Content-Type: application/json
```

```
{}
```

Operation: DELETE /unpair/{account_id}

Description: Removes the pairing related to the user account (account_id).

Request:

```
DELETE /pair/9Mk

Accept: application/json

{

        "account_address": "www.facebook.com",

        "account_username": "recred"

}
```

Response:

```
200

Content-Type: application/json

{}
```

5.4.2 Account Locking policies

Users are able to define policies for locking their accounts (on IDPs and SPs) from using the ReCRED platform (Figure 33). A user can define policies based on the account risk level and the day/time of the week. Specific account locking policies can also be defined.

More specifically, a user can define the following type of policies:

- Lock all account(s) of a specific risk level (low, medium or high) at specific days and times of the week.
- Lock specific IDP account (regardless the risk level associated with it) at specific days and times of the week (Figure 34).
- Lock specific SP account (regardless the risk level associated with it) at specific days and times of the week (Figure 35).
- When a user is requesting access to an SP or an IDP account, the IDC will check if the account is locked with the following order:
- Check if the specific account is locked due to a BAA alert.
- Check if the user has locked this account or if it unlocked by the user.
- Check if the specific account has a policy associated with it and if yes, check if the current time/date matches the policy.
- Check there is a policy associated with the risk level of this account.

After checking the above cases, the IDC uses the locking functionality that is described as “Latch”.

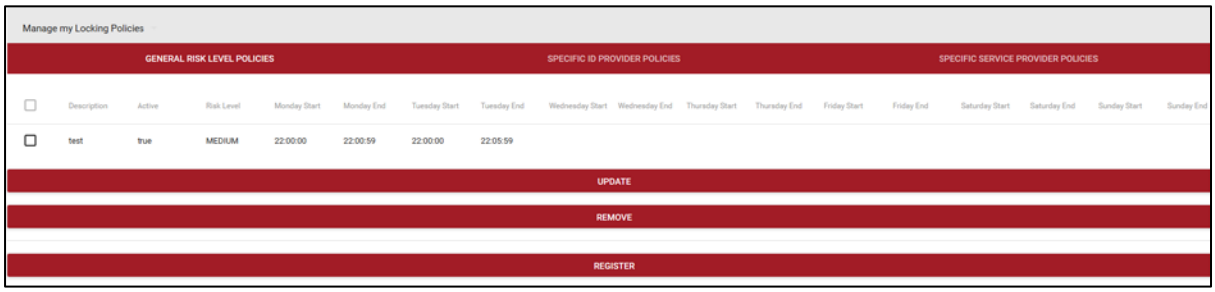

Figure 35. Account locking policies management screen

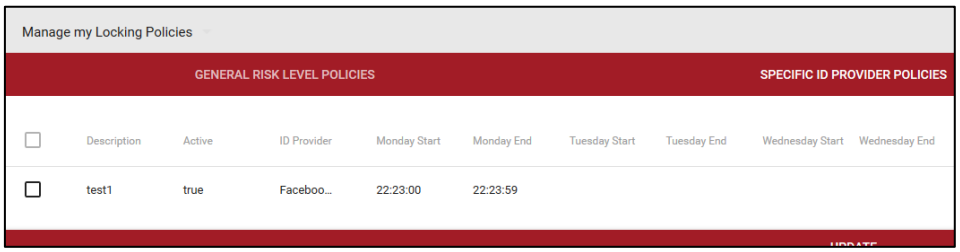

Figure 36. Account locking policies for Identity providers

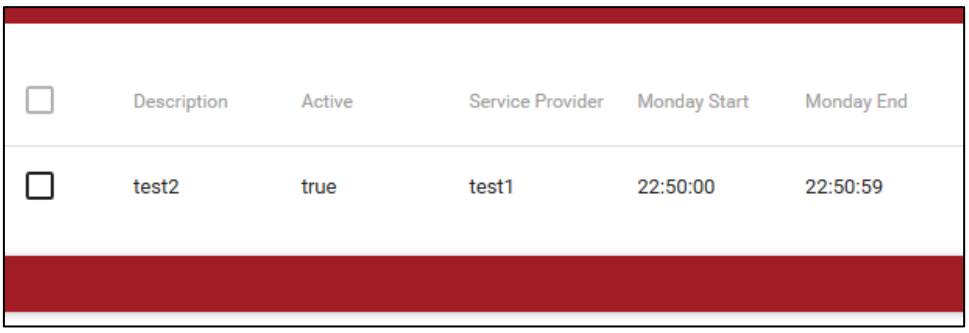

Figure 37. Account locking policies for Service Providers

Related API calls:

***Get all Risk Level Policies for a user account**

@Path("/api/v1/user_account/{id}/policy/risk_level")
@GET

***Get all ID Provider Policies for a user account**

@Path("/api/v1/user_account/{id}/id_provider/policy")
@GET

***Get all Service Provider Policies for a user account**

@Path("/api/v1/user_account/{id}/service_provider/policy")
@GET

***Register a Risk Level Policy to a user account**

@Path("/api/v1/user_account/{id}/policy/risk_level")
@POST
@Consumes(JSON: {Description, Active, RiskLevel, MondayStart, MondayEnd, TuesdayStart, TuesdayEnd, WednesdayStart, WednesdayEnd, ThursdayStart, ThursdayEnd, FridayStart, FridayEnd, SaturdayStart, SaturdayEnd, SundayStart, SundayEnd})

***Register an ID Provider Policy to a user account**

@Path("/api/v1/user_account/{user_account_id}/id_provider/{id_provider_id}/policy")
@POST
@Consumes(JSON: {Description, Active, MondayStart, MondayEnd, TuesdayStart, TuesdayEnd, WednesdayStart, WednesdayEnd, ThursdayStart, ThursdayEnd, FridayStart, FridayEnd, SaturdayStart, SaturdayEnd, SundayStart, SundayEnd})

***Register a Service Provider Policy to a user account**

@Path("/api/v1/user_account/{user_account_id}/service_provider/{service_provider_id}/policy")
@POST
@Consumes(JSON: {Description, Active, MondayStart, MondayEnd, TuesdayStart, TuesdayEnd, WednesdayStart, WednesdayEnd, ThursdayStart, ThursdayEnd, FridayStart, FridayEnd, SaturdayStart, SaturdayEnd, SundayStart, SundayEnd})

***Update a Locking Policy of a user account based on the ID of their join**

@Path("/api/v1/user_account/locking_policy/{id}")
@PUT
@Consumes(JSON: {Description, Active, MondayStart, MondayEnd, TuesdayStart, TuesdayEnd, WednesdayStart, WednesdayEnd, ThursdayStart, ThursdayEnd, FridayStart, FridayEnd, SaturdayStart, SaturdayEnd, SundayStart, SundayEnd})

***Delete a Locking Policy from a user account based on the ID of their join**

@Path("/api/v1/user_account/locking_policy/{id}")
@DELETE

5.5 Account Recovery mechanism

The user is presented with a list of the accounts that he has and from there he is able to recover the secret keys (e.g., private keys) to a new device (e.g., after a device failure).

If the user loses his device, it is the account management module that is responsible for guiding the user through the recovery process. To this end it needs to interface with the credential management module.

When the user loses his device, he logs in to the IDC, to the account management module (AMM) and requests recovery on a new device. The AMM lists to him all his accounts and guides him (Like with a Next Menu) through the process of recovering his credentials on all his connected accounts (ID provider's).

If the Credential Management Module has backed up credentials for an account, then the AMM uses that to recover access to that account. If there are no backed-up credentials, the AMM takes the user to the ID provider's account recovery site/API, where he may be prompted for security questions/passwords, SMS, etc. by that ID provider.

Note that if the user does not have access to the consolidator due to losing all his authenticated devices, he first needs to use the Authentication Management Module to prove to the IDC that he is the legitimate user.

5.6 Mobile Connect

5.6.1 APIGEE OneAPI Exchange Discovery Service

During the user journey, there are two APIs that would be contacted in the process, one of them is the Mobile Connect API, which is used for the Authentication, Authorization and Identity services that is to be implemented using Mobile Connect. The other API is the Discovery API, this is responsible for determining or discovering the user's Mobile Network Operator (MNO) and then providing the endpoints details as well as credentials used to make the Mobile Connect calls.

Every mobile operator has a Mobile Country Code (MCC) and Mobile Network Code (MNC) to identify themselves. The Discovery service identifies the MCC_MNC and returns the details as well as the MNO's Mobile Connect API details to the Application/Service Provider (SP), which then enables the application/Service Provider to call the relevant Mobile Connect service.

The Discovery service can determine the MCC_MNC in a number of ways, such as making calls in the background without user input and in other situations the Discovery service requires the mobile number from the user, in which case the use enters their MSISDN in the application/Service Provider which then makes the REST calls to the OneAPI exchange, which determines the MNO and sends it back to the SP. The SP then makes the requisite calls to the Mobile Connect API of the MNO to have the user authentication, authorized (introduced in Mobile Connect v1.2).

5.6.2 Identity Consolidator as a Discovery Service

The process or user journey is very similar to the above operation of the OneAPI Exchange Discovery Service. However in this scenario, rather than the Service Provider(s) or application making a direct call to the OneAPI gateway, the calls are made to the ID Consolidator. The SP might not be Mobile-Connect capable and only trusts the ID Consolidator as the only endpoint to contact for determining the Mobile Network Operator (MNO) of the user's phone number, MSISDN, etc..

The Identity Consolidator acts as an OpenID Connect provider that stores attributes Mobile Connect sessions. The Service Provider is oblivious to the Mobile Connect infrastructure, but it knows that there are Level of Assurance (LoA) attributes that can be retrieved from the MNOs such as the Phone number, if the last bill was paid, etc.

The Identity Consolidator invokes the GSMA apigee API Discovery Service (on-behalf of the Service Provider(s)) on a trusted Mobile Connect provider. The API Exchange uses federation to ensure that

the SPs (in this case the IDC) only need to connect to one MNO that is in a federated Mobile Connect infrastructure and is able to access customers of all connected/federated MNOs.

5.7 Degree of Privacy Selection and Enforcement

5.7.1 Least Degree of Privacy

If a user wishes to maintain the least degree of privacy against the ID consolidator, he can choose to fully trust the ID consolidator. In this case the Identity Consolidator holds the user’s identity attributes in Identity Repository and acts as an Identity Provider that provides identity attributes to Service Providers by using the OpenID Connect specification.

A user has to authenticate with the highest LoA they have previously achieved with the Identity Consolidator in order to change their preferences regarding this functionality.

5.7.2 Highest Degree of Privacy

With highest degree of privacy the Identity Consolidator holds a reference to where the user’s identity attributes can be retrieved. When a Service Provider requests an identity attribute, the Identity Consolidator responds with a URI that the Service Provider can then use to resolve the value of the requested attributes.

When the user has previously selected the least degree of privacy option, then the IDC deletes all the stored identity attributes from its Identity Repository.

Additionally, the user should authenticate with the highest LoA they have previously achieved with the IDC in order to change their preferences regarding this functionality.

The Identity Federation is used when a Service Provider asks an Identity provider for a proof of identity, and the Identity Provider does not know that identity. In this case, the Identity provider redirects the user to the IDC for further actions. Subsequently, the IDC either responds with the identity attributes (if he has the attributes and the user trusts the consolidator) or redirects the Service Provider to the appropriate Identity Provider that maintains the requested attributes. The Identity Consolidator acts as an Identity provider discovery service, for the Identity Federation.

The Identity Federation will be the only way that the IDC interacts with the user and the IDPs in case of highest degree of privacy. However, the Identity Federation will also be used in less than highest degree of privacy modes, as it will not always be the case that the user has transferred all his attributes from the IDPs to the IDC.

6 Identity Management Module

This section provides the description and the implementation details of the Identity Management module. The Identity Management module is divided in the Identity Profile Management and the Consent Management. The same as all the other modules of the Identity Consolidator, this module can be access through the main page of the Identity Consolidator platform.

6.1 Identity Profile Management

The Identity Profile Management module is responsible for providing functionality that allows a ReCRED user to manage his identity attributes.

The Identity Profile Management module comprises the following:

- A backend that provides a REST API including methods covering all the management functionality. The backend communicates with the Identity Repository over the Storage API. Users are authenticated using the ReCRED Authentication module
- A web frontend that offers a UI for ReCRED users that wish to manage their Identity Profile over a web connection
- A mobile application that offers ReCRED uses the opportunity to manage their Identity Profile using their mobile device

6.1.1 Identity Profile Management Web Front-end

The Identity Management web application allows users to view and manage their identity data alongside with attributes maintained by each Identity Provider. It also exposes a REST API to be used by other applications such as the Identity Profile Management mobile application.

The functionality offered is:

- The user can view all current data that are stored in the Identity Repository.
- The user is also able to centrally update the value of some identity attributes, as long as this is allowed by the ID provider-defined policy for these attributes.
- Transfer identity attributes amongst Identity providers that enable such a procedure. Currently Google+ and Facebook don’t allow this operation however.
- Modify an attribute and nullify it if possible.
- Allow users to check which of their attributes are present in which Identity Providers.
- View de-anonymization risks with regards to identity attributes which are unknown to the Identity Providers

6.1.1.1 Main menu

The main menu of the Identity Profile Management module allows an authenticated user to have access to almost all the identity information that is relevant to him regarding Identity providers’ data. The mapping is based on the Storage API and allows the user to view and/or edit the following information:

- Identity Providers acquired information (ID Provider Fields)
- Identity Providers acquired information risks data (ID Provider fields risk)
- Financial Information (Financial Info)
- Financial Information risks (Financial Info risks)
- User declared names (User name)
- User captured location information (User Location)
- Url data
- Uploaded Picture information (Media Item)
- Physical Identity documents information (Physical ID)
- User Phone numbers
- User Email Addresses
- User Addresses information

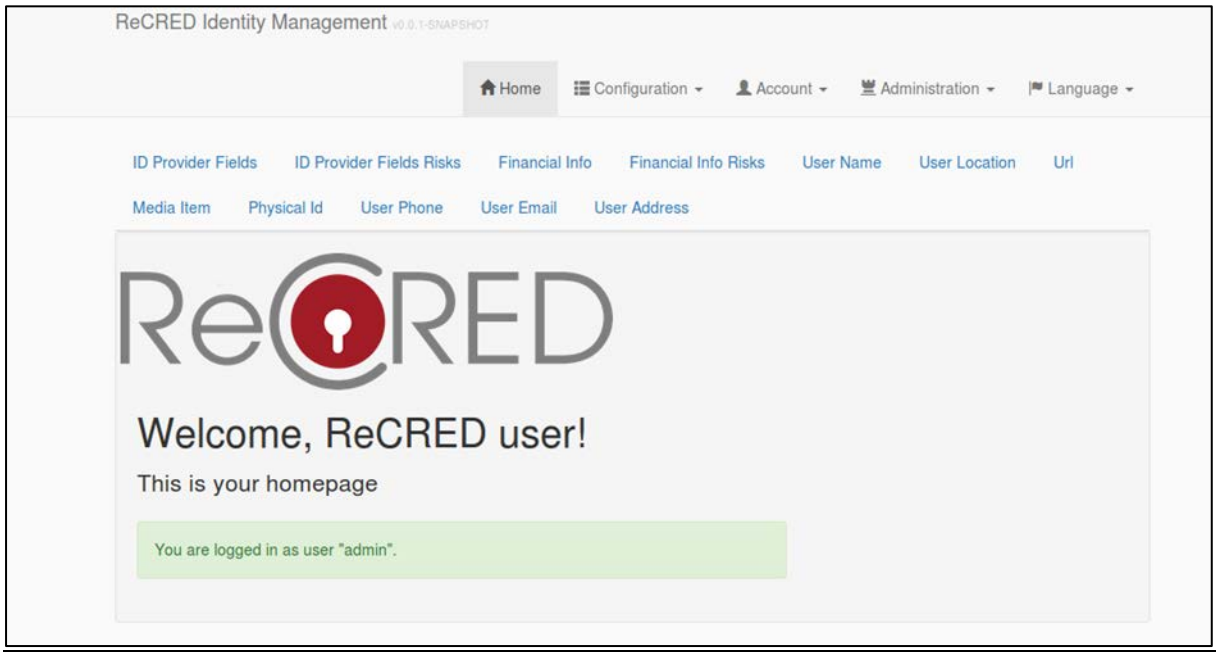

Figure 38. Identity Profile Management web application main page

In general, the user has the ability to do the same for most of the identity attributes categories above. For the identity attributes that are available and maintained only by the Identity Providers and we just store the reference to those identity providers are presented with a gray square surrounding them while all the other identity attributes without a gray square are maintained by the Identity Consolidator.

6.1.1.2 Users' acquired identity attributes from Identity Providers

In the Id Provider fields menu the user can view his identity attributes defined in all the Identity Providers. The Identity Management web application retrieves all the required data from the Identity Repository through the Storage API. Currently the administrator of the Identity Profile Management module has access and is presented with all the available identity attributes for all users. The create button serves only for testing purposes.

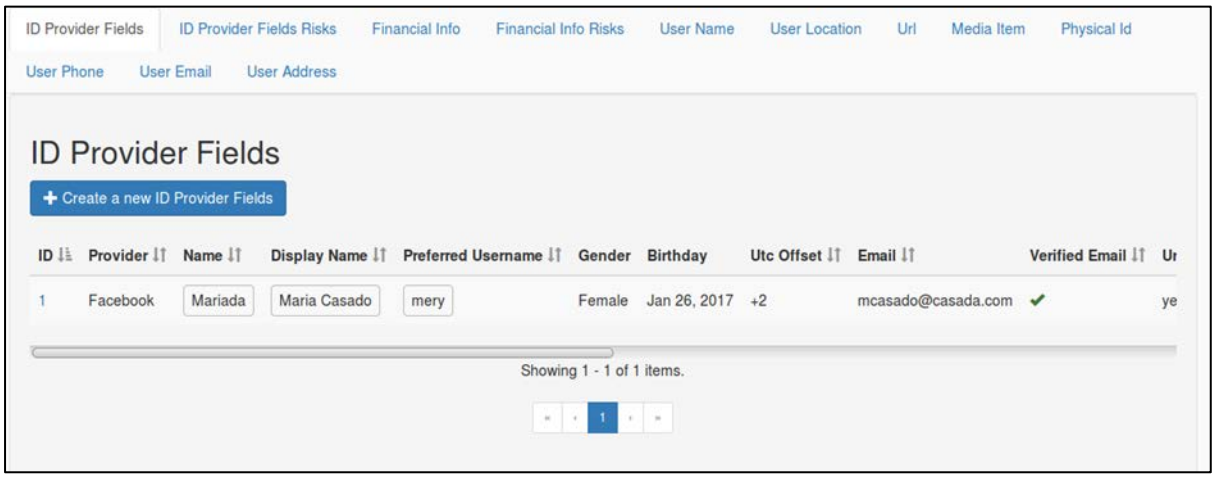

Figure 39. Identity Profile Management web application. Identity Providers users' identity attributes

By pressing the edit button a user can edit any of the available attributes as shown in figure 40 below.

Figure 40. Edit the value of a user’s identity attributes acquired from his Identity Providers

When a user chooses to edit the identity attributes for one of his Identity Providers a popup dialog menu opens and there the user can edit/check all the attributes as shown in figure 41 below. If for example the user changes the name, the attribute value is updated in the Storage API using the appropriate REST API calls at the backend, when the user presses the save button in the popup window.

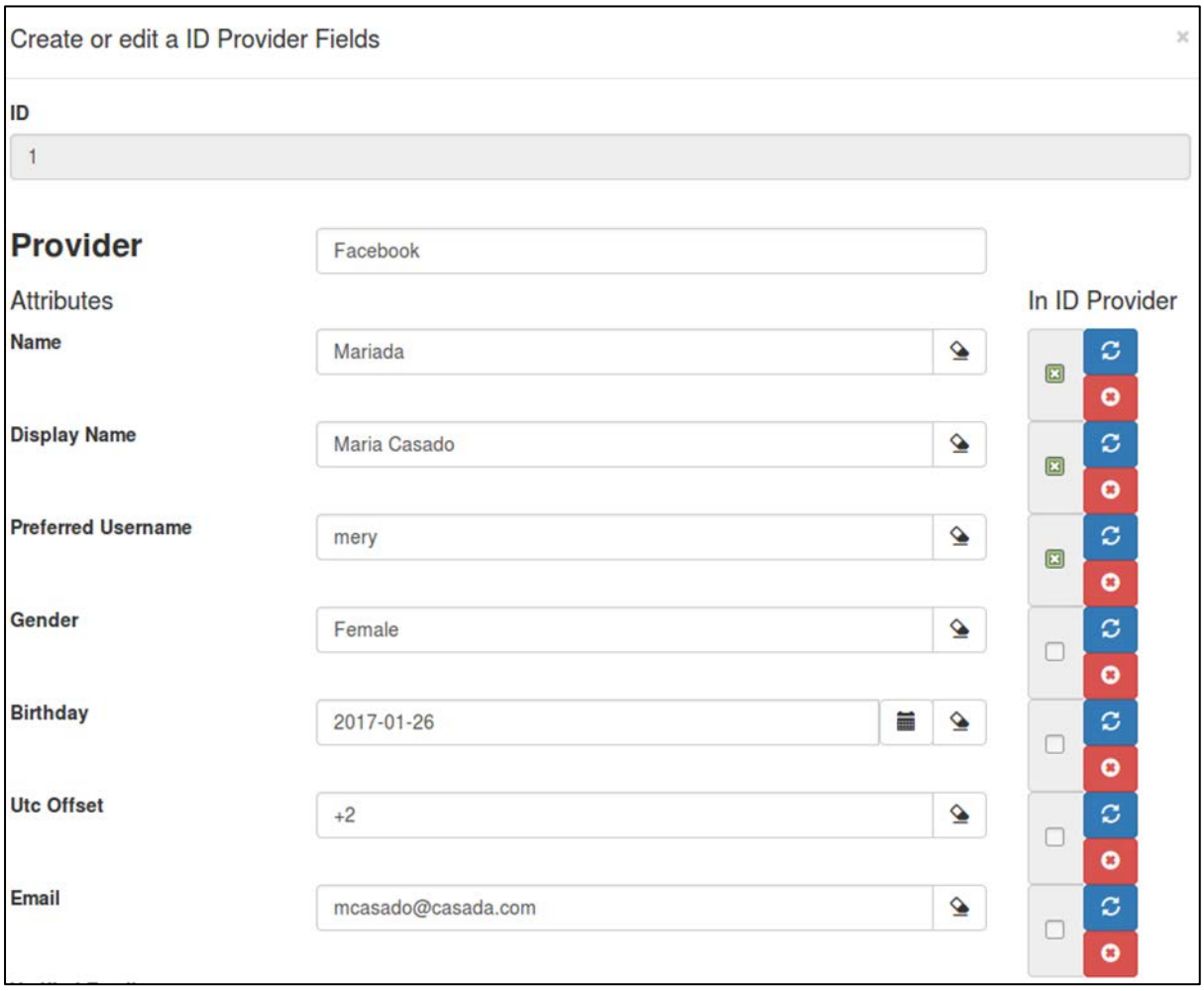

Figure 41. Edit dialog fort identity attributes acquired from an Identity Provider

The transfer button shown in figure 42 below allows the user to transfer one or more identity attribute from one Identity Provider to another. This functionality currently utilizes the Facebook API as a showcase but Facebook doesn’t allow the editing of user attributes although no error occurs.

We will implement this functionality as part of the Identity Profile management module but using and extending OpenID Connect and the functionality will be described in detail in the next deliverable (D4.3) of WP4.

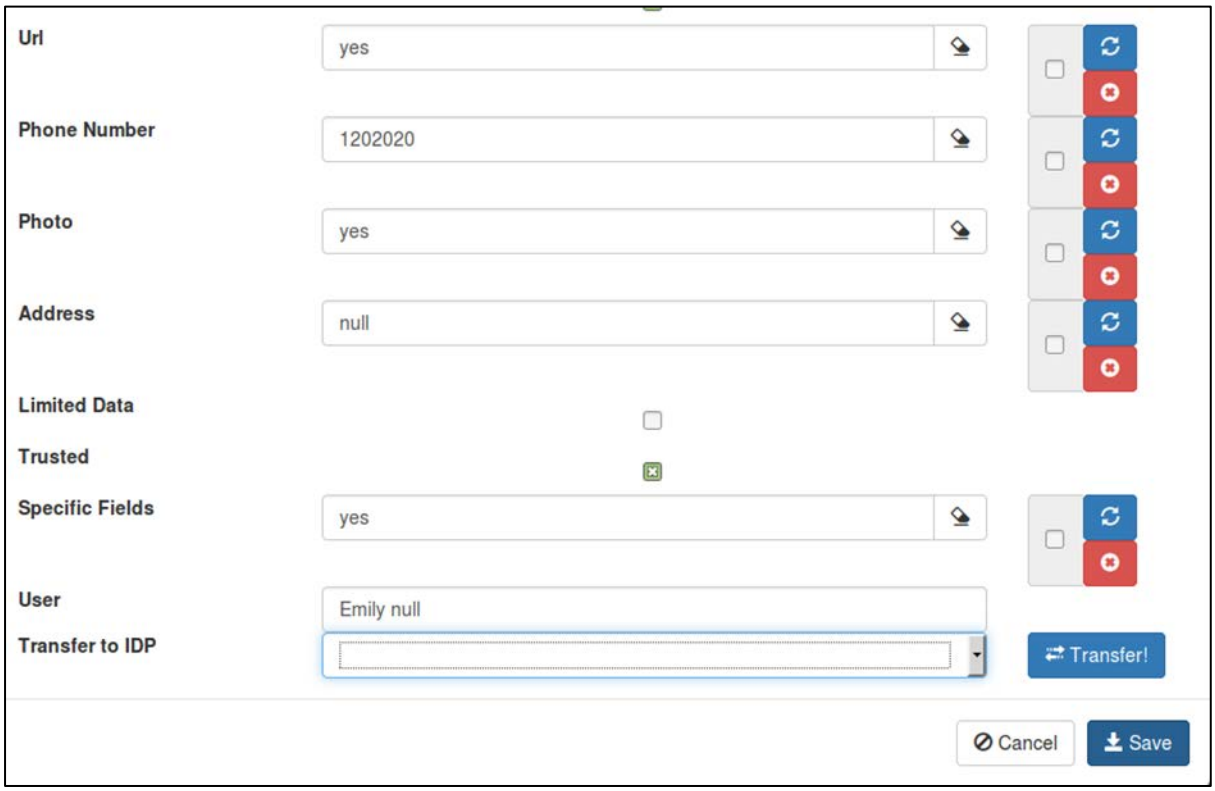

Figure 42. Identity Profile Management web application. Edit identity attribute values or transfer identity attributes among IdPs functionalities

Identity attributes transfer among different Identity Providers or identity attributes deletion update from Identity Providers: For the implementation of these functionalities we plan to extend the OpenID Connect implementation (OpenAM) on the Identity Providers so that the Identity Profile Management module supports identity attributes transfer among different IdPs and at the same time support the dynamic update and deletion of identity attributes from Identity Providers.

Based on the typical OpenID Connect flow, for the description of this functionality we assume that the user wants to transfer on of his identity attributes from IDP_A to the IDP_B. In a typical OpenID Connect scenario the OpenID Connect Provider is IDP_A that holds the identity attribute that the user wants to transfer and the Service Provider is IDP_B that will receive the transferred identity attribute. In order the identity attribute to be transferred to IDP_B (Service Provider) the user has to authenticate with IDP_A (OpenID Connect Provider) and provide his consent. When he has successfully authenticated with IDP_A and provided his consent the IDP_B receives the identity attribute and stores it.

In the case where the user wants to delete or update the value an identity attribute from an Identity Provider then he first has to authenticate against this ID Provider (OpenID Connect Provider) and the he has to provide his consent for the deletion or update of the identity attribute. In this scenario the Service Provider that request the deletion or update of the identity attribute is the Identity Consolidator (Identity Profile Management web application).

6.1.1.3 User’s Financial Information

The Financial Info page of the Identity Profile management web application presents to the user all his acquired financial information (e.g., his Income, Monthly Loan etc.). The figure below is an example of this page.

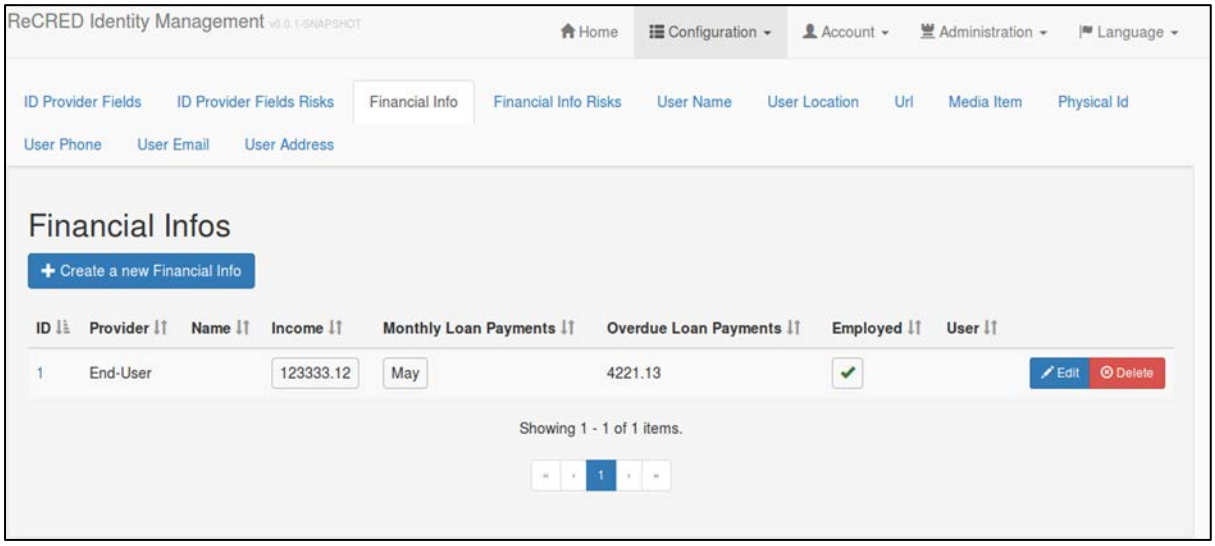

Figure 43. Identity Profile Management web application. User's financial information page

6.1.1.4 User’s declared names information

The page shown in figure 44 below presents to the user the information of the different names acquired through Online Identity Acquisition from his multiple online accounts and the information of the different names acquired through Physical Identity Acquisition from his multiple identity documents. We allow the storage of multiple names because the declared user’s name in his multiple online accounts and real-world identities may differ.

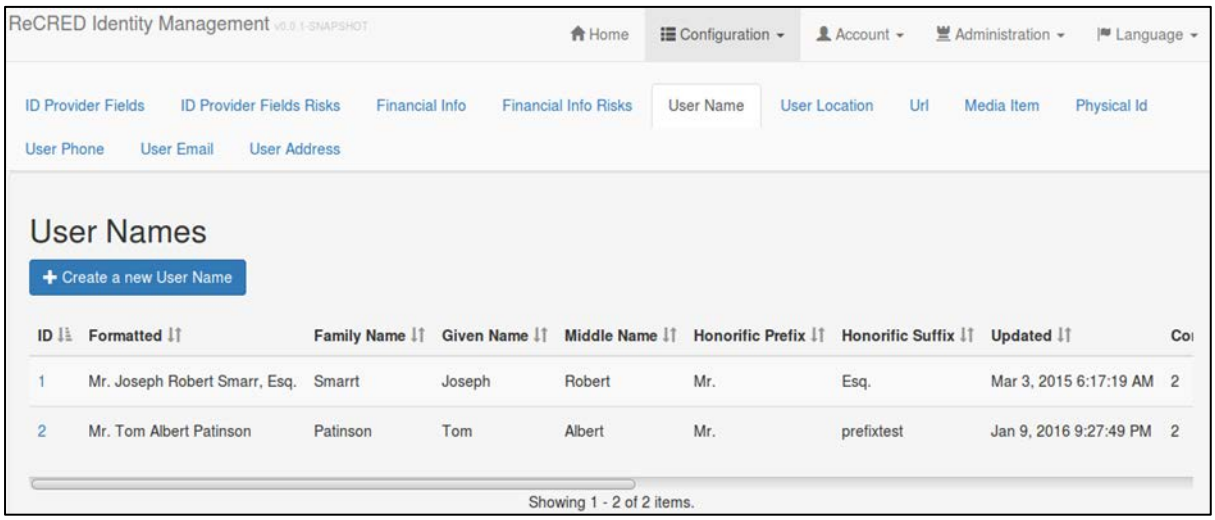

Figure 44. User's names information

Furthermore, verification information is also displayed for all the names that have been acquired through the Physical Identity Acquisition module as shown in the figure below.

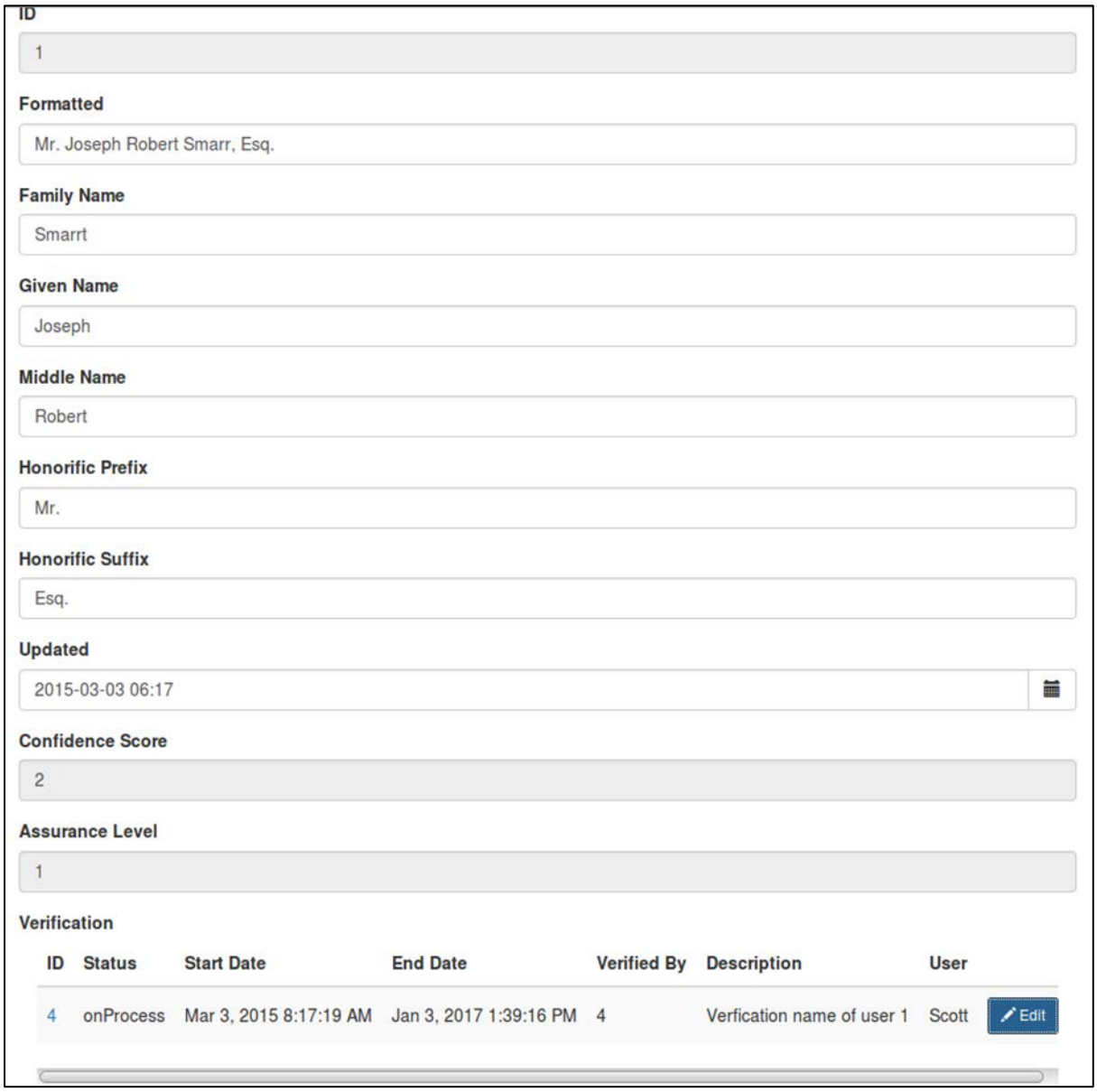

Figure 45. User's acquired names details and verification information

6.1.1.5 User’s acquired Locations information

The Identity Profile management module also offers a page that presents to the user all the locations acquired through the location tracking functionality of the Physical Identity Acquisition module for address verification. Figure 46 below is an example of this page.

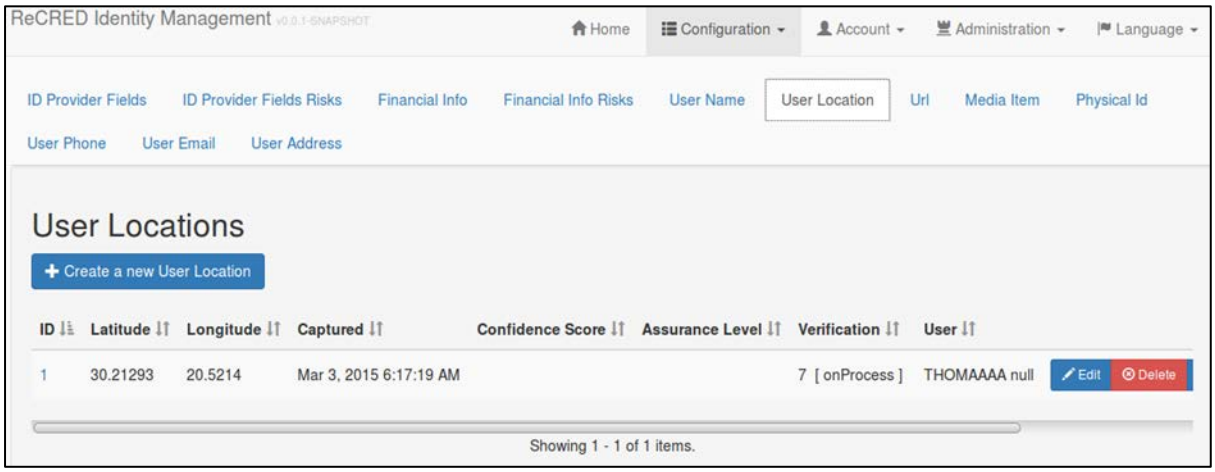

Figure 46. User's acquired locations information page

6.1.1.6 *URLs information*

The Identity Management module also offers a page the presents to the user the information of his declared websites etc.

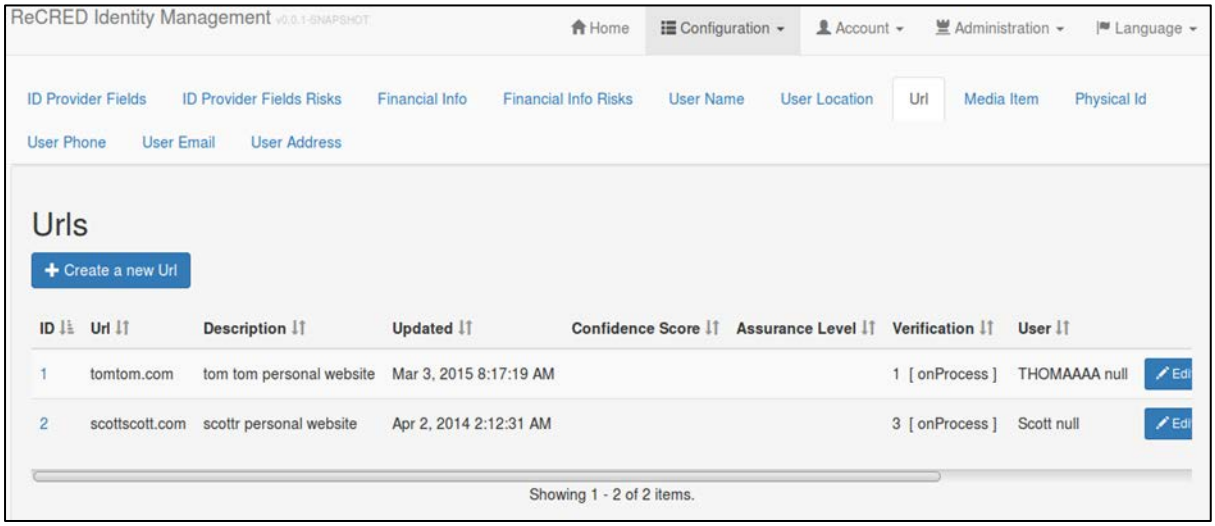

Figure 47. User's declared websites information

6.1.1.7 *Uploaded Media Items information*

This page shows to the user the information of the media items (pictures, videos, etc.) that are maintained by the Identity Consolidator. Most of the stored media items are acquired by the Physical Identity Acquisition module for the verification of the users’ physical identity documents.

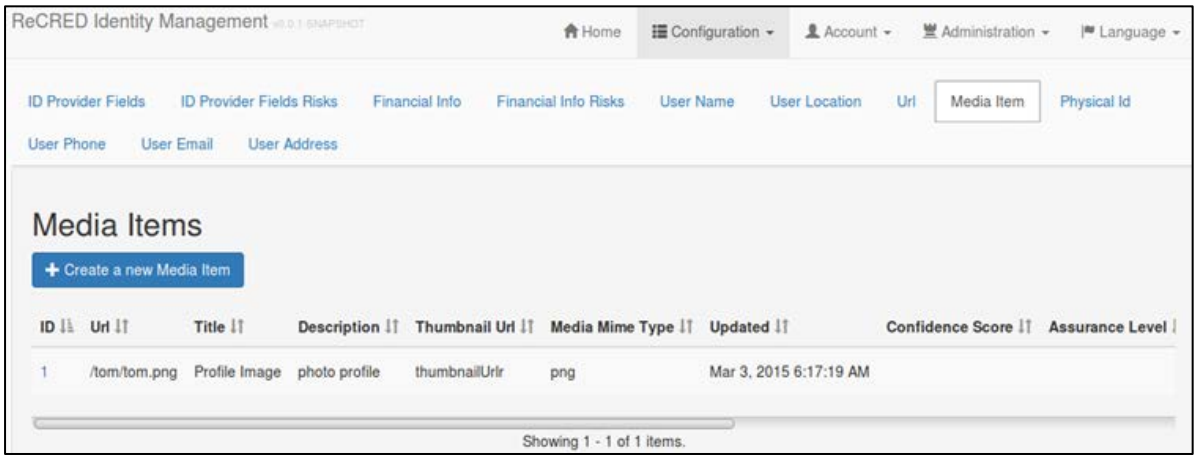

Figure 48. User's acquired media items information

6.1.1.8 *Physical Identity Documents information*

The Identity Management module allows the users to view and manage the information of the physical identity documents that he has declared and verified through the Physical Identity Acquisition module. Figure 49 below is an example of this page that offers this functionality.

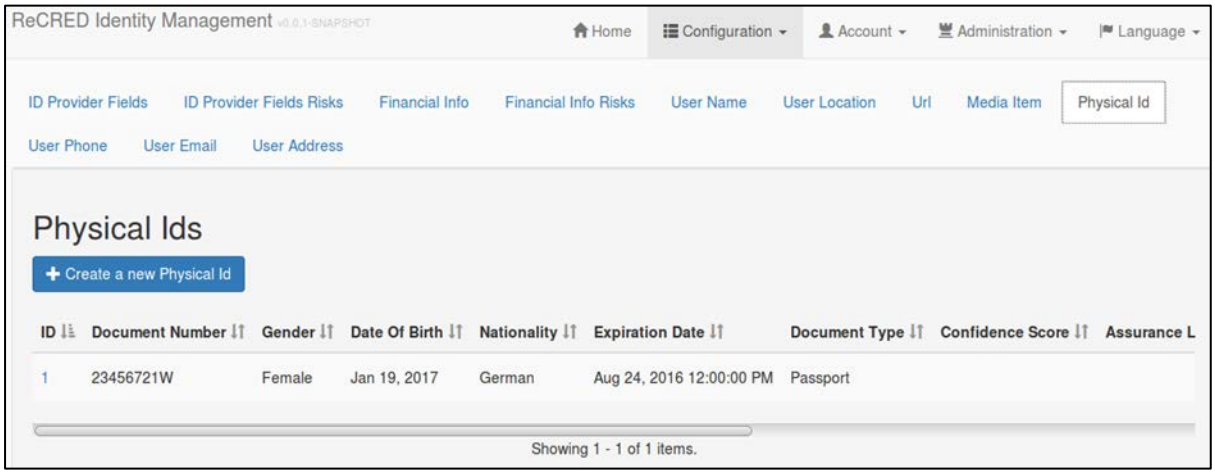

Figure 49. User's Physical Identity Documents information

6.1.1.9 *User Phones information*

Additionally, the Identity Management module offers a page that presents to the user the information of the phone numbers that the Identity Consolidator has stored about him.

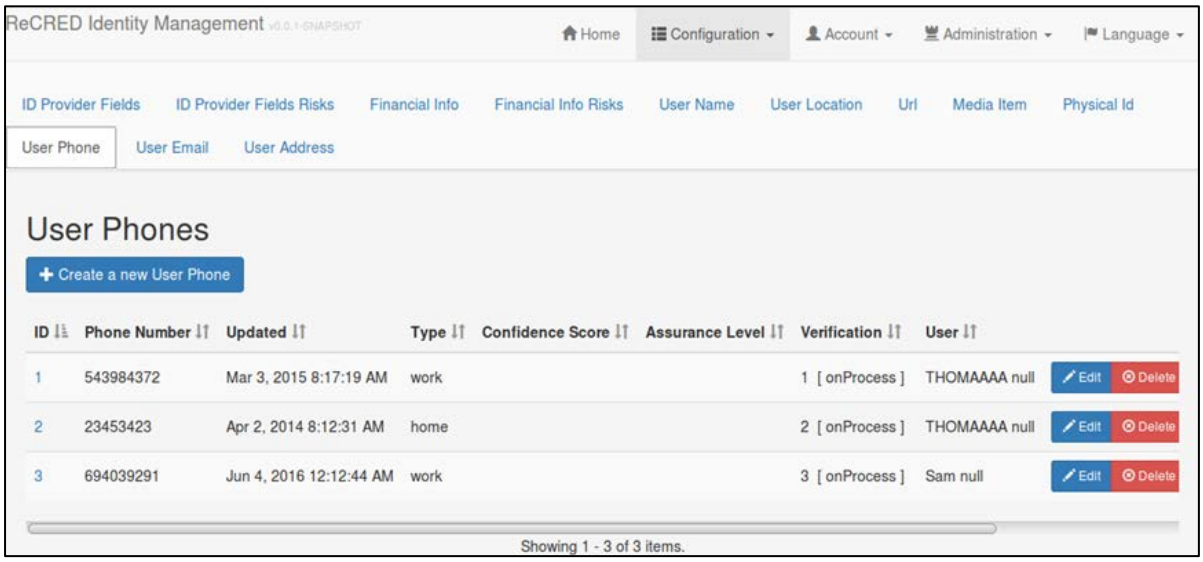

Figure 50. User's phones information

6.1.1.10 User’s Email addresses information

This page presents to the user the email addresses the Identity Consolidator acquired about him through Online Identity and Physical Identity Acquisition modules.

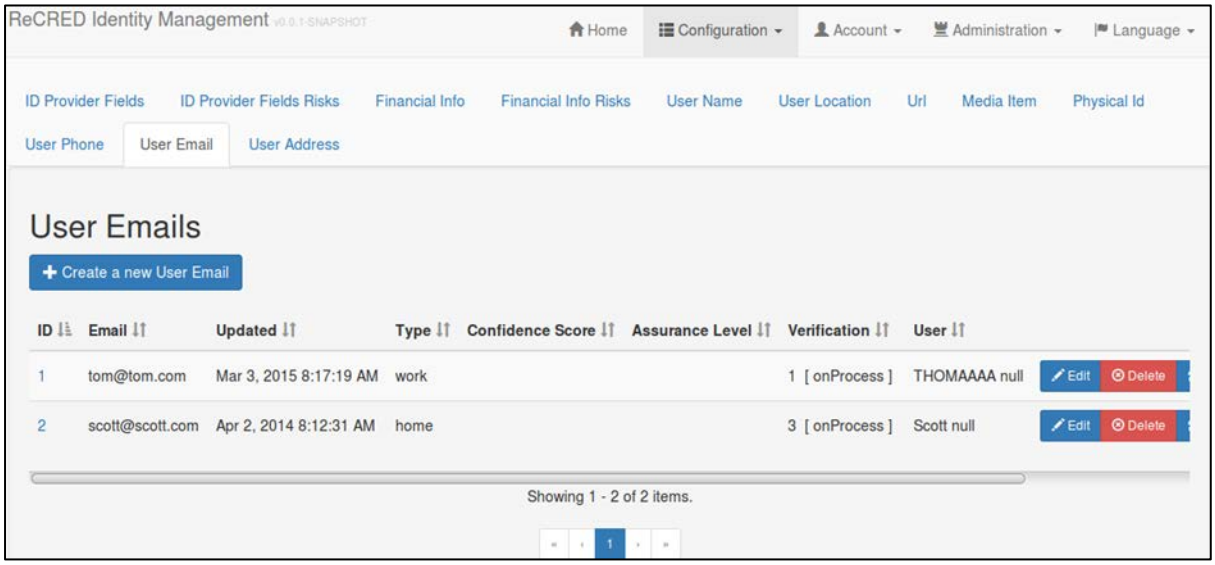

Figure 51. User's declared email addresses information

6.1.1.11 User’s Addresses information

The last functionality that the Identity Profile Management module offers to the users in terms of viewing their identity attributes is a page that presents the information of their addresses information acquired and verified through the Physical Identity Acquisition module. Additionally, this may include addresses acquired from the various online accounts of a user.

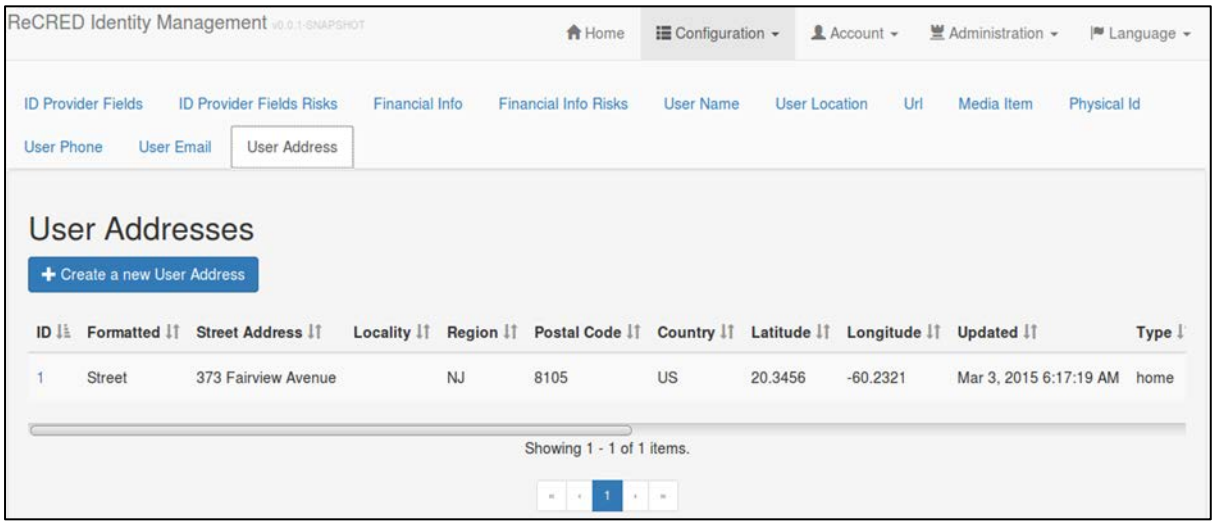

Figure 52. User's declared addresses information

6.1.2 Identity Profile Management Mobile Application

The Identity Profile Management mobile application is essentially an Android mobile application aiming to provide the same functionality as the web application through a mobile device. The application communicates with the Identity Profile Management back-end, in order to allow the user to view and manage his online accounts and physical identities. More specifically, through the Identity Profile Management mobile application, the users can have access to the following functionalities:

- View their online and physical identities, as they are acquired by the online and physical acquisition modules respectively,
- View which identity attributes are maintained by which Identity Providers
- Transfer identity attribute values among different Identity Providers or from an Identity Provider to the ID Consolidator
- Delete an identity attribute value from an Identity Provider that maintains it
- View which identity attributes have been revealed to Service Providers
- View de-anonymization risk indicators regarding unrevealed identity attributes to Identity Providers and Service Providers
- Create and manage partially verifiable profiles

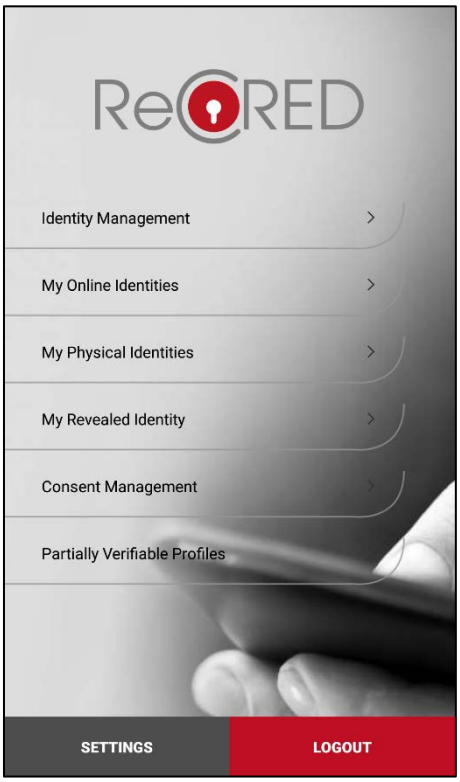

Figure 53. Identity Profile Management mobile application landing page

6.1.2.1 My Online Identities

Through the “My Online Identities” option, the user can see which of his identity attributes are maintained by which Identity Providers. These Identity Providers are connected with the user’s account through the Online Identity Acquisition Module. This functionality is offered to the user through the following two alternative approaches:

1. *View which Identity Providers know an Identity Attribute:* The user can see a list with all his identity attributes as shown in figure 55. After selecting a specific attribute, a new list is displayed, with all the Identity Providers that the user has connected, and what each Identity Provider knows or can infer regarding the selected attribute (figure 54).

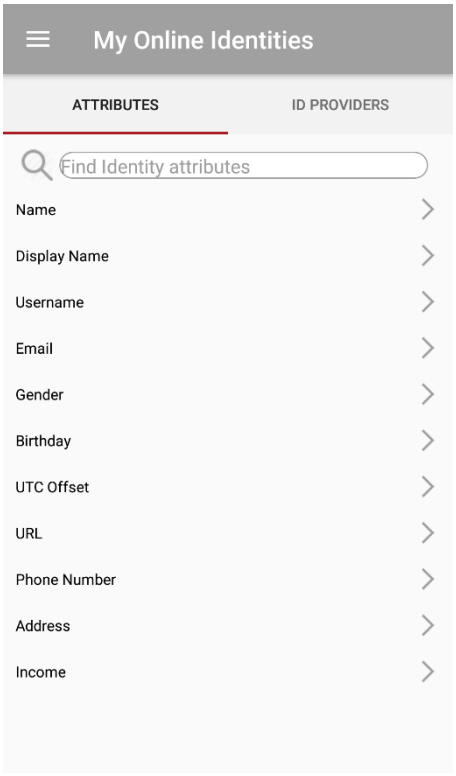

Figure 55. List of Identity attributes

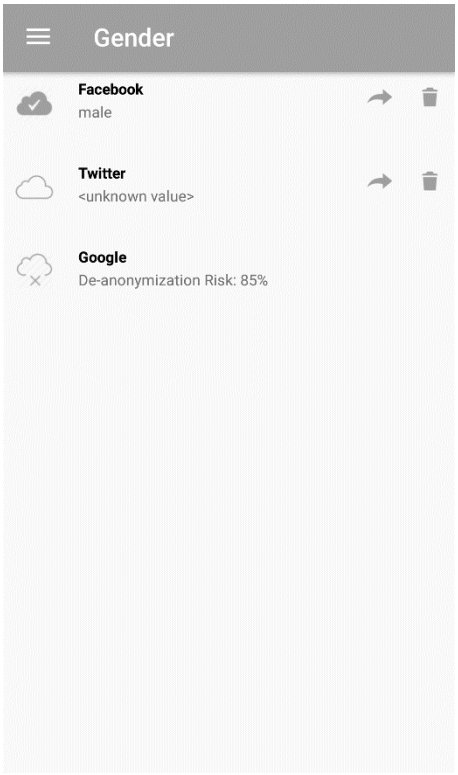

Figure 54. Identity Providers that know a selected identity attribute

2. *View which Identity attributes that are known to Identity Providers:* The user can see a list with all his Identity Providers, and for each Identity Provider the number of the attributes it maintains is also displayed (figure 57). After selecting a specific Identity Provider, a new list is displayed, with all the identity attributes and what the selected Identity Provider knows about each attribute (figure 56).

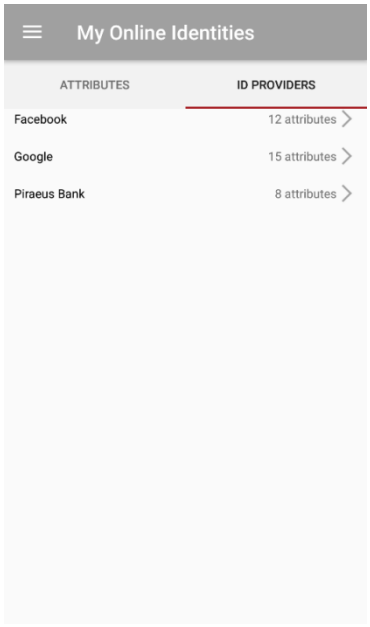

Figure 57. List of Identity Providers

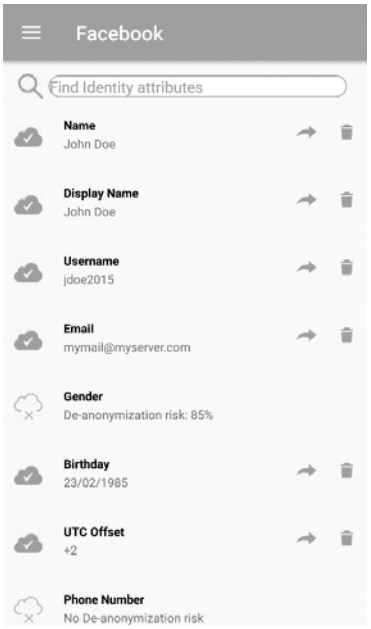

Figure 56. Identity attributes known to an identity provider

In both approaches, three different cases are identified, regarding what an Identity Provider and the Identity Consolidator know about an attribute:

- Both the Identity Consolidator and the Identity Provider know the value of a given attribute. This is depicted by an icon next to the Identity Provider, and the value of the attribute is also displayed.
- The Identity Consolidator knows that the Identity Provider maintains a given attribute, but it does not know its actual value (mainly because the user has chosen not to trust the Identity Consolidator). This is depicted by an icon next to the Identity Provider, but the actual value of the attribute is not displayed (since it is unknown).
- The Identity Provider does not maintain a given attribute, which is depicted by an icon next to the Identity Provider. In that case, the de-anonymization risk is also displayed, as long as the selected attribute is not unique (e.g. email address or telephone number).

We are currently investigating an extension to the OpenID Connect protocol, so that a user can authorize a service provider to request from an identity provider to update and/or delete his attributes. In that case, the user will be able to also transfer attributes from one Identity Provider to other Identity Providers, overriding any existing values maintained by the target Identity Providers, while respecting, at the same time, the user’s policies defined through the Consent Management module. The user will also be able to delete an attribute value from an Identity Provider that maintains that attribute.

6.1.2.2 My Physical Identity Documents

Through the “My Physical Identities” option, the user can see his identity attributes that have been transferred to the Identity Consolidator through the Physical Identity Acquisition module. At first, the user can see all the official documents that she has used for physical acquisition, such as an identity card, a passport or a driving license (figure 58). After selecting a specific document, the values of the respective attributes are displayed, along with the confidence score and the Level of Assurance (LoA) for that specific document type (figure 59).

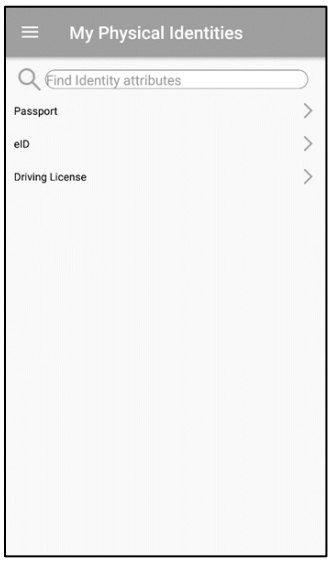

Figure 58. List of submitted Physical Identity documents

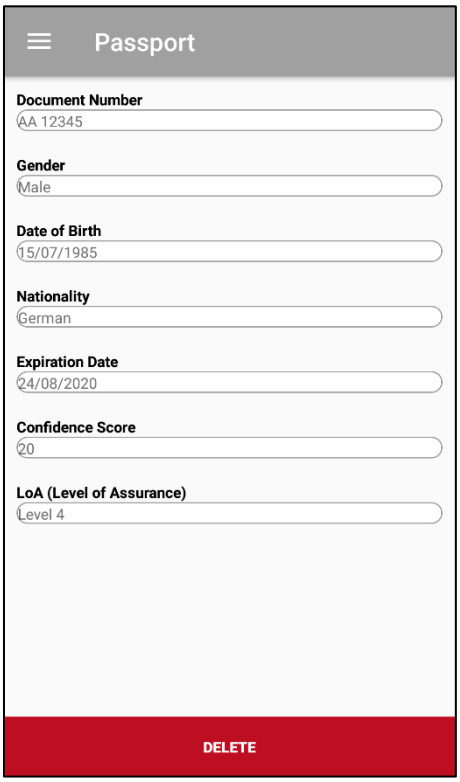

Figure 59. List of acquired identity attributes for a specific Physical Identity document

6.1.2.3 My Revealed Identity Attributes

Through the “My Revealed Identity” option, the user can see which Service Providers and Identity Providers know which aspects of his identity. This functionality is offered to the user through the following two approaches:

1. *View Service Providers that know an identity attribute:* The user can see a list with all his identity attributes (figure 60). After selecting a specific attribute, a new list is displayed, with all the Service Providers to whom that attribute has been revealed (figure 61). If an attribute has not been explicitly revealed to a Service Provider, the de-anonymization risk is displayed, as long as the selected attribute is not unique (e.g., email address or telephone number).

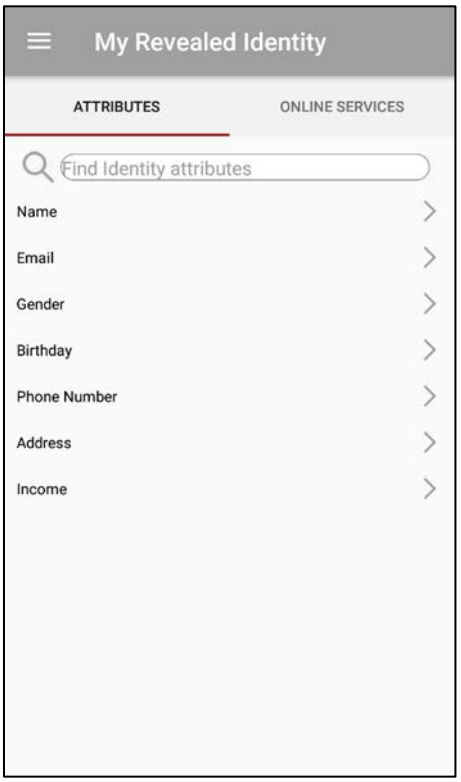

Figure 60. List of identity attributes

Figure 61. List of Service Providers that know an identity attribute

2. *View identity attributes known to a Service Provider:* The user can see a list of Service Providers, and for each Service Provider the number of the attribute values that it knows is also displayed (figure 62). After selecting a specific Service Provider, a new list is displayed, with all the identity

attributes and what the selected Service Provider knows or can infer about each attribute (figure 63).

Figure 62. List of the Service Providers that a user has shared identity attributes with

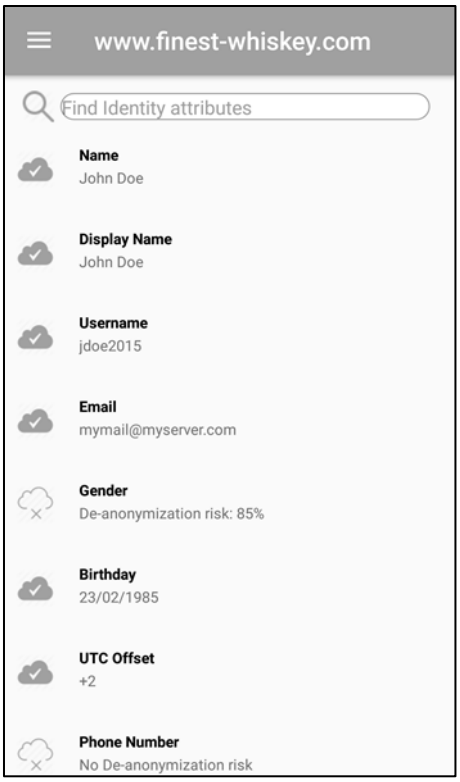

Figure 63. List of identity attributes known to a Service Provider

In both approaches, if the service provider knows the value of a given attribute, an icon appears next to the Service Provider, and the value of the attribute is also displayed. Otherwise, an icon appears, along with the de-anonymization risk.

6.1.2.4 Partially Verifiable Profiles

Through the “Partially Verifiable Profiles” option, the user can select a number of identity attributes and create Partially Verifiable Profiles (PVP) that will contain these attributes. These PVPs can then be presented to verifiers, depending on the context and the access control requirements.

At first, the user can see all the PVPs that he has created, along with the attributes they contain and the date they were last updated and this functionality is shown in figure 64 below.

In order to share a particular PVP, the user can tap on the icon, in which case a sharing dialog appears, showing the attributes included in the selected PVP, along with the actual URL that the user can use in order to share his partial verifiable profile (figure 65).

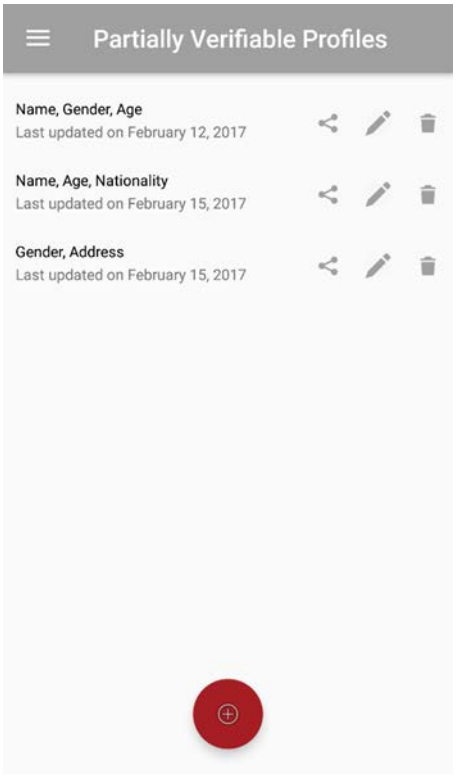

Figure 64. List of created partial verifiable profiles

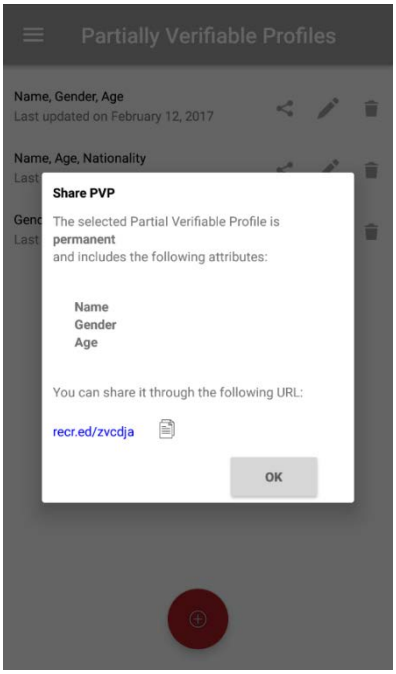

Figure 65. Partial verifiable profile share dialog

The user can also tap on the icon in order to modify any PVP or tap on the icon in order to delete it.

Finally, in order to create a new partial verifiable profile as show in figure 66 below, the user must select his identity attributes that will be included in the new PVP. The user must also select the type of the URL that will be produced (one-time, expiring, permanent).

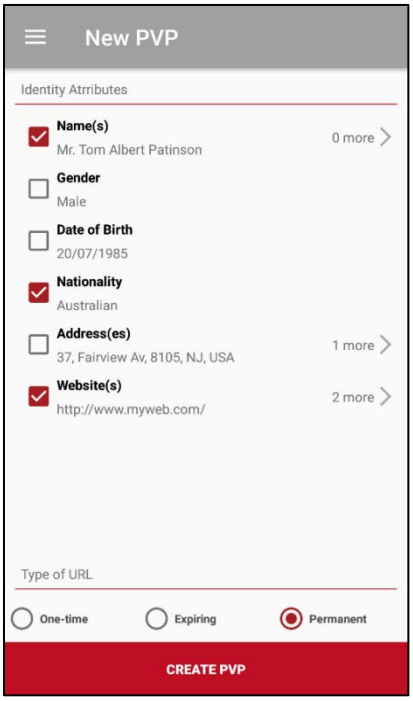

Figure 66. Create partial verifiable profile page

6.1.3 De-Anonymization Risk Assessment

Besides identity attributes view and management, the Identity Profile Management module offers the de-anonymization risk assessment functionality. This functionality has been integrated with the Identity Profile Management web application in order to provide a user friendly and more coherent User Experience.

In general, de-anonymization risk assessment functionality helps a user to protect his privacy with respect to confidence in unauthorized identity attribute value inference. De-anonymization risk is calculated for identity attributes related to Identity Providers, Service Providers, and for all the identity attributes that compose the financial information of a user. The de-anonymization risks that are taken into account involve risk of identity attribute value inference, and risk of user identification without his authorization.

Confidence in unauthorized identity attribute value inference is essentially the risk that an Identity Provider or Service Providers can infer the value of another identity attribute of a user that he has not shared with them based on the identity attribute values that the user shared with them and based on the distribution of the identity attributes across the web. In statistical terms it is an indication of where the user fits within the user population as this is segmented based on the revealed identity attribute values and the unrevealed identity attributes that the risk is calculated for.

The current implementation of this functionality addresses the risk of identity attribute value Inference for Identity Providers and for the financial information of the users. Implementation of risk calculation for the Service Providers and of risk calculation for user identification is still in progress.

Figure 67 and 68 below shows an example of the page that allows the users to view the risk calculation for Identity Providers.

Figure 67. De-anonymization risk calculation page for Identity Providers

Figure 68. De-anonymization risk calculation example for Facebook

Each one of the identity attributes where de-anonymization is applicable is associated with a confidence percentage that represents the possibility that an Identity Provider can infer the value of this attribute.

As shown in figure 68, a detailed view of the de-anonymization risk calculation for Identity Providers displays the confidence percentage next to each identity attribute that is applicable. A star ‘*’ symbol is used to indicate which attributes have been used for risk calculation, i.e. the *known* attribute values.

Calculation is based on the assumption that all the Identity Providers have acquired the distribution of identity attributes across the web from external means and this is distribution is at least the same as the one that we have in the Identity Repository of the Identity Consolidator platform.

As mentioned before, de-anonymization risk assessment is also calculated for all the identity attributes related to the financial information of the user stored in the Identity Repository. An example of the risk calculation for those attributes is shown in figures 69 and 70 below.

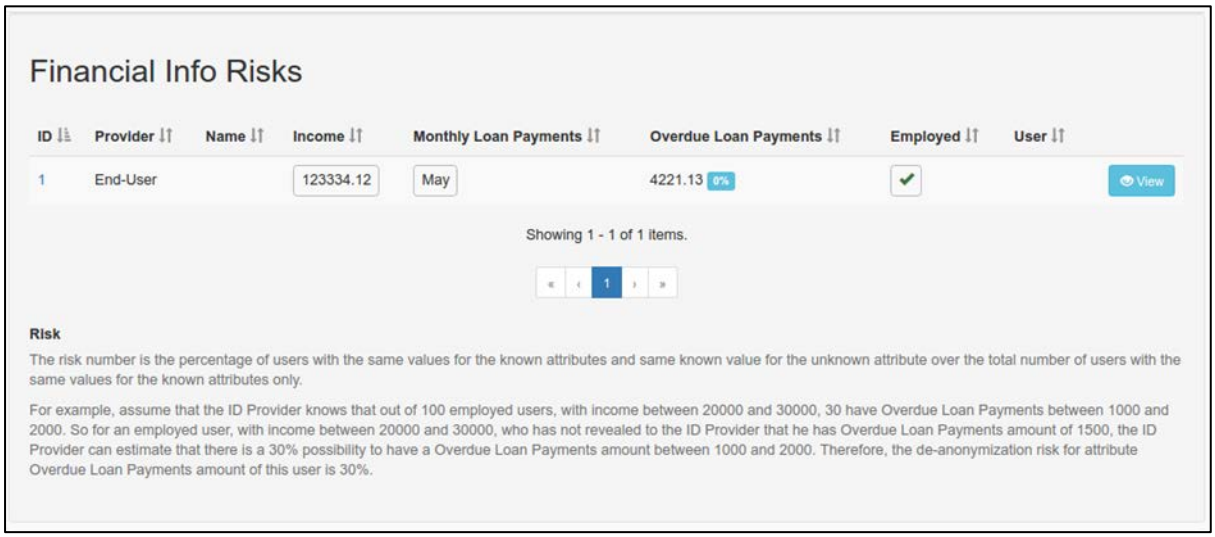

Figure 69. Financial information identity attributes de-anonymization risk

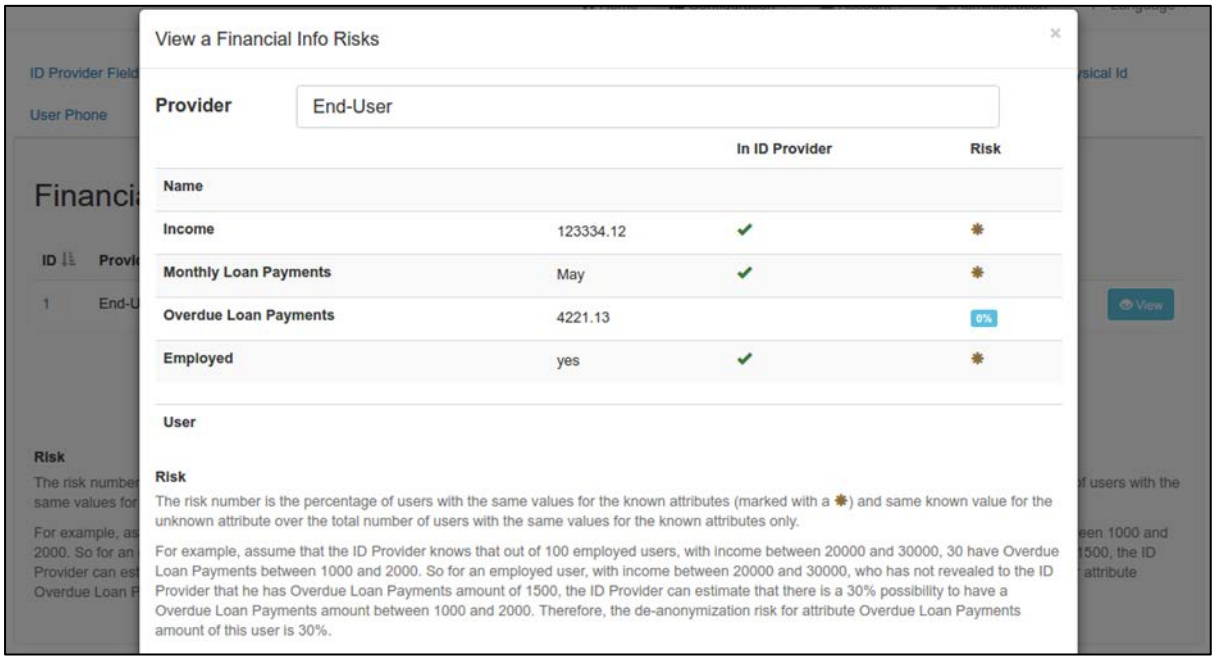

Figure 70. Financial information identity attributes de-anonymization risk calculation

In general, for the implementation of this functionality for de-anonymization risk calculation for ID Providers and Service providers is separated in two different categories depending on the method that the user is using to prove the (holding of identity attributes)… and access the Service.

These categories are the following:

Vanilla OpenID Connect: The first is when a user uses the vanilla OpenID Connect to prove an identity attribute to a Service Provider. With OpenID Connect we have no untraceability and any Service Provider or Identity Provider can track the user across the web. In this case we want to protect the user against Service Providers and Identity Providers that has a unique identifier for each user and slowly by retrieving the identity attributes of each user they are in place to build a complete profile for each user. As a result, anywhere that a user goes to the web the Identity

Providers that authenticates him are in place to trace him. The only thing we can do here is to prevent Identity Providers and Service Providers to build the complete profile of a user.

In order to achieve this, the Identity Profile Management module needs to be aware of the identity attributes shared with what Service Providers either from the Identity Consolidator or from an external Identity Provider. Such information will be provided by OpenAM, which keeps logs each time a user shares an identity attribute with a Service Provider.

In general, the only anonymity that we can provide with vanilla OpenID Connect is that we can produce the confidence probability whether a Service Provider can infer the value of an attribute, which the user has not revealed to this Service Provider. So as soon as this functionality is implemented the user should be able anytime to go to the Profile Management module and see that will happen if he revealed a specific identity attribute or a combination of identity attributes. We should call this Privacy with respect to Confidence in unauthorized attribute value inference.

Idemix and U-Prove: The second case is when a user uses ABAC (Idemix and U-Prove) to prove that he is an identity attribute and access a Service Provider. Using ABAC we have untraceability and unlinkability. With Idemix we have the Verifying Identity Providers who run an Idemix stack. In this case the user runs idemix with the verifying Identity Provider and with an idemix credential he proves to this Identity Provider one of his identity attributes or a combination of identity attributes (e.g., his age) and then the Identity Provider assures the Service Provider that the user is the holder of a credential that proves his age.

With Idemix we have two concepts of anonymity (untraceability and unlinkability - untraceable and unlinkable credentials). This is because when the same user shows a combination of three attributes to an Identity Provider and Service Provider at time A, if the same user go to another Service Provider and prove the same three attributes at time B there is no mechanisms that can infer or assure that I am the same person as the one at time A. This means that I am untraceable and my credentials cannot be linked. This also means that no Identity Provider or Service Provider can built a complete profile for me.

Based on the aforementioned, in the case of Idemix the risk assessment is different and we need to calculate the risk per session. Additionally, with Idemix the calculation of the risk for de-anonymization does not depend on the credentials that a user shared before with Service Providers. Also, it does not depend on the policies that a Service Provider may have for access control but on the specific credential (identity attribute) or combination of credentials that a user is about the reveal to a Service Provider.

When we are talking about risk assessment with Idemix we need to know whether a given attribute can be inferred by a Service Provider and what is the confidence probability based on the combination of the credentials that a user is about to share with this Service Provider. For this we assume that all the Service Providers and Identity Providers have acquired the distribution of identity attributes and the distribution of the combinations of identity attributes (the frequency this combination of attributes occurs in the complete population) from external means.

For the computation of risk for de-anonymization in the case of Idemix we need to make sure, for each session separately, that when a user reveals some identity attributes (credentials) to the verifying Identity Provider or the Service Provider they does not learn any other of the user's identity attributes that he does not want to reveal to them. In general, we want whenever a user goes to an Identity Provider to inform him for example –given the distribution of identity attributes - that this

combination of identity attributes that you are going to reveal are also appear 10 times across the population and if you reveal them, the Service Provider has the knowledge to guess the values of some more of your identity attributes. This computation should be offered by the Identity Profile Management module and the Idemix client that runs on the verifying Identity Provider should invoke the Identity Management module and display the calculated risk that the given Identity Provider or Service Provider can infer other identity attributes of the user.

6.2 Consent Management

The Consent management is part of the Identity Management module and allows users and Identity Providers to their consent for their various identity attributes. Moreover, the Consent management is divided to the following:

- *User-defined policy for attribute transfer and proof:* Such functionality is used when the user wants to define policies about to which Identity Providers and Service Providers/verifiers, their attributes should be revealed. The Consent Management will provide the user with the ability to involve the LoA in the policy decision process. For example the user may define that it does not wish to reveal their address to online social network services or any service provider under certain LoA.
- *Identity Provider-defined policy for attribute transfer and proof:* It is responsible for obtaining policies (with respect to what identity attributes can each verifier see) for individual attributes from identity providers ensuring that the Identity Consolidator reveals attributes to verifiers according to these policies. The Consent Management enables the Identity Providers to also involve the LoA in the policy decision process. This is used, especially from ID Providers who do not wish certain attributes, or attributes with specific LoA, to be revealed to certain unauthorized parties or other identity providers. For example, the Social Security Administration (ID provider) provides the social security number that should be revealed only to reputable verifiers, such as banks. Or an IDP may decide that it does not want the attributes of its users to be proven using idemix/u-prove and that they should be proven through the IDP via OAuth instead (so that the IDP always knows where these credentials have been shown).

The Credential Management module is responsible for abiding by these set of policies (for example not issue Idemix credentials to the device if the Identity Provider specified so). The Authentication Management module running on the device also has to abide by this policy. The Identity Profile Management module also has to respect the IDP's and the users' created policies regarding identity attribute transfer between different Identity Providers.

6.2.1 Consent Management Back-end

The Identity Consolidator provides interfaces to define user consent profiles. An example of a consent rule that might be defined in one such profile might be: As a user, I do not want to share my Date of Birth with Service Provider(s) Foo and Bar.

This consent rule will mean that when a user tried to access service Foo and if service Foo requests the users Name, Email address and Date of Birth, the request that will be presented to the user will consist of a request for approval to disclose their Name and Email address ONLY. The use will be be asked to share their Date of Birth, even though the Service provider has requested this information. The SP can still request a different form of this information though, for example asking that the user

prove they are over 16 years of age. The identity consolidator, in this context, presents the user will a form asking for their consent to share this information.

All the data related to the user defined policies is stored in the Identity Repository by utilizing the API calls offered by the Storage API and accessible through the Identity Consolidator REST interfaces.

6.2.2 Consent Management Web Front-end

When users initially identified themselves and share identity attributes with a Service Provider this approval is stored. Users who has logged into their ReCRED Identity Consolidator account, they can access the Consent Management from the “Consent” tab, which is under the “My Account” page. From this tab a user can perform the following actions:

- Review previously granted consents
- Modify consents that have previously been granted
- Define new consent policy rules

A historic record is kept which shows what information has been shared in the past with a Service Provider and what data is currently shared / accessible by a Service Provider.

6.2.3 Consent Management Mobile Application

The Consent Management mobile front-end has been implemented as an Android mobile application that allows the users to define their consent for their various identity attributes by defining policies regarding the Identity Providers and Service Providers to which their attributes should be revealed. More specifically, the functionality regarding the consent management is included in the general Identity Profile Management android application, providing a seamless user experience through a unified application for managing the various aspects of a user’s identities.

The mobile app communicates with the Consent Management back-end, allowing the end-users to access the following functionality:

- Create new consent policies, by hiding specific identity attributes (or groups of attributes) from Identity Providers and/or Service Providers
- View the consent policies that he has created, and group them by attribute, Identity Provider or Service Provider
- Modify or delete consent policies that he has already created

6.2.3.1 Create new Consent Policy

The user can create new consent policies through a two-step procedure: the user defines the attribute(s) that he wants to hide and then he defines the Identity Provider(s) or Service Provider(s) from whom he wants to hide these specific attributes (figure 71). Hiding an attribute from an Identity Provider means that this attribute cannot be transferred to that Identity Provider (or revealed to it, in cases where the Identity Provider might act as a Service Provider). Hiding an attribute from a Service Provider means that this attribute cannot be transferred to that Service Provider.

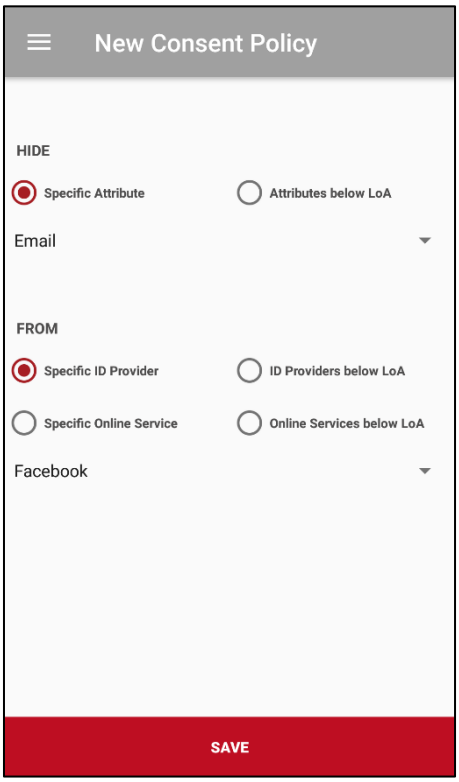

Figure 71. Create new consent policy page

More specifically, at first the user must select the attributes that he wants to hide. There two options here:

1. The user can select to hide a specific identity attribute, which he can select from a drop-down list (figure 72).
2. The user can select to hide all his identity attributes that are below a certain Level of Assurance (LoA), which he can also select from a drop-down list (figure 73

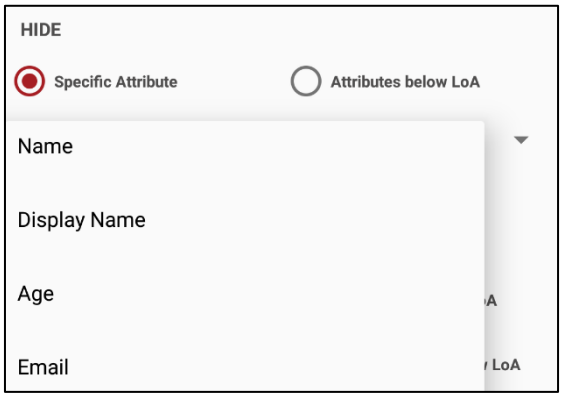

Figure 72. Selection of specific identity attribute

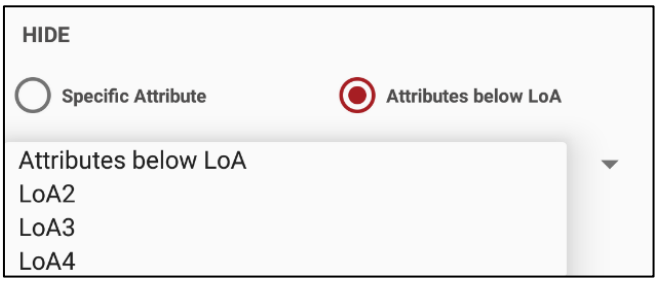

Figure 73. Selection of identity attributes below a certain LoA

After that, the user must select the Identity Providers or Service Providers from whom the selected attribute(s) will be hidden. Here, there are four possible options:

1. The user can select a specific Identity Provider (figure 74)
2. The user can select all the Identity Providers below a certain LoA (figure 75)
3. The user can select a specific Service Provider (figure 76)
4. The user can select all the Service Providers below a certain LoA (figure 77)

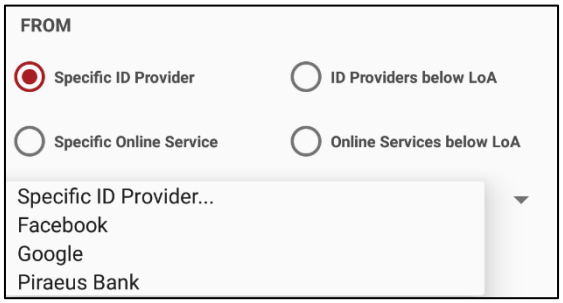

Figure 74. Selection of specific Identity Provider

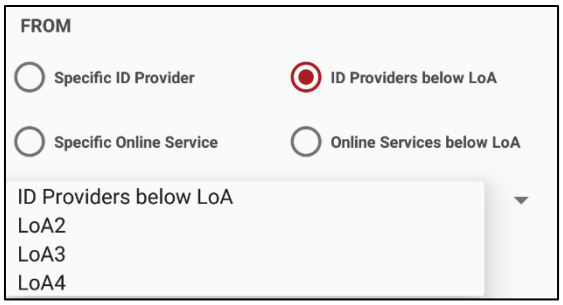

Figure 75. Selection of Identity Providers below a certain LoA

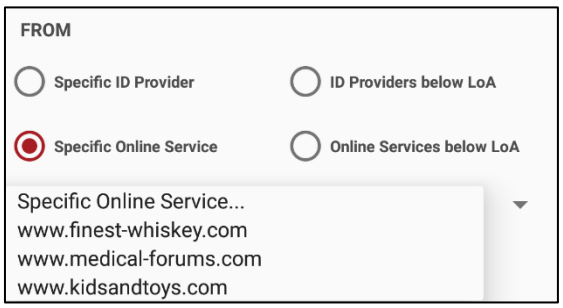

Figure 76. Selection of specific Service Provider

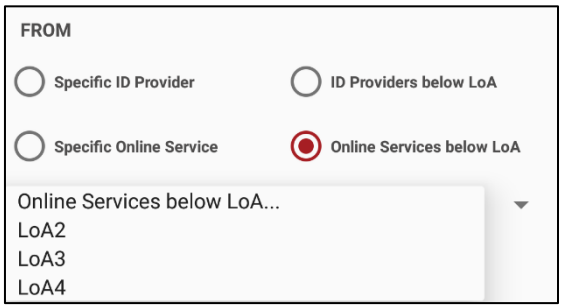

Figure 77. Selection of Service Providers below a certain LoA

6.2.3.2 View created Consent Policies

The user can see a list with all the consent policies that she has created. These attributes are grouped under three different sections:

1. View created consent policies by an Identity attribute

Here, the user can see a list with all his identity attributes and how many consent policies he has created for each attribute (figure 78). After selecting a specific attribute, the user can see all the consent policies that he has created for it (figure 79), either explicitly for that attribute or for an LoA where the attribute is included.

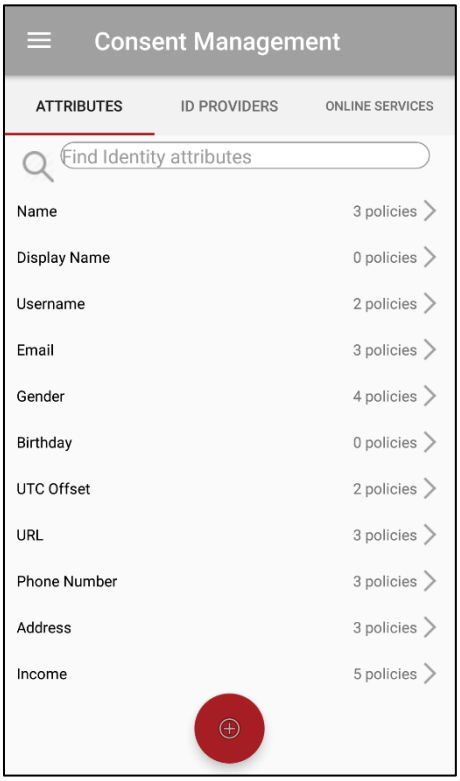

Figure 78. List of a user's identity attributes

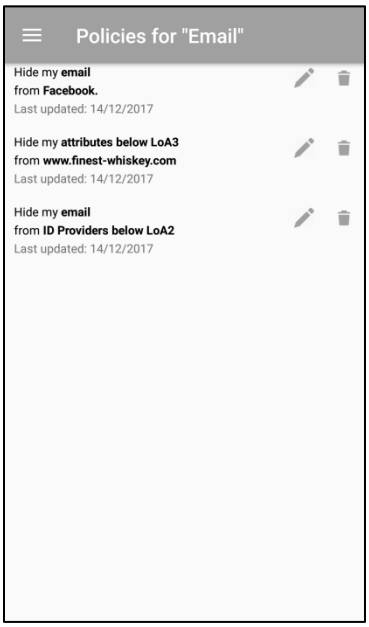

Figure 79. View created policies per identity attribute

2. View created consent policies by an Identity Provider

Here, the user can see a list with all his Identity Providers and how many consent policies he has created regarding that Identity Provider (figure 80). After selecting a specific Identity Provider, the user can see all the consent policies that he has created in order to hide attributes from it (figure 81), either explicitly for that Identity Provider or for an LoA where the Identity Provider is included.

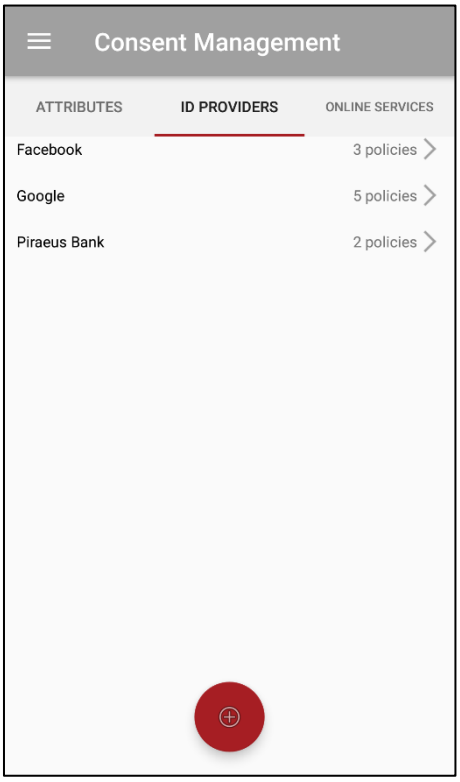

Figure 80. List of Identity Providers

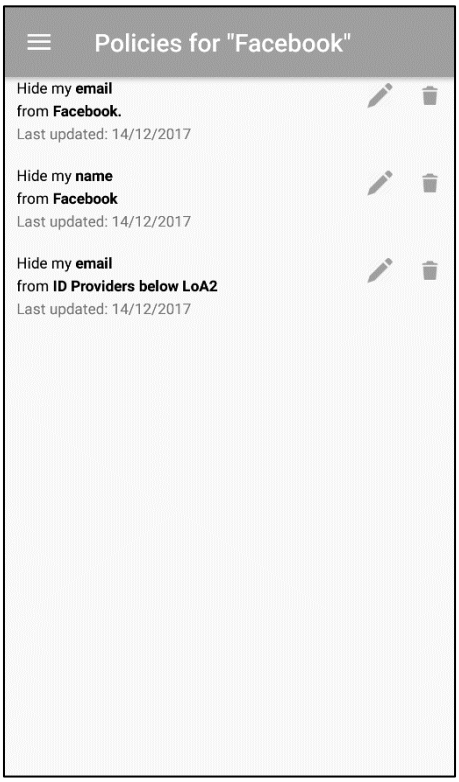

Figure 81. List of created policies per Identity Provider

3. View created consent policies by Service Provider

Here, the user can see a list with all his Service Providers and how many consent policies he has created regarding that Identity Provider (Figure 82). After selecting a specific Service Provider, the user can see all the consent policies that he has created in order to hide attributes from it (figure 83), either explicitly for that Service Provider or for an LoA where the Service Provider is included. The user can also modify or delete any policy.

Figure 82. List of Service Providers

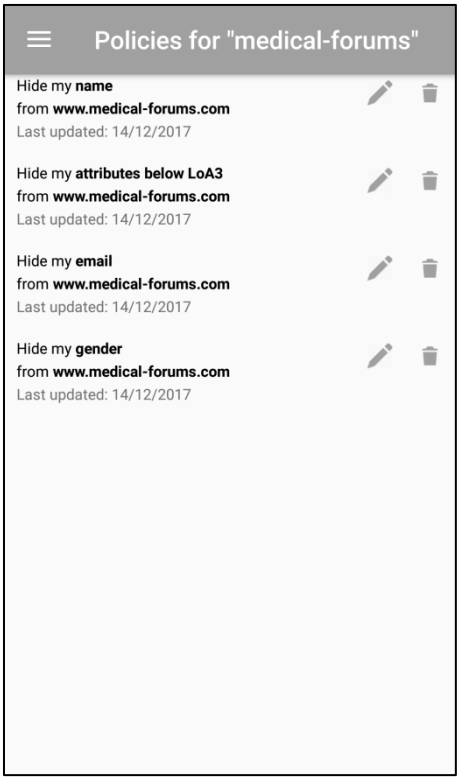

Figure 83. List of created consent policies per Service Provider

6.2.3.3 Manage created consent policies

At any point while viewing her consent policies, the user has the option to modify them or even delete them.

In order to modify a policy, the user can tap on the icon beside it, in which case the *Edit Consent Policy* screen is appeared, filled-in with the details of the selected policy. The user can modify the policy as he wishes and save her changes.

In order to delete a policy, the user can tap on the icon beside it, in which case a confirmation dialog is shown and the consent policy is deleted.

7 Online Identity Acquisition Module

The Online Identity Acquisition module obtains the user identity information from different Online Identity Providers, Facebook and Google. This info can be obtained in two manners. On the one hand, the identity acquisition can be initiated by the end-user. In this case the module relies on a registration process based on OAuth2 to gain access to the identity provider and retrieve the end user information. On the other hand, the user can grant permission to the module to obtain its identity information from a service through crawling techniques. We refer to the former as user assisted online identity binding process and the latter as non-assisted online identity biding processes. Note that the assisted process provides more guarantees with respect to the veracity of the collected data since the OAuth2 authentication process requires the user to register in the online identity provider with his/her credentials before proceeding to the identity info collection.

The information obtained by the ID Acquisition Module from the ID Providers supported, is stored in the Identity Repository of the Identity Consolidator (IDC). In addition, the Identity Integration Module integrates the info of a user coming from different providers using as reference the algorithm described in Section 11.1.

Figure 84 shows the homepage of the Online ID Acquisition Module. The current implementation of the module includes two Online Identity providers: Facebook and Google. For each provider, a user can choose to provide the information using the two options described above: assisted represented by the “Login” buttons and non-assisted, represented by the “Crawler” buttons.

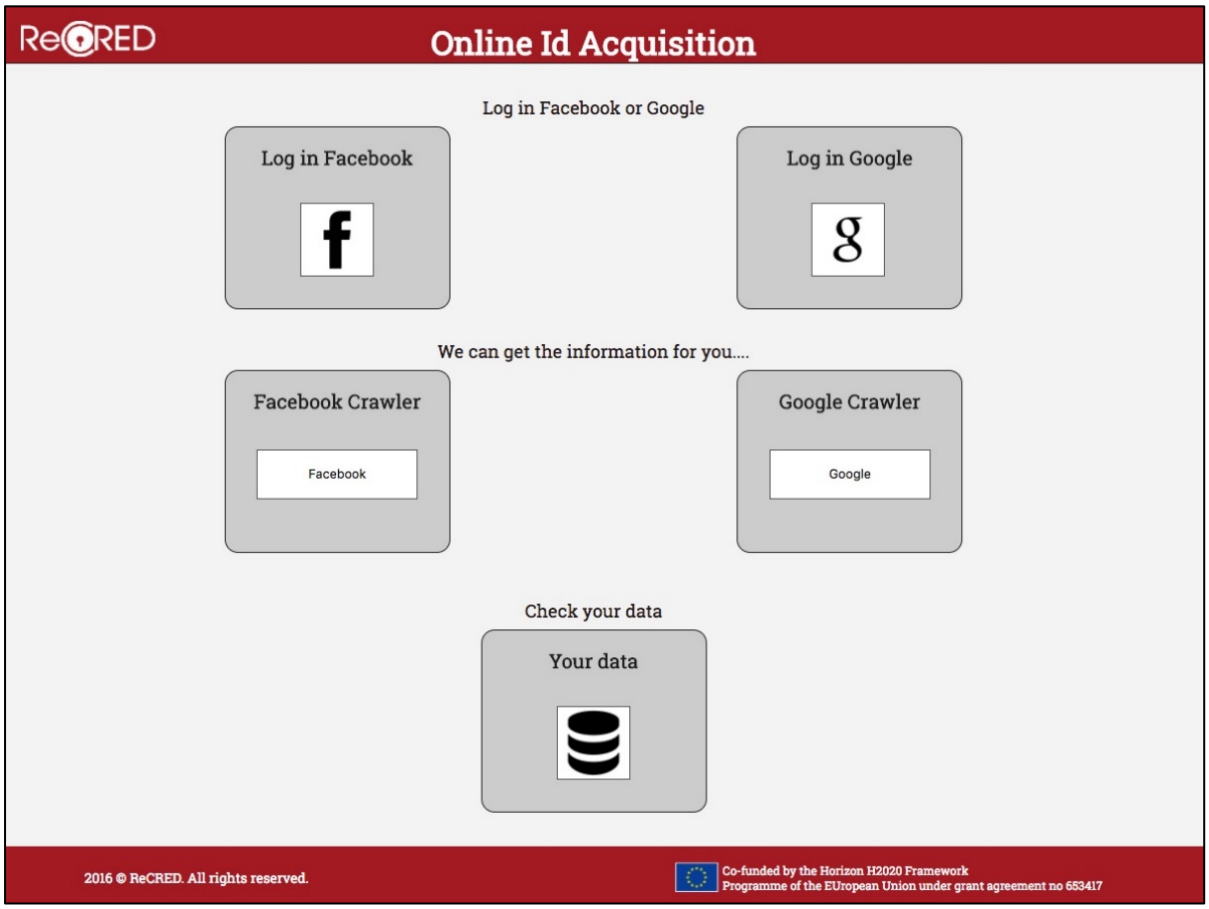

Figure 84. Online Identity Acquisition module main page

7.1 User assisted online identity binding process

The user assisted online identity binding process relies on an authentication process executed via the Online Social Network OAuth2 protocol which is required in order to login in several online services such as Facebook or Google.

7.1.1 Identity Information Acquisition process

Once the login has been conducted, the user is notified about the data that the service (in our case the online identity acquisition module) is requesting to obtain. The user should give explicit authorization to our service to access the notified user’s personal information. Upon the authorization of the user, our module obtains a allowing it to collect the information, and thus it proceeds to the attributes acquisition. Some attributes available in the considered identity providers (Facebook and Google) are date of birth, location, education, occupation, etc. These attributes are stored to the Identity Repository of the IDC through the Storage API. Figure 85 and figure 86 show the login access to Facebook and Google through the Online Identity Acquisition Module.

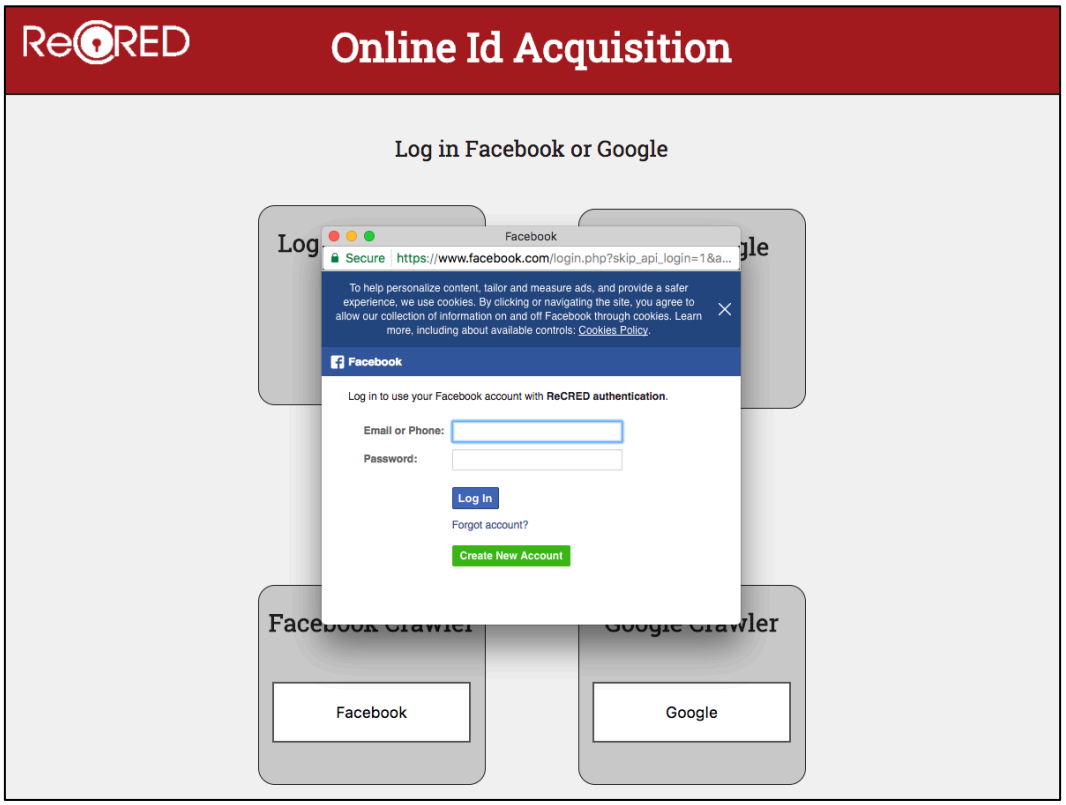

Figure 85. Online Identity Acquisition Facebook Login

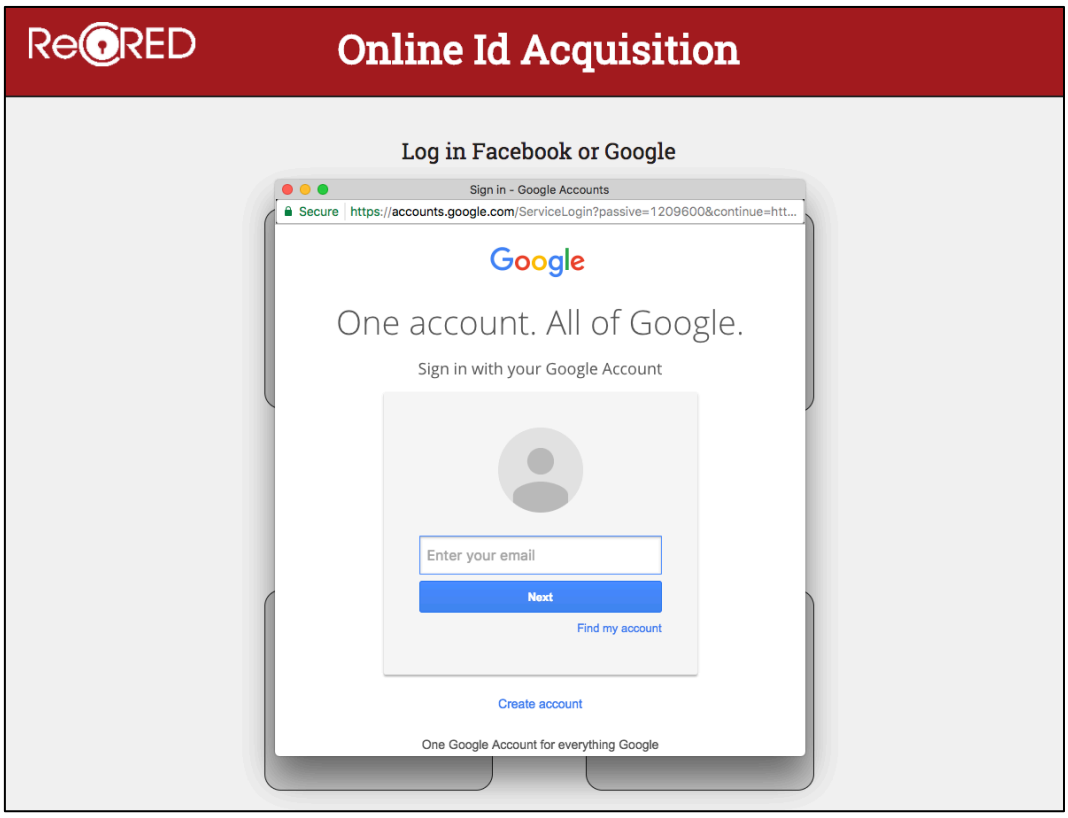

Figure 86. Online Identity Acquisition Google Login

7.1.2 Interaction with the Identity Repository

The identity information collected from the different ID providers is stored in the Identity Repository through the Storage API. In particular, the table storing this information is Identity Providers Data Table described below in Section 9.

Note that the information stored in the Identity Repository for a given ID provider is updated every time the user is authenticated in such provider.

7.1.3 Access to the Identity Providers Information

As indicated above, the access to the Identity Providers considered in the current prototype (Facebook and Google) is implemented using the OAuth2 protocol, an open standard for authorization. The OAuth 2.0 enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf RFC6749 [19].

7.1.4 Authorization and Privacy Management

The IDC through the Online Identity Acquisition module request users their authorization to collect information from their profiles in the ID provider (i.e., Facebook or Google). In the case of Facebook, in the authorization process, each attribute has a checkbox associated. The user should select only those attributes he is willing to share with the IDC. In the case of Google, the user cannot make such fine-grained selection. Instead Google informs the user about the attributes to be collected and the user can grant binary permission to collect either all or to none identity attributes. Figures 87 and 88 show the authorization process for both ID providers, Facebook and Google.

Therefore, the assisted binding process of the Identity Acquisition Module respects the privacy of the user since information is only collected upon the explicit authorization of the user.

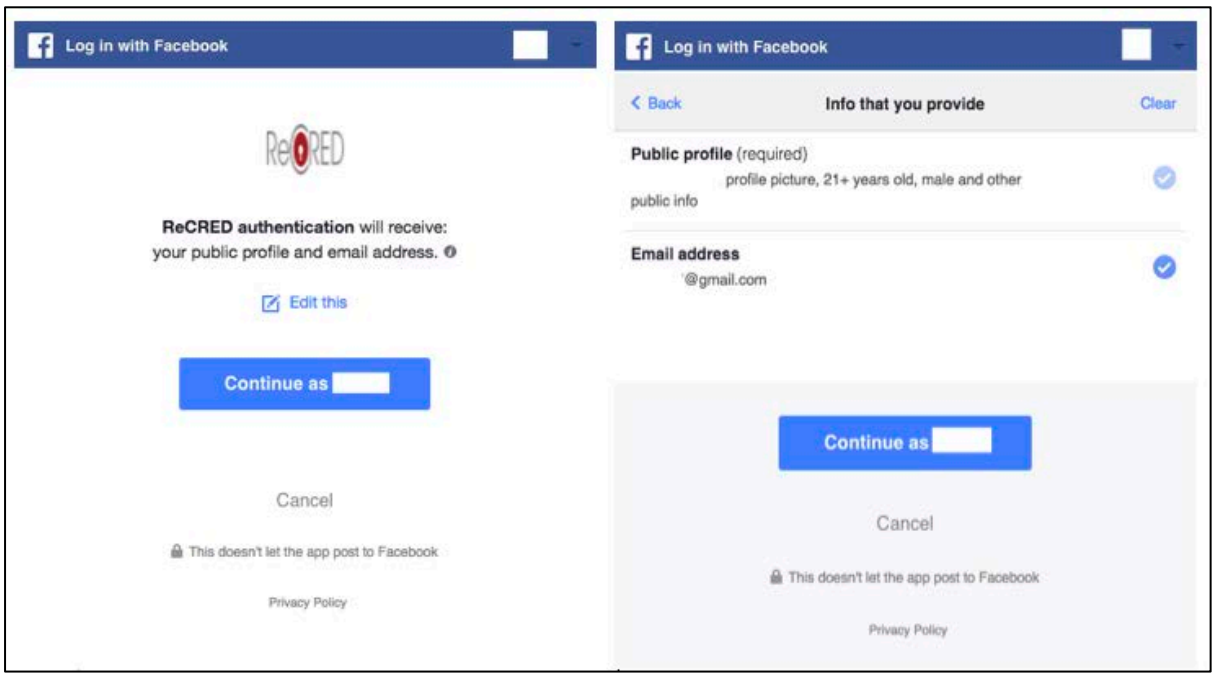

Figure 87. Facebook Login permissions authorization dialogs

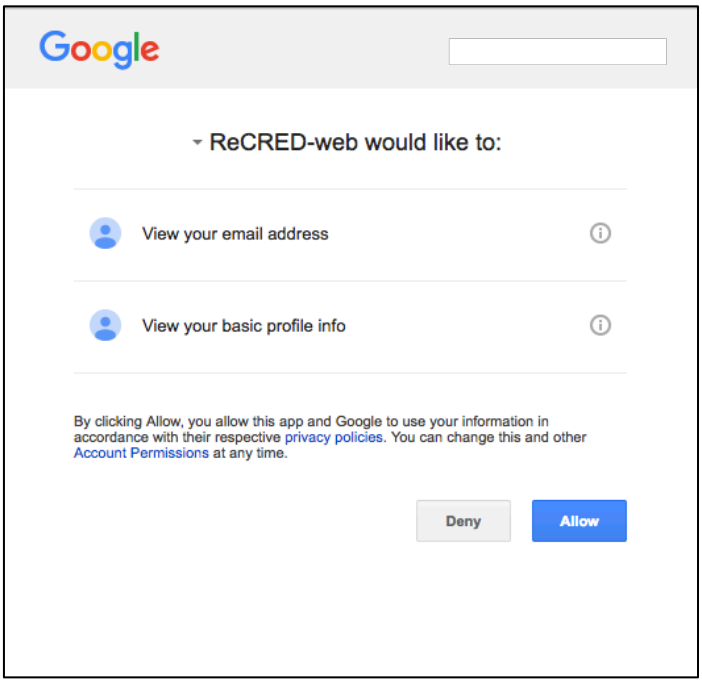

Figure 88. Google Login permissions authorization dialog

7.2 Non-user assisted Online Identity binding process

The non-user assisted online identity binding process is based on crawling techniques and then, it does not require user authentication. Instead, this process requires the user’s consent to retrieve his information through crawling techniques.

7.2.1 Acquisition process

The user triggers this process by clicking on the “crawling” buttons for Google or Facebook in the home page of the Online Identity Acquisition Module. Then, a pop-up message is shown to the user requesting: 1) his/her authorization to retrieve the public information available in its profile in the correspondent ID provider (i.e., Google or FB), 2) the URL of its Facebook (or Google+) account. After this, the Online Identity Acquisition Module checks if the provided URL is correct and launches the crawling process, which retrieves the public information from the profile associated to the URL provided by the user. The crawler is launched using a POST request to select which of the crawlers (the one for Facebook or the one for Google) should be run. The collected information is stored in the Identity Repository through the Storage API.

Figures 89 and 90 show the user interfaces for the crawler option, and the pop-up message shown to the user to authorize the data collection process.

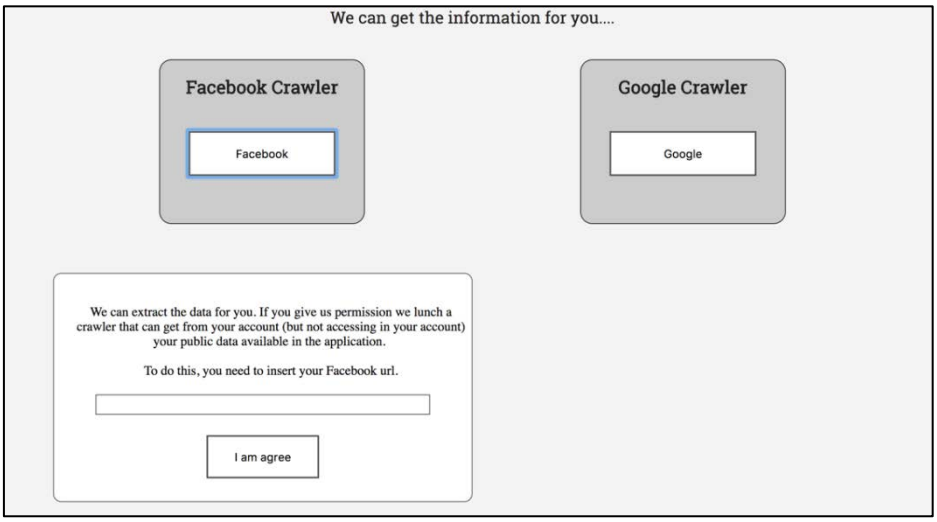

Figure 89. Online Identity Acquisition Facebook crawler

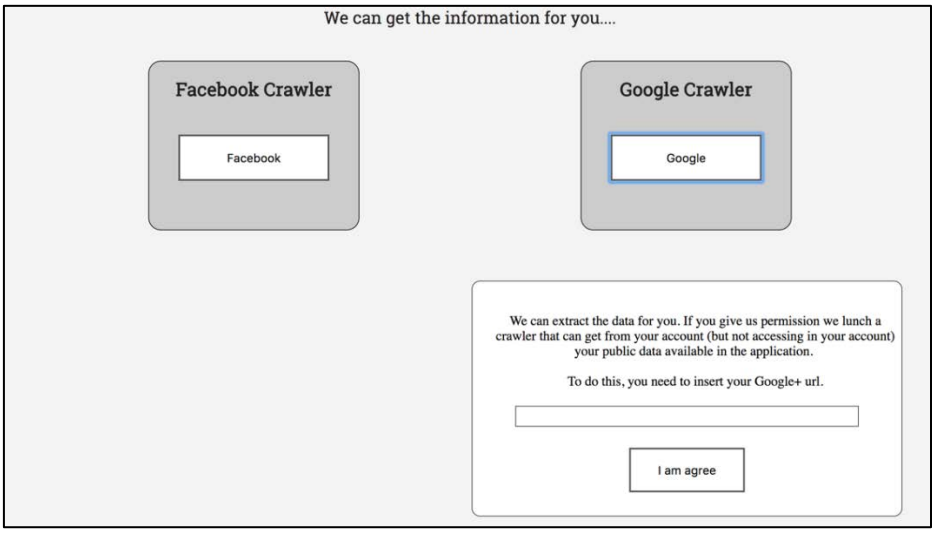

Figure 90. Online Identity Acquisition Google crawler

7.2.2 Interaction with the Identity Repository

The identity information collected from the different ID providers with crawling processes is stored in the Identity Repository through the Storage API. In particular, the table storing this information is Identity Providers Data Table described below in Section 9.

Note that the information stored in the Identity Repository for a given ID provider is updated every time the user voluntarily triggers the crawling process.

7.2.3 Access to the Identity Providers Information

To access the information available in the public profile of a user we use a crawling based on Selenium WebDriver [20], a tool for browser automation. Selenium WebDriver sends commands to a browser through a driver and retrieves the desired results, e.g. open a Facebook user page and get his username.

7.2.4 Authorization and Privacy management

In the case of the non-assisted binding process, the crawler to retrieve the information is launched only if the user has provided the consent to do so. In particular, the pop-up message informs the user that the information collected would be the public information available in its Facebook (or Google+) profile. Note that this information may vary across users depending on the privacy configuration set up of each user account.

Therefore, the non-assisted binding process of the Identity Acquisition Module respects the privacy of the user since information is only collected upon the explicit authorization of the user.

7.3 Supported Functionalities

This subsection describes the functionalities that the ReCRED platform offers to end-users through the Online Identity Acquisition Module:

- Connect Online Account: The user can connect to the IDC an online account he owns, so that all his online accounts are eventually consolidated. In the case of an assisted binding process, an OAuth Connect connection is established, so that the user needs to be authenticated in the ID provider. In other words, the IDC issues a Proof of Account Ownership upon successful authentication to the service. In the case of non-assisted binding process, the proof of account ownership is not required, since the account considered is one provided by the user through an URL.
- List Connected Accounts: The user is presented with a list of all the online accounts that have been connected to the IDC.
- Transfer User Data to the IDC: The user can authorize the retrieval of of data from a selected connected online account to the IDC (through an OAuth / OpenID Connect connection or through a crawling process).
- Disconnect Connected Account: The user can select a connected online account, in order to disconnect it from the IDC. Any data associated to that account is deleted from the IDC.

Figure 91 shows the web interface listing all the connected Online Accounts and the option to disconnect each single account from the IDC.

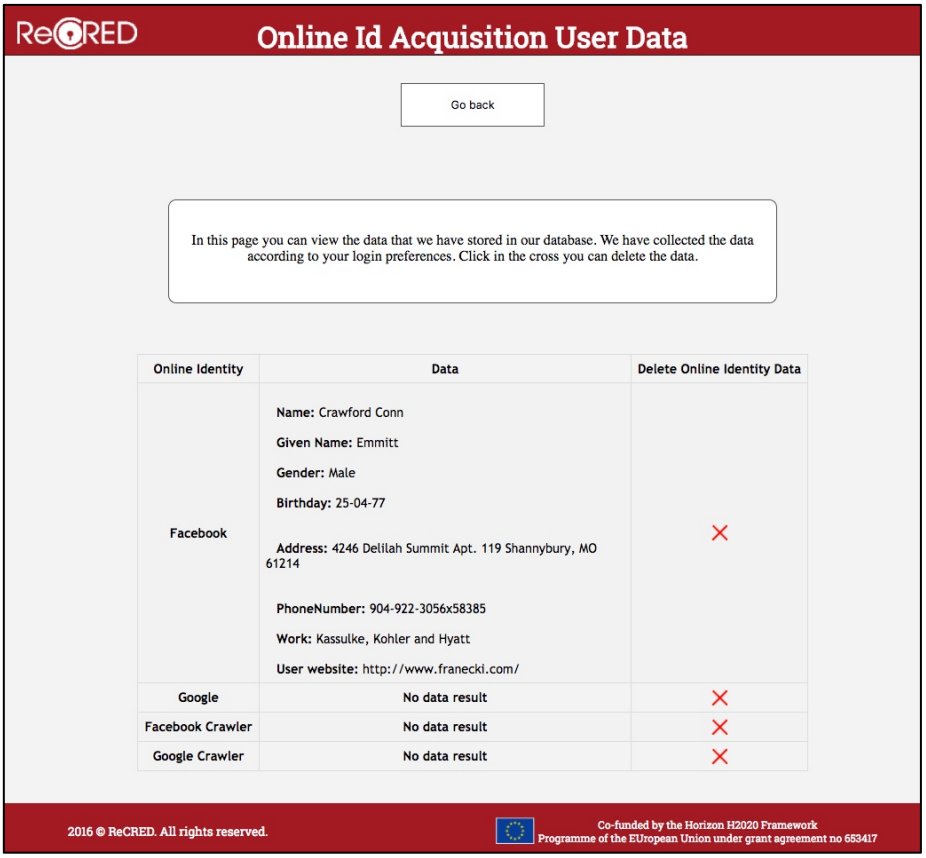

Figure 91. List connected Online Accounts page

8 Physical Identity Acquisition Module

The Physical Identity Acquisition module of the Identity Consolidator is responsible for vertically binding the real-world identities of the user. Its main purpose is to collect all the identity information in the real-world identities of the user, verify them and convert them into independent verifiable identity attributes. This module has been implemented as a web application on desktop computers and as an application on smart trusted-computing-enabled mobile devices. The Physical Identity Acquisition module is subdivided into two main processes, the identity acquisition and the identity verification process.

The identity acquisition process is the one that requires users’ involvement at most and involves the acquisition of the physical characteristics of the users as well as the identity information included in their physical identity documentation (e.g., Identity Card, Passport, etc.). This process makes use of the trusted paths on the devices in order to enable the users to securely capture photos of their face and their physical identity documentation. Additionally, the physical characteristics of the users are captured (such as location) by exploiting existing techniques and the available sensors on the devices. All the captured pictures are encrypted and securely stored in a protected directory in the Identity Consolidator server while the acquired identity information are securely stored in the Identity repository through the Storage API.

The identity verification process uses various secure methods for the verification of all the acquired users’ identity information. More precisely, for the identity verification processes we use crowdsourcing techniques and automated means (such as OCR) in order to verify all the collected

identity information, pictures, and the physical characteristics of the users. Identity verification process also includes the verification of the acquired identity information by trained auditors with Face-to-Face identity verification. Figure 92 below depicts both the Identity Acquisition and Identity Verification processes from the beginning until the user has successfully verified his real-world identity.

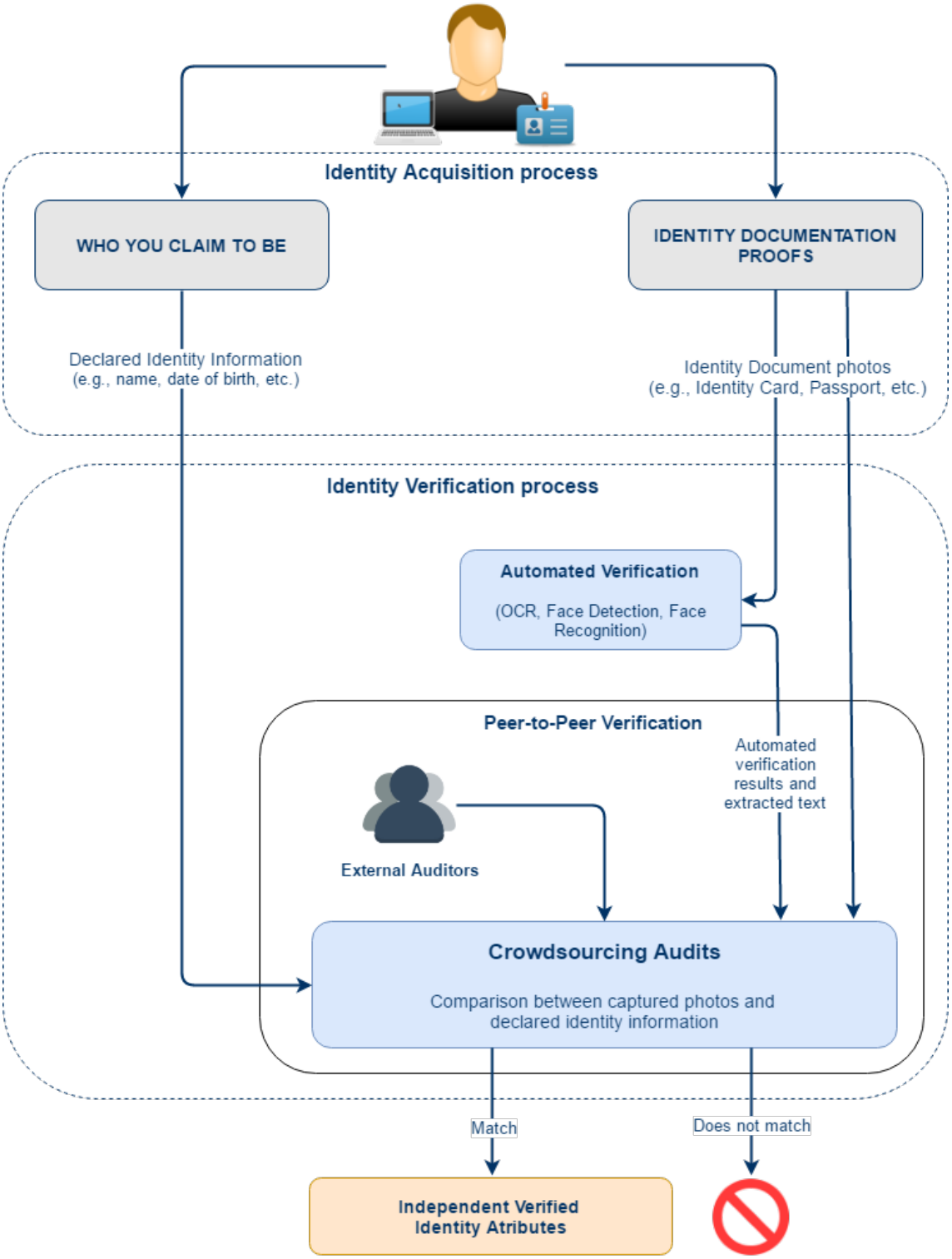

Figure 92. Physical Identity Acquisition and Verification processes

We have implemented the Physical Identity Acquisition module as a web and mobile application. We have developed the web application in PHP using Laravel [9], a PHP Framework for web development. The application is running on an Apache HTTP Server and can be accessed through the Identity Consolidator’s main page at https://consolidator.recred.eu.

In general, the Physical Identity Acquisition module comprises the following functionalities:

1. Physical Identity Acquisition process
   - Identity Document submission for physical identity acquisition process
   - View/manage identity information and uploaded pictures for each identity document
   - Real-time video presentation of Identity documents using WebRTC [10]
   - RFID Identity document scanning with NFC-enabled devices [11]
2. Physical Identity Verification process
   - Face-to-Face identity document verification by trained (f2f) auditors
   - Address verification
   - Peer-to-peer identity document verification using crowdsourcing techniques
   - Automated verification
   - View the results of the verification process for each identity document

8.1 Physical Identity Acquisition

Physical identity acquisition enables users to initiate the process for verifying their real world identities. Currently, the developed Physical Identity Acquisition module allows users to verify the following Physical Identity documents:

i. Identity Card
ii. Passport
iii. Profession Certificate
iv. Driving License

When choosing to verify a specific identity document the application requests from the user to declare the identity information included in the document like his name, surname, document number, date of birth, etc. After declaring the requested information, the user is requested to capture a photo of his whole identity document from his device’s camera. The, the user is requested to crop multiple parts from his identity document photo and also capture photos of his face from the device’s camera. Furthermore, physical identity acquisition includes the acquirement of additional physical characteristics of the users (e.g., his location) for verification purposes that will be described in the identity verification section. Physical Identity Acquisition process allows users to declare his work and/or home address. For the verification of these addresses the application allows users to choose between two different methods that will also be described in the next section. Figure 93 below shows the home page of the web application of the Physical Identity Acquisition.

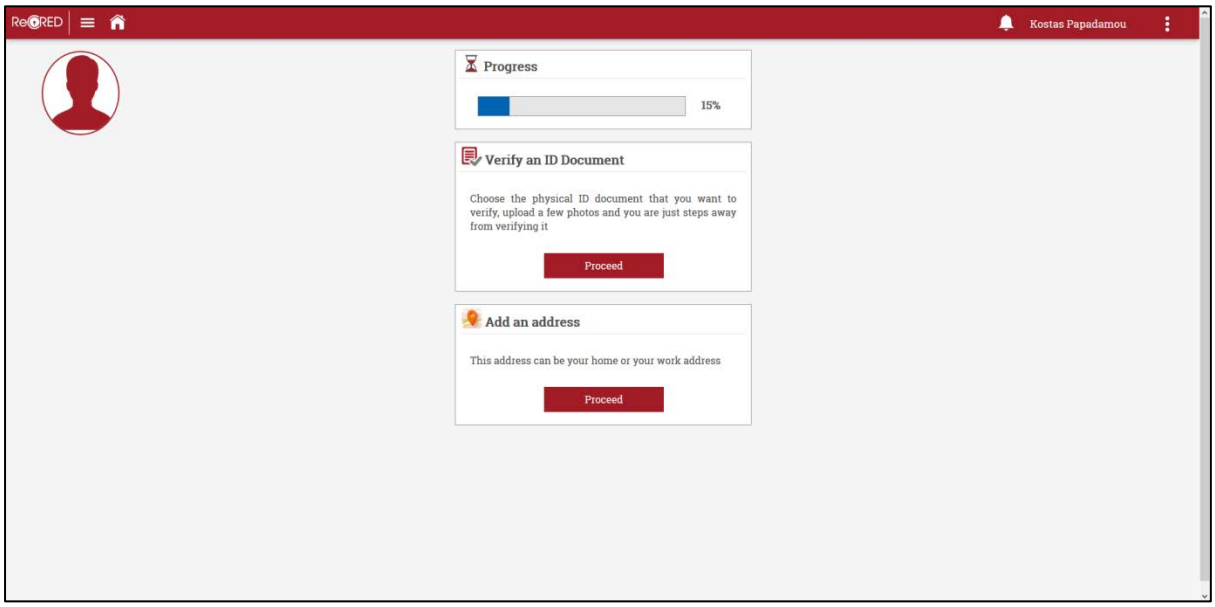

Figure 93. Physical Identity Acquisition web application home page

Moreover, the Physical Identity Acquisition module offers to the users some additional methods to declare and verify their identity attributes. First, a user is able to present his identity document to a verifier via real-time video conference using WebRTC. The second method integrates the IRMA [] findings and allows users to scan their NFC-enabled identity documents. This functionality enables the ease and secure acquirement of identity attributes without any further verification.

Below we describe each one of the supported functionalities of the Physical Identity Acquisition process.

8.1.1 Initiate Physical Identity Acquisition process

Through the physical identity acquisition application the user is able to initiate the identity acquisition and verification process for various identity documents. Initially, the user is able to choose among his identity card and passport. As soon as he has verified one of the two identity documents then he is able to initiate the verification process for his profession certificate or his driving license. Figure 94 below shows the page where the user is able to choose the document that he wants to verify.

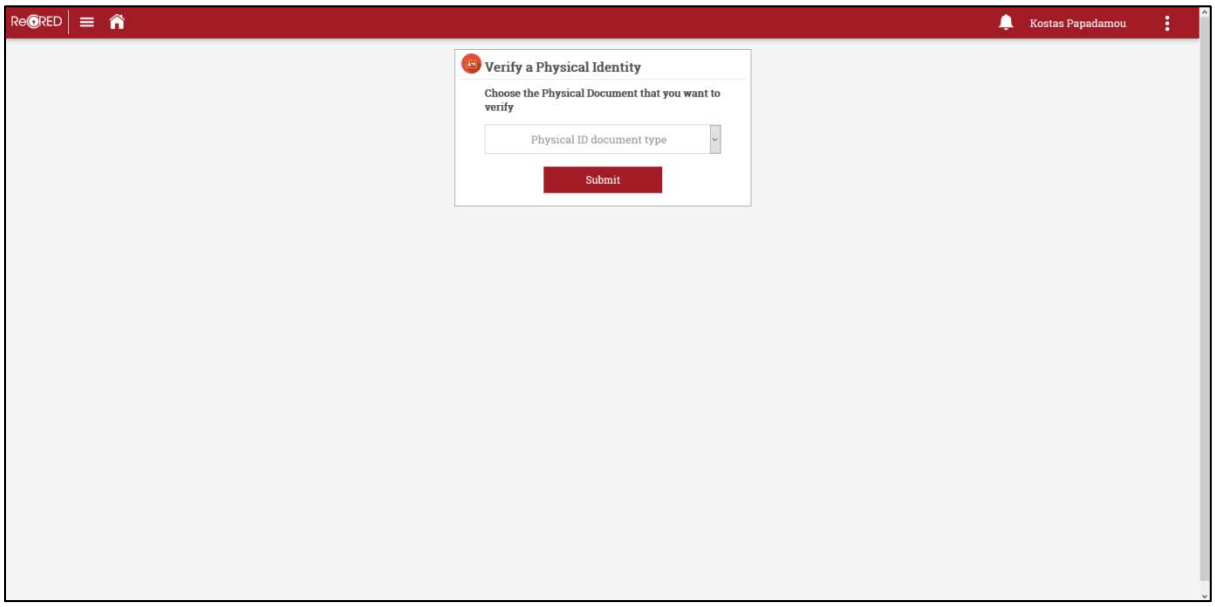

Figure 94. User chooses the Physical Identity document to verify screen

As soon as the user has chosen the identity document that he wants to verify then he is requested to declare the identity information that are included in the document. This screen in the web application is shown in figure 95 below.

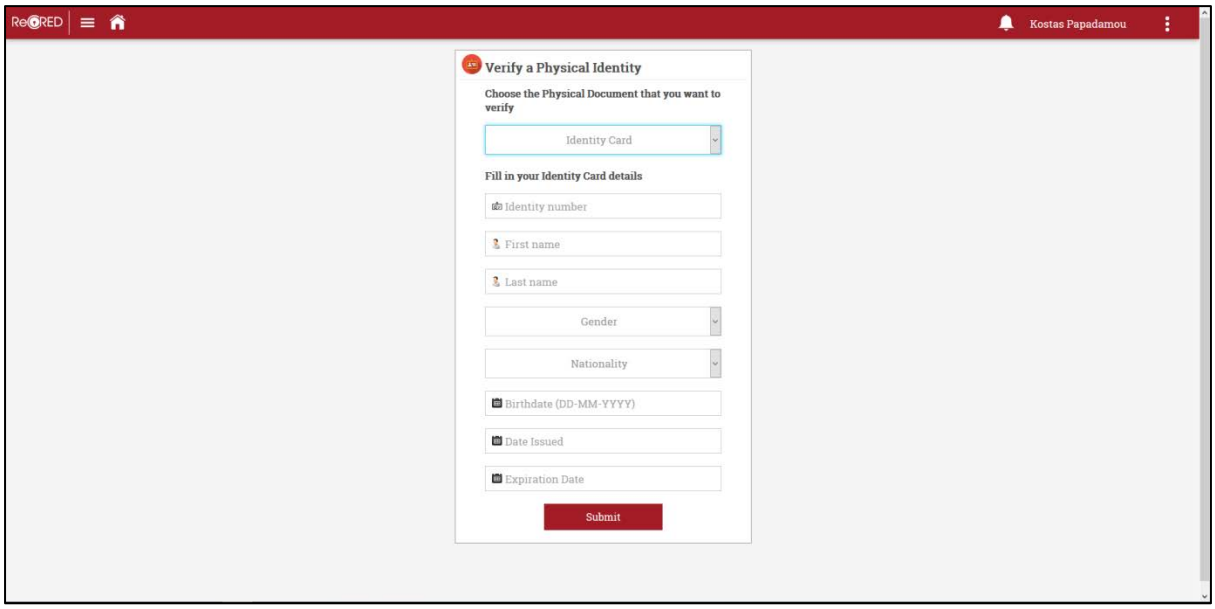

Figure 95. Declare identity document information screen

8.1.2 Capture Physical Identity Document and Face photos

The Physical Identity Acquisition module, in order to be able to generate the peer-to-peer audits and verify the declared identity information, it requests from the user to capture photos of his identity document and also of his face. Initially, the application requests from the user to capture a photo of his whole identity document.

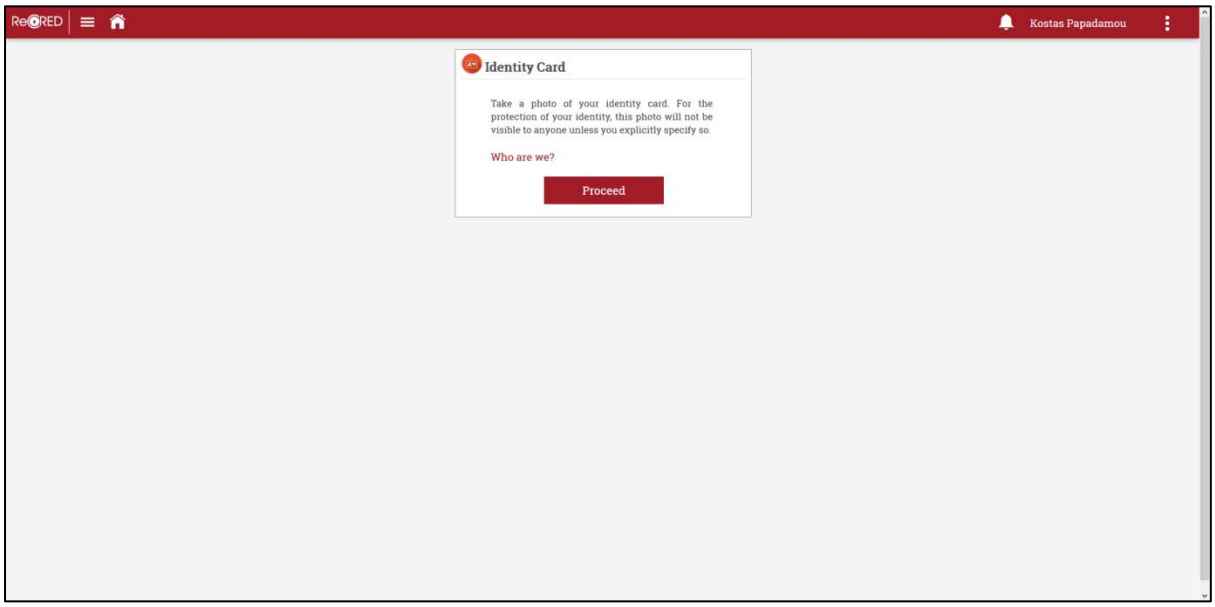

Figure 96. Notification to capture and upload a photo of the identity document

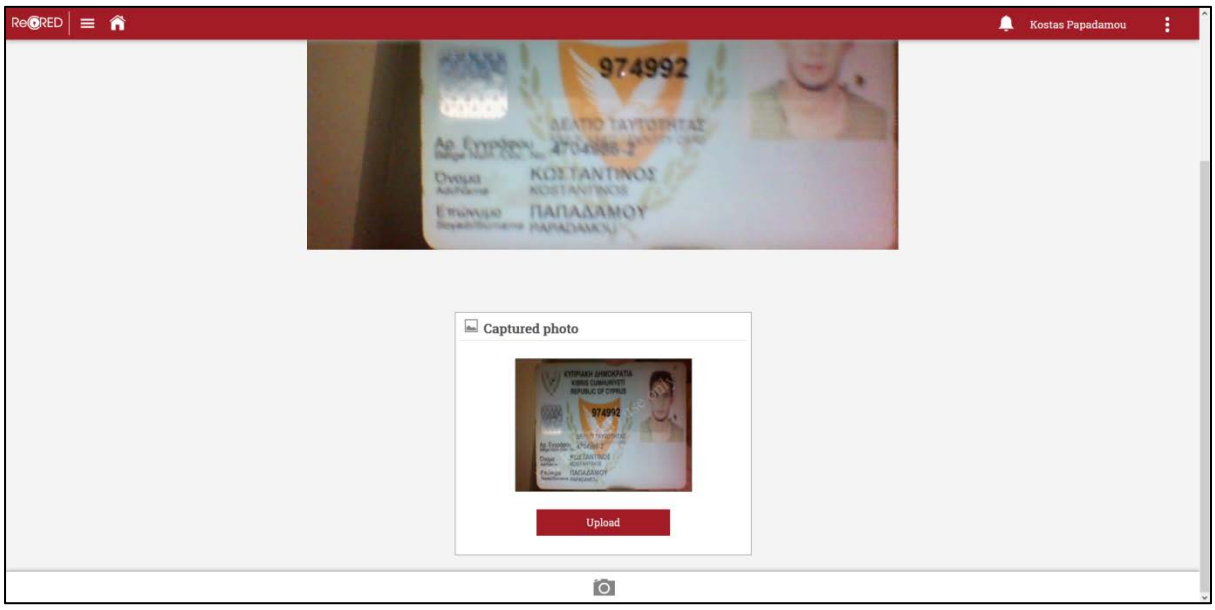

Figure 97. User captures a photo of his whole identity document

After capturing and uploading the photo of his whole identity document, the user is presented with a list of required actions requiring his to crop multiple parts from the captured identity document photo as shown in figure 98. In the case of the identity card these parts are the document number, the name and surname, the face photo included in the identity document and the date of birth. In the case of the passport the user is requested to crop all the aforementioned and the machine readable zone field included in the passport also.

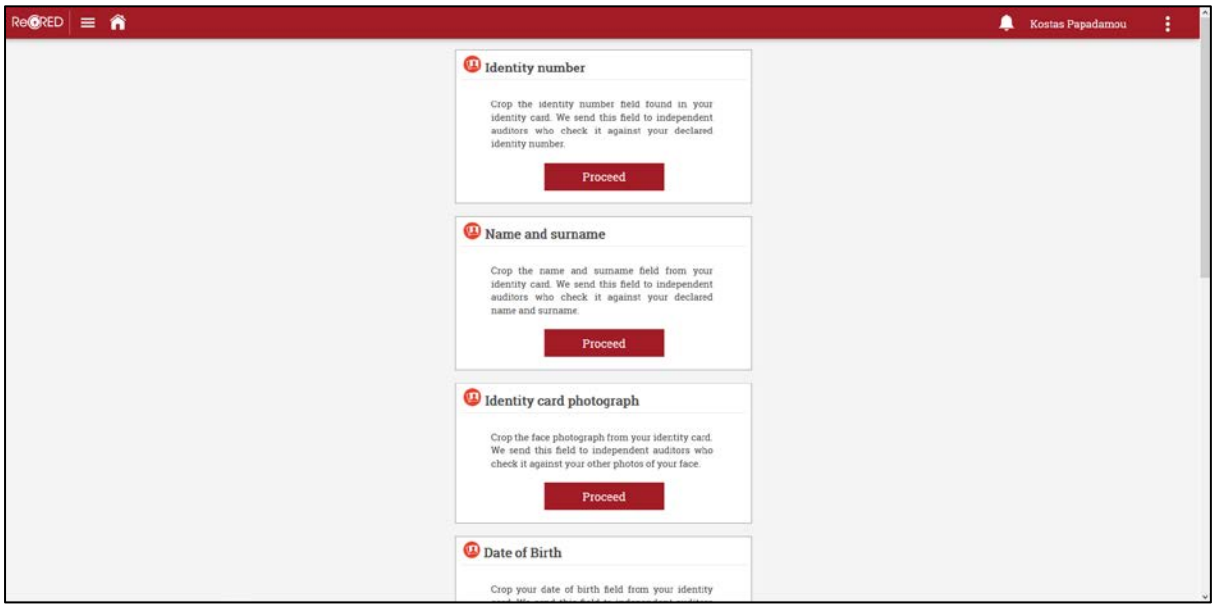

Figure 98. List of required parts to crop from the captured identity document photo

Here, it is important to mention that all the users’ captured photos are encrypted using AES 128-bit encryption and are securely stored in a private directory in the Identity Consolidator server. Additionally, when a physical identity document is successfully verifier all the related photos are deleted since the physical identity acquisition application does not need them anymore.

8.1.2.1 Cropping parts of the Identity document photo

As mentioned above, the user is requested to crop multiple parts of his whole identity document photo. Figure 99 below is an example of this functionality that shows the user cropping the part of the identity document photo that includes the identity number.

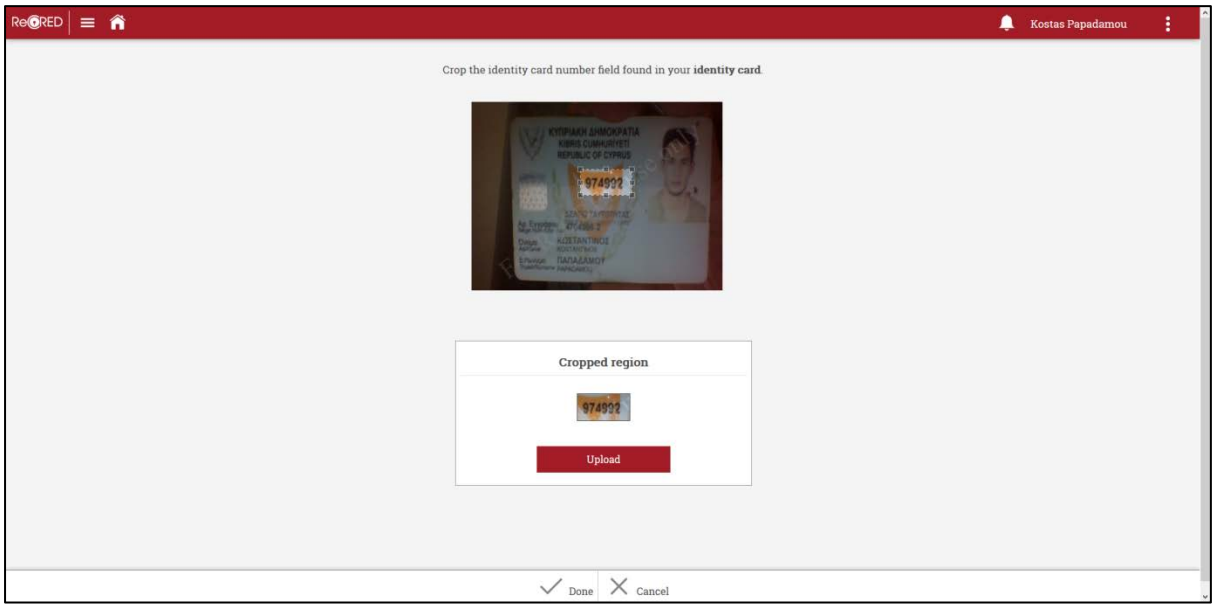

Figure 99. The user is cropping the part of the identity document photo that includes his identity number

We have implemented this functionality using Jcrop API [12], an open-source API for image cropping written in JavaScript. This API allowed us to add cropping functionality to our application

Below is a list of the parts that the user is requested to crop depending on the physical identity document that he wants to verify:

1. Identity Card
   a. Identity number
   b. Name and surname
   c. Identity photograph
   d. Date of birth
2. Passport
   a. Passport number
   b. Name and surname
   c. Passport photograph
   d. Machine readable zone
   e. Date of birth
3. Profession Certificate
   a. Profession
   b. Name and surname
4. Driving License
   a. Name and surname
   b. Driving license photograph
   c. Issue date
   d. Expiration date

8.1.2.2 Capture face photos for verification purposes

Besides cropping parts from the captured identity photo the user is required to capture some more photos including a selfie for verification purposes that will be explained below in the peer-to-peer verification subsection. He is requested a photo holding his identity document, a selfie, a photo of the right side of his face and a photo of the left side of his face as show in figure 100 below. These photos are requested only when the user tries to verify his identity card or passport.

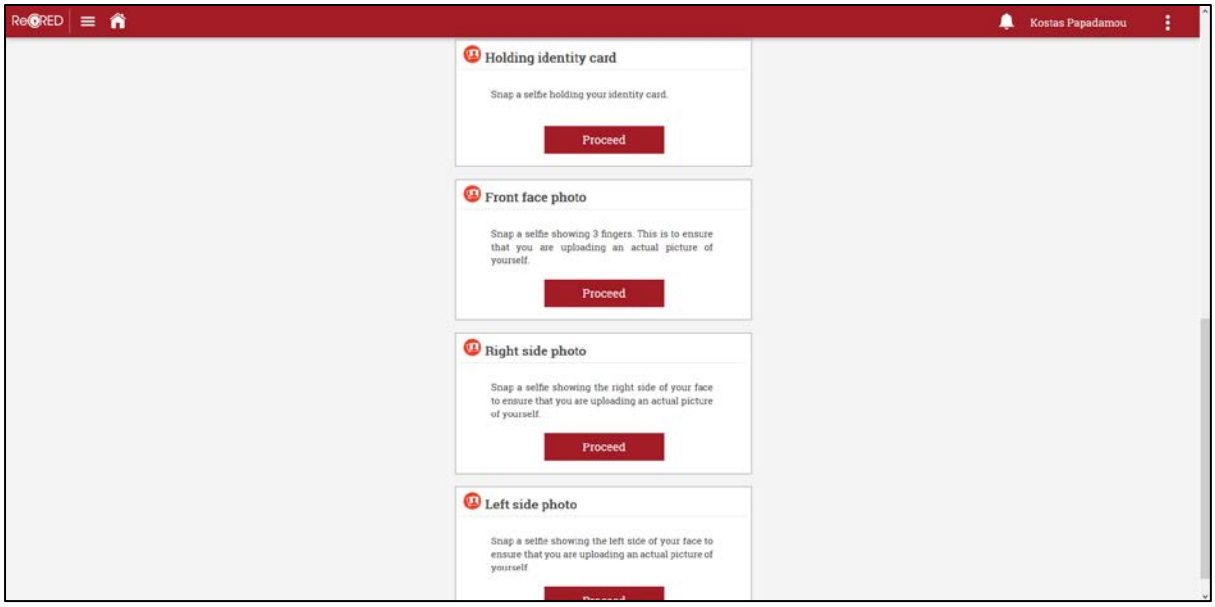

Figure 100. User is requested to capture face photos for verification purposes

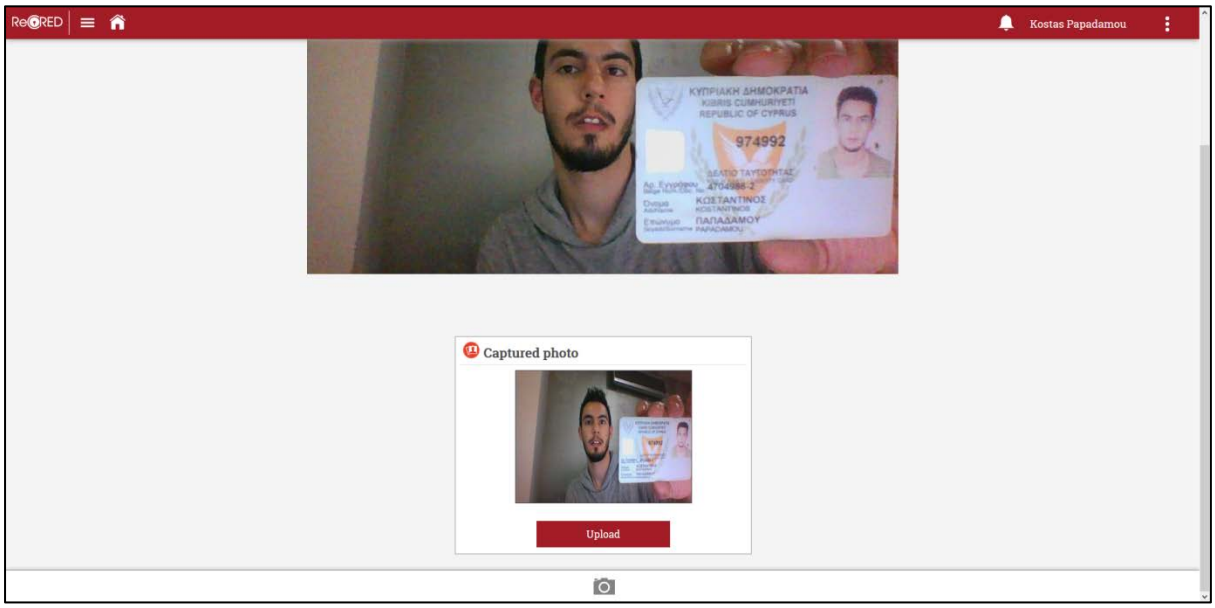

Figure 101. User captures a photo holding his identity document

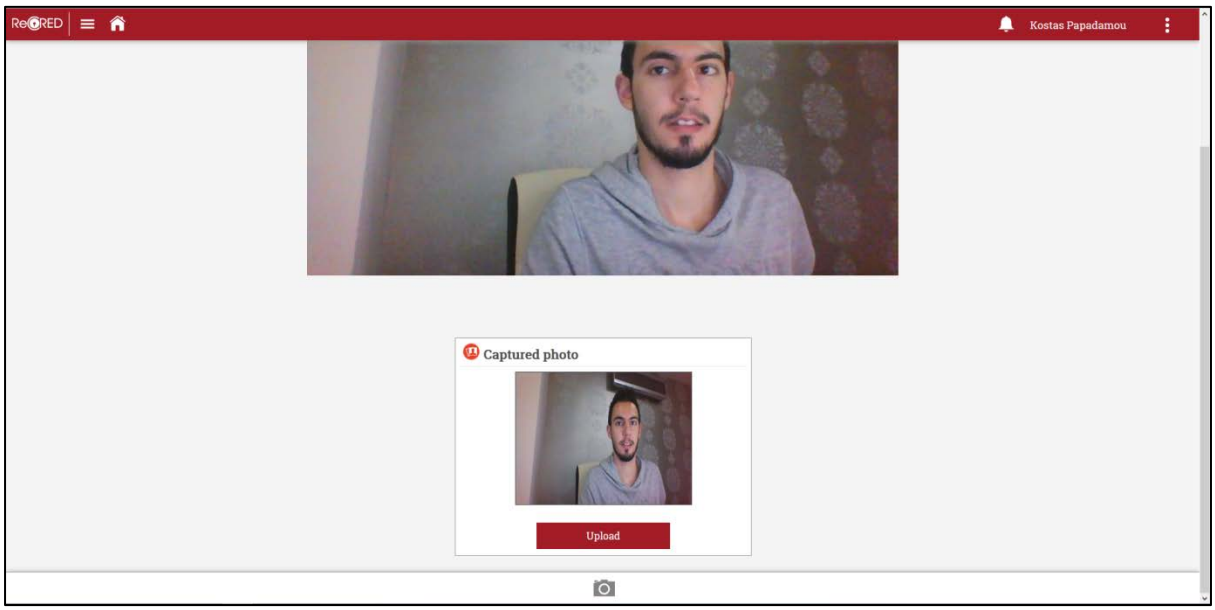

Figure 102. User captures a photo of his front face (selfie)

Figure 102 above shows the user capturing a photo of his front face. During this the application requests from the user to also perform a move (e.g., show two fingers) or captures the photo smiling and in a period of sixty seconds otherwise the page is refreshed. These actions are to ensure that the user captures the photo from his device’s camera.

In the back-end of the web application, as soon as the user has uploaded the required photos for each one of the peer-to-peer audits then the appropriate audits are issued and assigned to randomly selected users that exist in the system.

8.1.3 View/Manage identity information and uploaded pictures

The Physical Identity Acquisition application that we have developed offers to the users the ability to view and manage the pictures that they have uploaded as well as view their identity information.

With such functionality, the user is able to see all the photos that he has captured for each identity document separately. At the same time he is able to delete any photo that he prefers to capture and upload a new one because for example a photo is blurred. Furthermore, when a photo is deleted all the other photos that are related with this photo, in terms of peer-to-peer audits, are also deleted. If the user chooses to delete an identity document photo then all the related, to this identity document, photos are also deleted. Figure 103 below shows the screen in the application where the user can view the photos that he has uploaded and delete any photo if wants to.

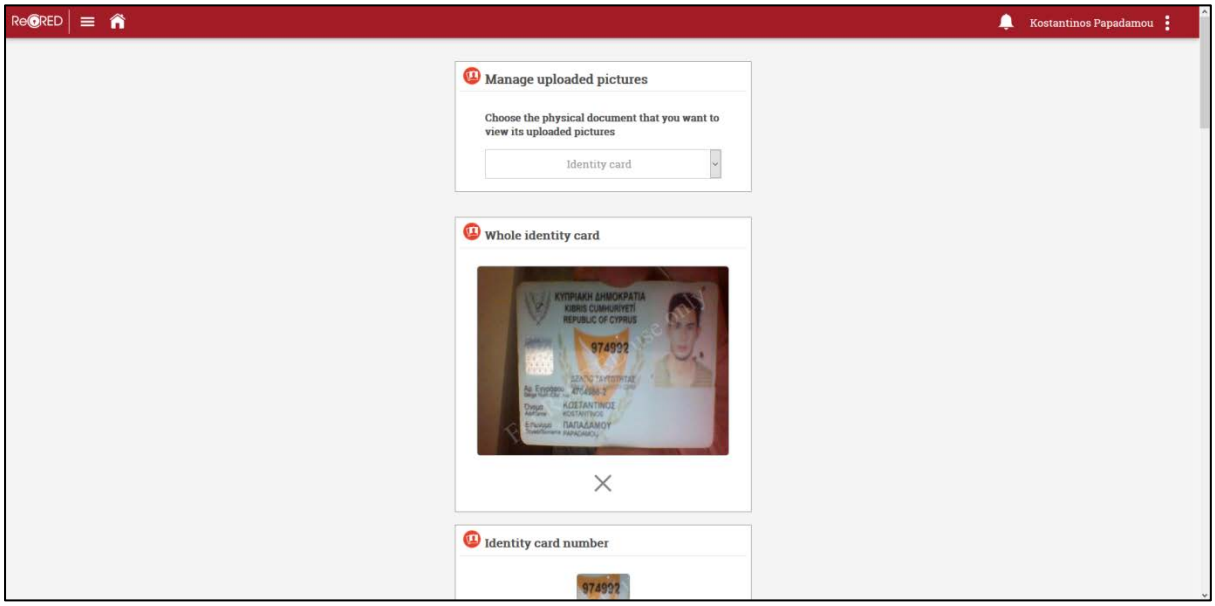

Figure 103. View/manage identity uploaded document photos screen

8.1.4 Address acquisition

Besides, identity document acquisition and verification, the developed Physical Identity Acquisition application allows users to declare and verify their address details for both their work and home address. Using this functionality a user has two options to declare and verify his addresses. The first is by clicking on a map where the address that he wants to declare is and the second option is to declare the details of the address while verifying on a map that the address that he wanted to declare is the correct one. Figure 104 below shows the screen where the user is able to choose the method that he prefers to use in order to declare and verify his address.

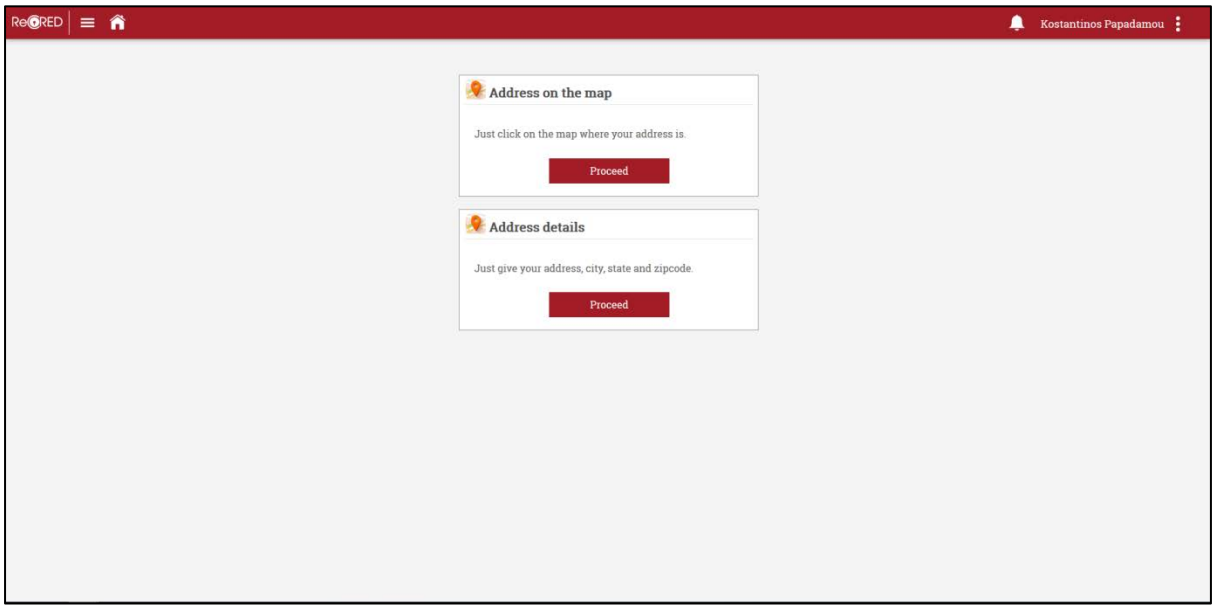

Figure 104. Methods that the user has to declare his addresses

For the implementation of this functionality we used the Google Maps JavaScript API [13], which allowed us to embed a map into this functionality. In the first option where the user is able to declare his addresses by clicking on a map, at the time when he clicks on a specific location on the map, in the background we use the Google Maps API to get the address details (street, county/state, city, country and postal code) as well as the coordinates of the selected location. In the second option we use the Google Maps API to get the coordinates of the declared address and also to show to the user the exact location on a map based on the declared address details. Both the two options that are offered to the users to declare their addresses are shown in figures X and X below.

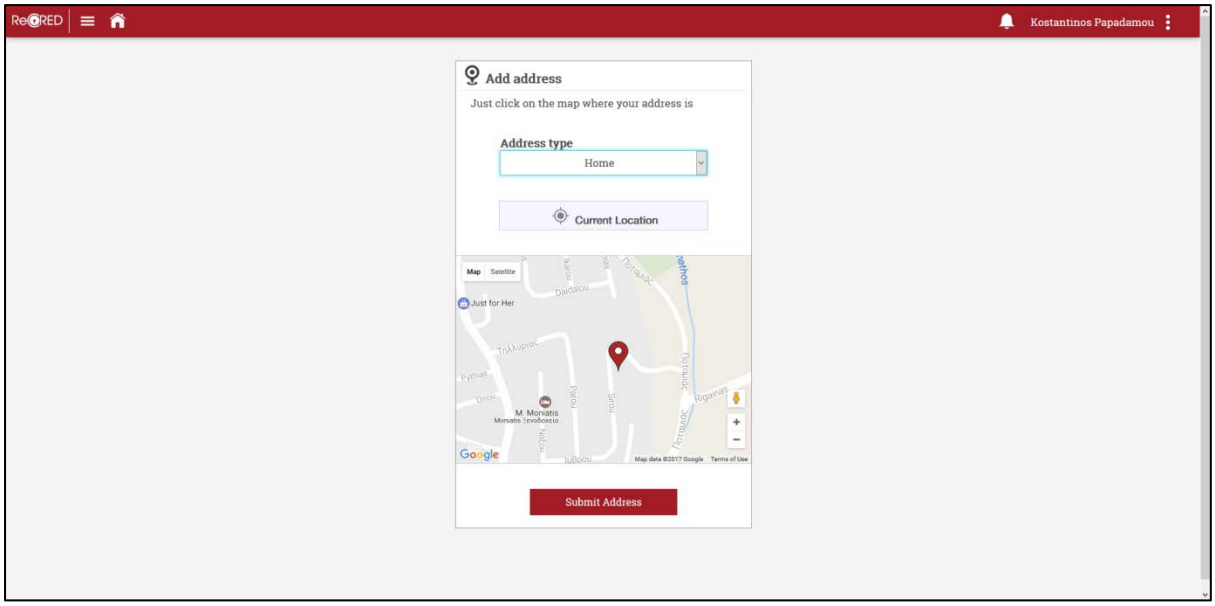

Figure 105. Address acquisition by clicking on a map

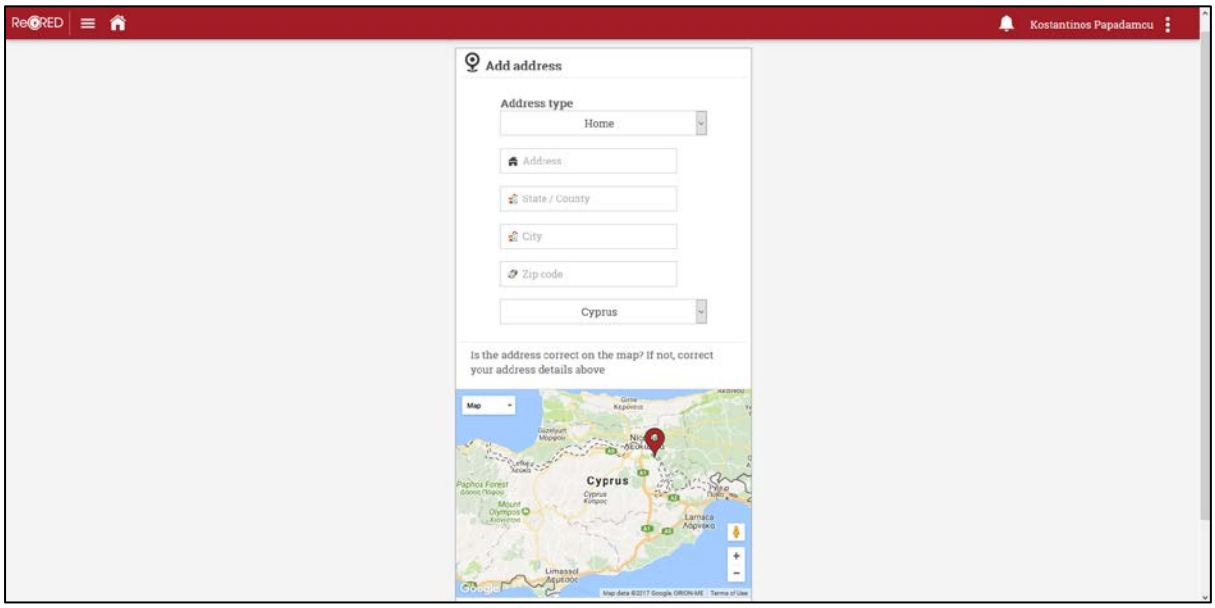

Figure 106. Address acquisition by declaring its details

The methods that are used to verify the declared users' addresses will be described in the physical identity verification section.

8.1.5 Real-time video presentation of Identity documents using WebRTC

In the context of the Physical Identity Acquisition web application we have developed another functionality that enables users to prove his identity to other users (verifiers) in the web through real-time video and audio presentation of his Identity document. This functionality has been implemented using WebRTC. In order for the communication to be initiated between the two users, the verifier should also have an account to the Identity Consolidator and has already proven his identity.

For the implementation of this functionality we made use of PeerJS library [14], which is a JavaScript library that wraps the browser's WebRTC implementation and provides a complete and easy-to-use peer-to-peer connection API. In order to initiate the real-time identity presentation, the only thing that a user has to do is to enter the email of the verifier and then the appropriate user is contacted to setup the connection between the two peers. Below there are some screenshots of this functionality.

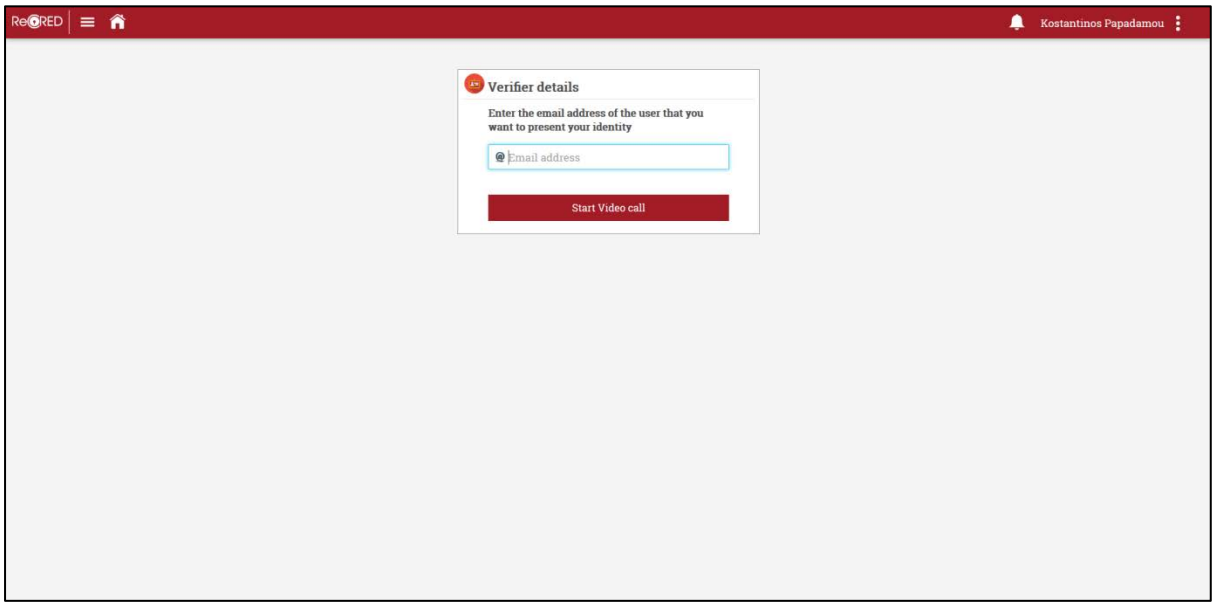

Figure 107. Screen where the user declares the email address of the real-time video presentation verifier

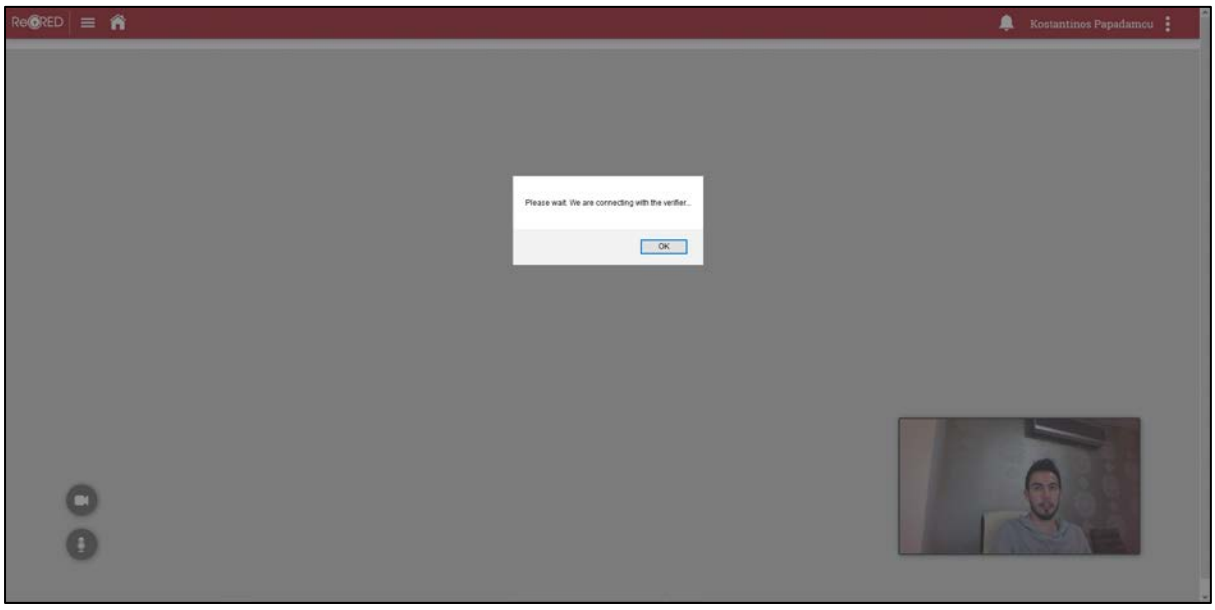

Figure 108. Real-time identity presentation using WebRTC

8.1.5.1 WebRTC (Web Real-Time Communication)

WebRTC is a collection of communication protocols and APIs that enables real-time communication, including video and audio, over peer-to-peer connections. It is an open project that enhances browsers, mobile applications and IoT devices with real-time communication capabilities.

8.1.6 RFID ID document scanning using NFC-enabled devices

Besides the standard identity document acquisition and verification process that the Physical Identity Acquisition module offers, it also offers to the users the ability to scan and directly verify their identity documentation using NFC-enabled mobile devices. This functionality is based on the findings of the IRMA project [15].

Taking advantage of the NFC technology this functionality enables the physical identity acquisition application (NFC reader) to communicate with the RFID-enabled identity card or passport and securely read all the identity attributes included in the identity document.

In order the physical identity acquisition application to be able to retrieve the identity information from the identity document, the device needs to perform BAC (Basic Access Control) against the identity document. Basic access control [16] is a mechanism used to ensure that only authorized parties can wirelessly read personal information included in identity documents with an RFID chip. Active Authentication (AA) is also performed to detect and prevent a cloned/forged ePassport to be read as an authentic one. In our implementation we are performing Active Authentication in a slightly different way than the other standard ePassport readers. In our case we are performing remote Active Authentication instead of local because we want the server to be able to verify whether the data read from an RFID chip of an ePassport belongs to the genuine document. The requirements for remote ePassport verification are much more than the standard local ePassport verification. In order to achieve this we are performing Active Authentication between the server and the ePassport (the server sends the challenge) with the user device (ePassport reader) being just a proxy that scans the data from the ePassport.

For the implementation of this functionality we make use of JMRTD library [17]. JMRTD is an open-source Java library that implements the Machine Readable Travel Document (MRTD) standards as specified by the International Civil Aviation Organization (ICAO) [18]. Part of this implementation is a host side Java API, which is the one that we actually use in order to read, decode, and validate the information included in the RFID chip of the identity document.

As soon as the Physical Identity Acquisition application had successfully acquired all the identity attributes of the user included in his identity document, it uses the Storage API to securely store them in the Identity Repository.

Below there are there are the screenshots of the implemented functionality. We have implemented this functionality for both electronic passports and identities. Figure 109 shows the first page of the process that a user has to follow in order to verify his biometric ID document where we explain to him the steps that he needs to follow. Next, as shown in figure 110 the user is requested to choose the ID document type that he wants to scan (ePassport or eID).

After the user has chosen the document that he wants to scan we request from him to enter some basic information (Document number, Expiration date and date of birth) in order to perform Basic access control and be able to read the data stored on the RFID chip of the document as shown in figure 111. The next thing that the user has to do is to place his mobile device on top of his ID document in order to communicate with the installed RFID chip. After that, BAC is performed and if the provided information is correct then we perform AA. If AA is also successful then we are able to read the data stored on the document.

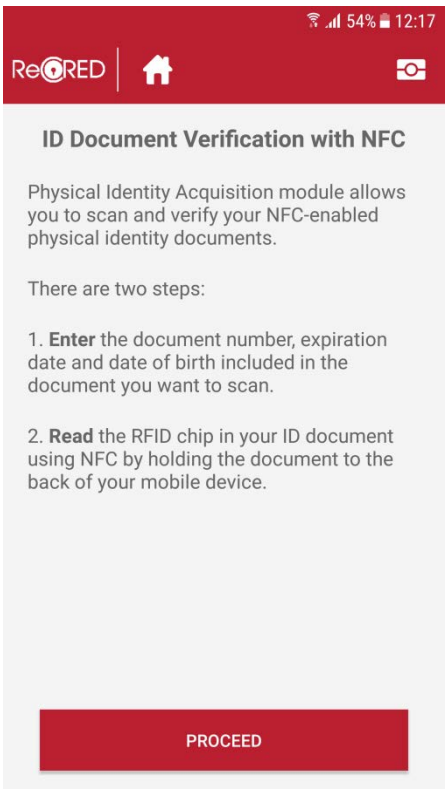

Figure 109. NFC ID Document verification guidelines

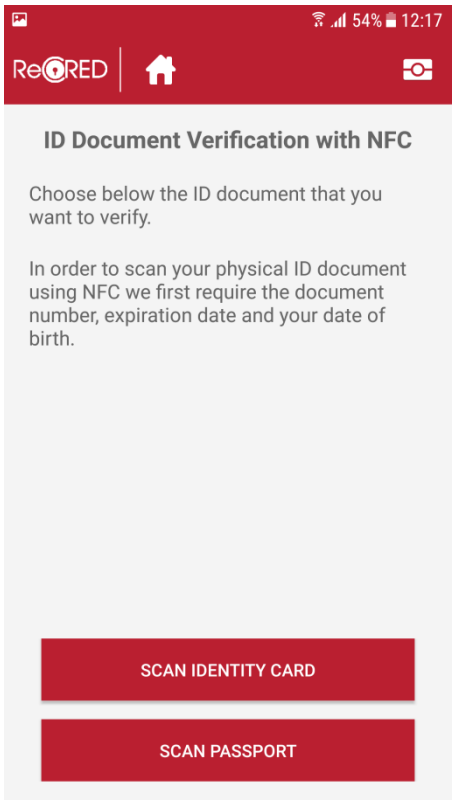

Figure 110. Choose the document (ePassport or eID) to verify using NFC

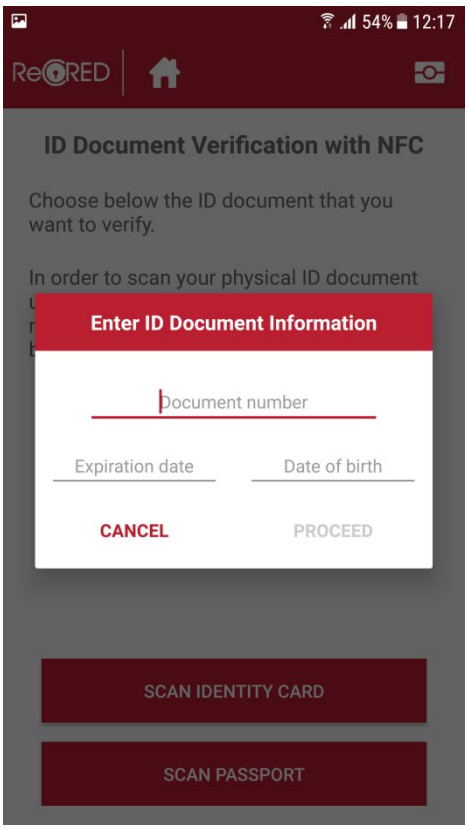

Figure 111. Enter ID document required information for Basic Access Control authentication

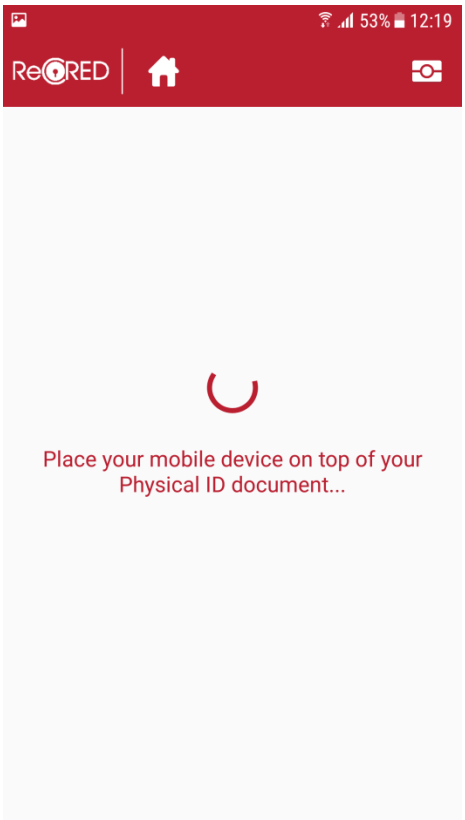

Figure 112. User is requested to place his mobile device on top of his ID document to be scanned

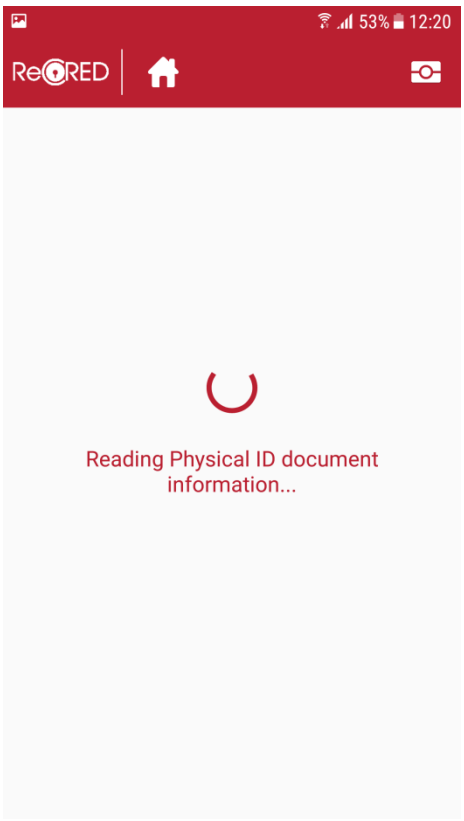

Figure 113. If BAC and AA are successful the data stored on the ID document are read

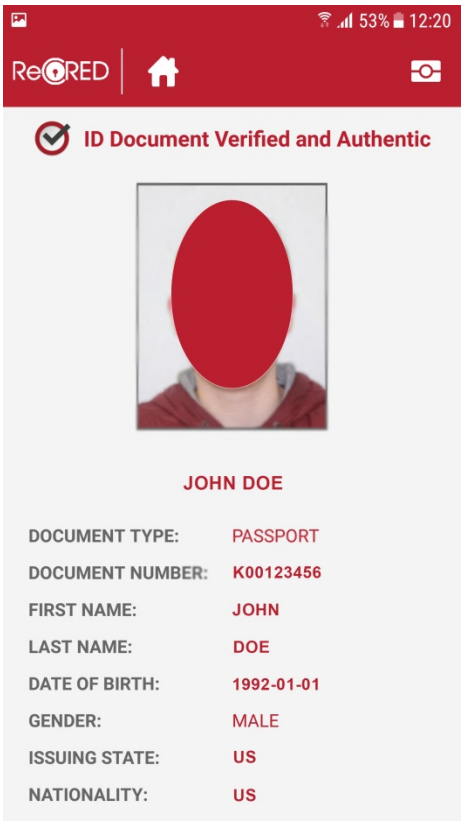

Figure 114. Data from the ID Document has been read successfully. Data has been anonymized for privacy reasons

8.1.6.1 NFC (Near Field Communication)

NFC technology is a form of secure contactless communication between smart devices. Such contactless communications allows users to wave the smart device over a NFC compatible device or card to send or receive information without the need to touch the devices together and with no extra steps for setting up the connection.

NFC technology allows a device, known as a reader or active device, to create a radio frequency current that communicates with another NFC compatible device or a small NFC tag holding the information the reader wants. Passive devices, such as the NFC tag in smart posters, store information and communicate with the reader but do not actively read other devices. Peer-to-peer communication through two active devices is also a possibility with NFC. This allows both devices to send and receive information.

8.1.6.2 Biometric Passport (ePassport)

Biometric passports are also known as ePassport or digital passports. It is an improved paper and electronic passport which contains all the personal information of a person as well as his biometric information (e.g., fingerprint) and can be used to authenticate his identity. ePassports use contactless smart card technology by having an embedded RFID chip. All the document and chip characteristics are documented in the International Civil Aviation Organization's (ICAO) [37].

In order a reader to be able to scan the passport and read the data stored electronically in the passports RFID chip, it first need to authenticate against the document. Public Key Infrastructure (PKI) is used for the authentication and this makes it difficult for a malicious user to forge when all the proposed security measurements are forced.

8.1.6.3 Security Mechanisms

Biometric passports are equipped with various security mechanisms in order to defend against various types of attacks. More precisely, those mechanisms are:

1. Non-traceable chip characteristics: When a reader tries to communicate and scan the RFID chip of an ePassport, the chip each time replies with a different chip number. This prevents malicious users from tracing ePassport chips.
2. Basic Access Control (BAC): is used to protect the communication channel between the RFID chip and the reader. The reader application needs to provide some basic information derived from the Machine Readable Zone (MRZ) of the ID document such as date of birth, expiration date, document number, before the data stored in the RFID chip can be read.
3. Passive Authentication (PA): Passive authentication is used to detect modifications of the date stored on ePassports chips. The RFID chip contains a file (SOD) that stores hash values of all files stored in the chip (picture, fingerprint, etc.) and a digital signature of these hashes. The digital signature is made using a document signing key which itself is signed by a country signing key. If a file in the chip (e.g. the picture) is changed, this can be detected since the hash value is incorrect. Readers need access to all used public country keys to check whether the digital signature is generated by a trusted country.
4. Active Authentication (AA): Active Authentication is used to prevent cloning of ePassport RFID chips. The chip contains a private key that cannot be read or copied, but its existence

can easily be proven. AA is based on a challenge response mechanism and allows the reader to verify that the data scanned from an RFID chip belongs the genuine ePassport. In our implementation AA is performed between the server and the ePassport because we want to perform remote NFC ID document scanning.

5. Extended Access Control (EAC): EAC offers the required functionality for checking the authenticity of both the chip and the reader. Furthermore, it uses stronger encryption than BAC and is typically used to protect the biometric information (e.g., fingerprint, iris, etc.) stored in the RFID chips. All the European Union’s ePassports use EAC.
6. Supplemental Access Control (SAC): This access control scheme has been introduced later on for address BAC weaknesses.
7. Shielding the RFID chip: By shielding the RFID chip of the ePassport we can prevent unauthorized reading of the document. Some countries have integrated a very thin metal mesh into the passport's cover to act as a shield when the passport cover is closed.

8.2 Physical Identity Verification

The Physical Identity Verification process includes all the methods used for the verification of all the acquired identity information of the users. Among the methods used for verification are peer-to-peer audits using crowdsourcing techniques, and automatic verification methods using OCR, face detection, and face recognition technologies. We have implemented most of these methods as part of the back-end of the Physical identity Acquisition module.

We have also implemented location tracking functionality for the verification of the declared users’ addresses. An additional identity document verification method is also offered to the users by the Physical Identity Acquisition module, which includes the face-to-face verification of identity documents by trained auditors.

8.2.1 Peer-to-Peer Verification using crowdsourcing techniques

As mentioned in the Physical identity Acquisition subsection, when a user has captured and uploaded the required photos for each identity document, in the back-end of the Physical Identity Acquisition module we assign peer-to-peer audits to randomly selected users. These audits are crowdsourced to other users in order to verify that the information included in the acquired photos of another user match his declared personal identity information. Watermarking and cropping techniques are used to ensure that nobody will have access to reusable or forgeable photos of the users’ physical identity documents. Additionally, we implemented this functionality so that no more than one audit is crowdsourced to each auditor in order to prevent them to have access to more than one identity document photos of the audited user.

For each identity document that is supported by the Physical Identity Acquisition module we have implemented different audit types where each audit verifies a different identity attribute of the user. These types are:

1. Identity Card
   a. Identity number audit: the auditor is requested to verify whether the identity number shown in the cropped identity card photo matches the identity number that the audited user declared.

b. Name and surname audit: the auditor is requested to verify whether the name and surname shown in the cropped identity card photo matches the name and surname that the audited user declared.
c. Holding identity card audit: the auditor is requested to verify whether the face of the user in the captured photo matches the identity card face photograph that the user holds in his hand.
d. Redacted identity card audit: the auditor is requested to verify whether the redacted identity document in the photo is an identity card issued by the country that the audited user declared.
e. Identity card user photograph audit: the auditor is requested to verify that all the photos include the same person and whether this person is performing a requested move in his front face photo.
f. Date of birth audit: the auditor is requested to verify whether the date of birth shown in the cropped identity card photo matches the date of birth that the audited user declared.

2. Passport
   a. Passport number audit: the auditor is requested to verify whether the passport number shown in the cropped passport photo matches the passport number that the audited user declared.
   b. Name and surname audit: the auditor is requested to verify whether the name and surname shown in the cropped passport photo matches the name and surname that the audited user declared.
   c. Holding passport audit: the auditor is requested to verify whether the face of the user in the captured photo matches the passport face photograph that the user holds in his hand.
   d. Redacted passport audit: the auditor is requested to verify whether the redacted passport in the photo is a passport issued by the country that the audited user declared.
   e. Identity card user photograph audit: the auditor is requested to verify that all the photos include the same person and whether this person is performing a requested move in his front face photo.
   f. Date of birth audit: the auditor is requested to verify whether the date of birth shown in the cropped passport photo matches the date of birth that the audited user declared.
3. Profession Certificate
   a. Name and surname audit: the auditor is requested to verify whether the name and surname shown in the cropped profession certificate photo matches the name and surname that the audited user declared.
   b. Redacted profession certificate audit: the auditor is requested to verify whether the profession information shown in the redacted profession certificate photo matches the profession that the audited user declared.
4. Driving License
   a. Name and surname audit: the auditor is requested to verify whether the name and surname shown in the cropped driving license photo matches the name and surname that the audited user declared.
   b. Driving license user photograph audit: the auditor is requested to verify that all the photos include the same person and whether this person is performing a requested move in his front face photo.

c. Driving license issue date audit: the auditor is requested to verify whether the date shown in the cropped driving license photo matches the driving license issue date that the audited user declared.
d. Driving license expiration date audit: the auditor is requested to verify whether the date shown on the cropped driving license photo matches the driving license expiration date that the audited user declared.
e. Redacted driving license audit: the auditor is requested to verify whether the redacted driving license in the photo is a driving license issued by the country that the audited user declared.

When all the audits of a user has been successfully performed by the responsible auditors and if all of them successful then the specific identity document is considered as verified and all the identity attributes related to this identity document are independently stored as verified into the Identity Repository. Examples of some of the implemented crowdsourced audits are shown in the figures below.

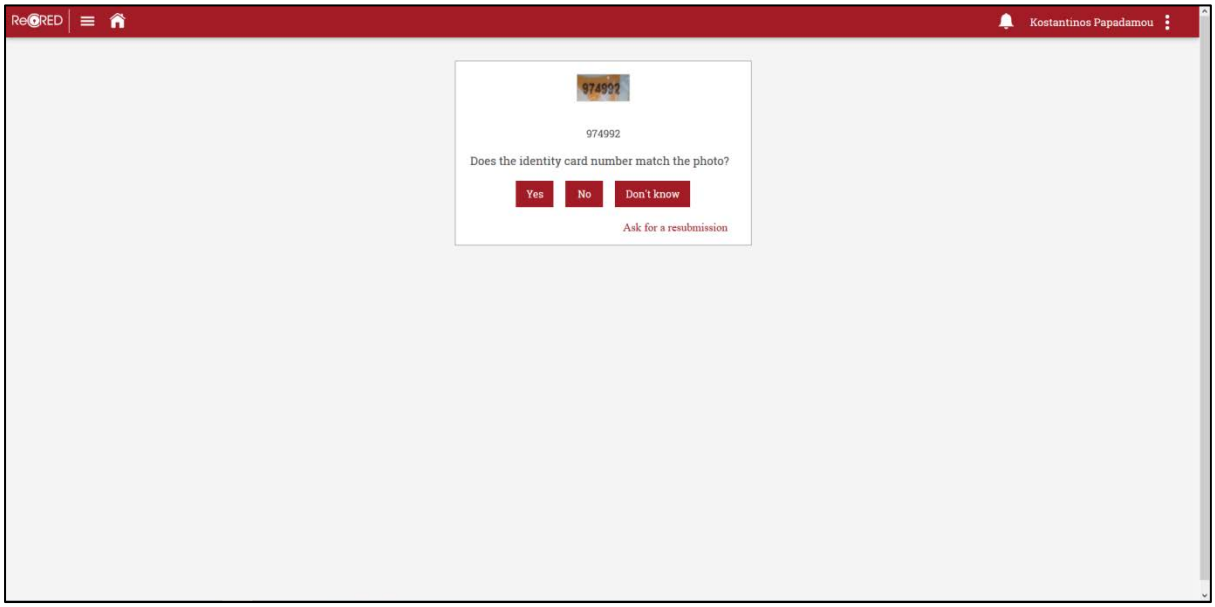

Figure 115. Identity number audit example

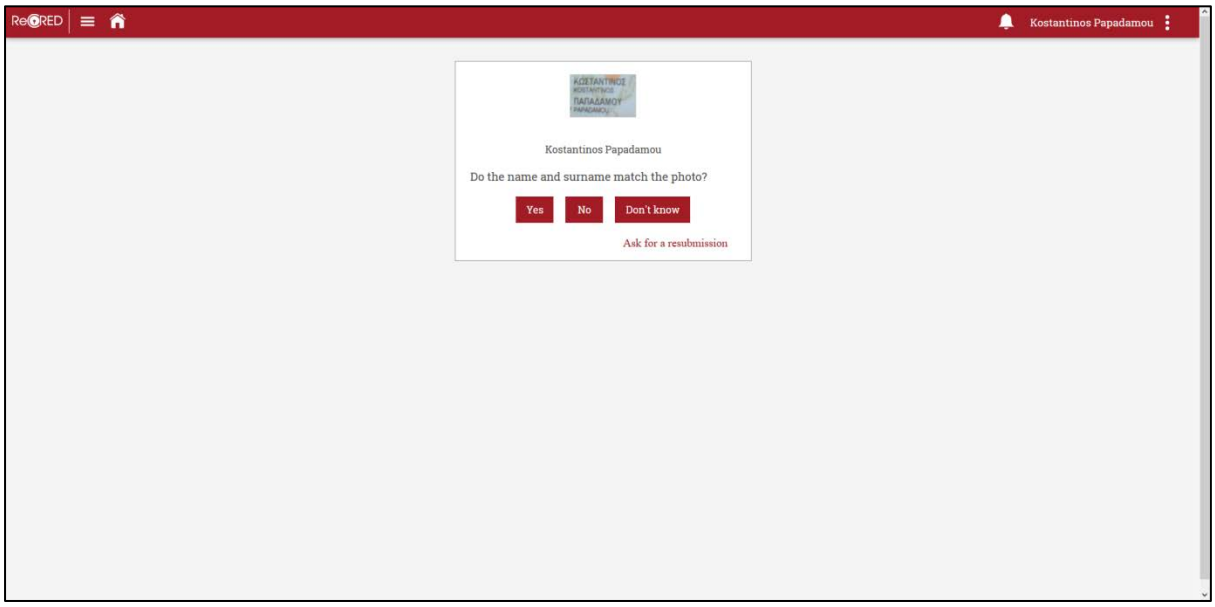

Figure 116. Name and surname audit example

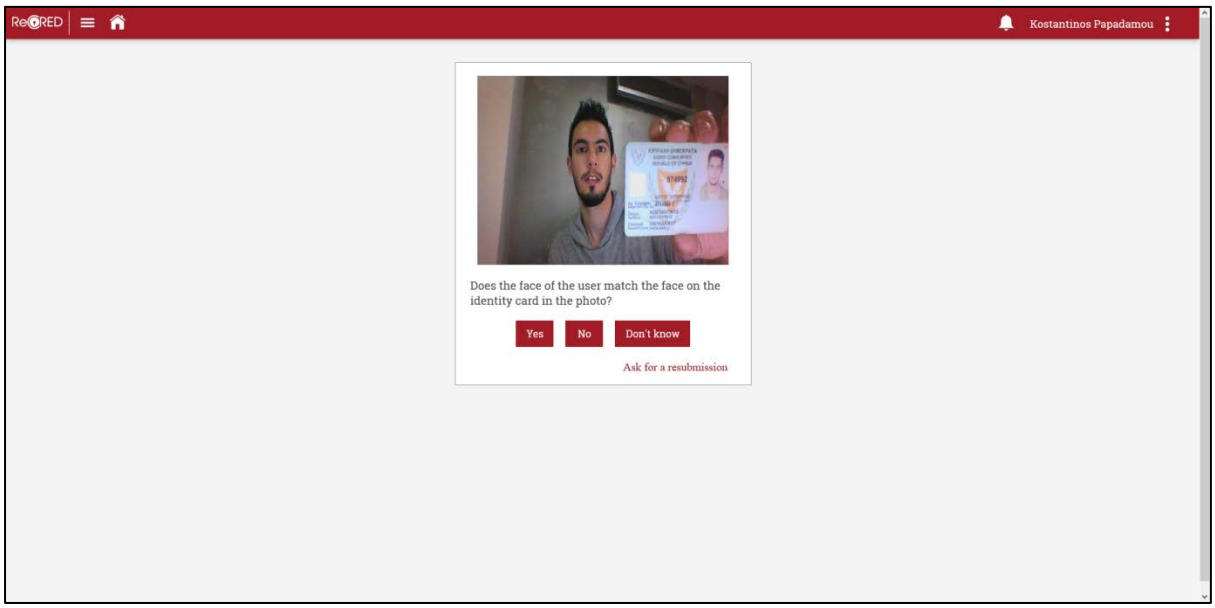

Figure 117. Holding identity document audit example

8.2.2 Automated verification methods

For the verification of the acquired identity information and uploaded photos of the user, in addition to the peer-to-peer verification the Physical Identity Acquisition module uses existing techniques like OCR, face detection etc. We employ such techniques in various parts of the identity acquisition and verification process in order to automatically verify the acquired identity information and pictures of the users.

More precisely, we employ the following techniques:

a. Optical Character Recognition (OCR) [21]: is the mechanical or electronic conversion of images of typed, handwritten or printed text into machine-encoded text. We employed this technique when a user crops and uploads parts of his identity document photo in order to

first detect that a text is included in each of those images and to also extract this text. In the background of our application we extract the text from the images and we then compare it against the appropriate declare identity information to verify that they match. We have implemented this functionality using the phpOCR library [22], which is an open-source OCR library written in PHP that allowed us to extract text from the images that users upload.

b. Face detection techniques: Face detection or facial detection [23] is a computer technology used in a variety of applications that enables the detection of human faces in digital images. In the Physical Identity Acquisition module we employ face detection to determine and verify whether a human face is included in the identity document photo and in all the photos that the user is required to capture photos of his face. For the implementation of this functionality we used OpenCV [24] and PHP Facedetect extension [25], which is an OpenCV wrapper implemented in PHP for face detection in images. We prefer to have automatic face detection in the back-end of the Physical Identity Acquisition application instead of using a cloud API because the images that are process contain users’ sensitive personal information that we want to protect.

c. Face recognition: Face recognition or facial recognition systems are capable of identifying or verifying that the person included in an image is the same person included in another image or a face database. We employ face recognition in our application to determine whether the person that is included in a user’s cropped identity photograph is the same person with the one included in the front face photo of the user. For the implementation of this functionality we use the Kairos SDK [26, 27], which is a PHP wrapper for the Kairos Face Recognition API. Kairos Face Recognition API provides a full-featured and robust Face Recognition back-end for all web applications that want to embed face recognition functionality.

8.2.3 View verification results

In order the verified user to be able to see the verification results of the crowdsourced audits for each of his identity documents, we have implemented a functionality that informs him of the positive and negative results for each audit. Figure 112 below shows the page of the web application that represents this functionality.

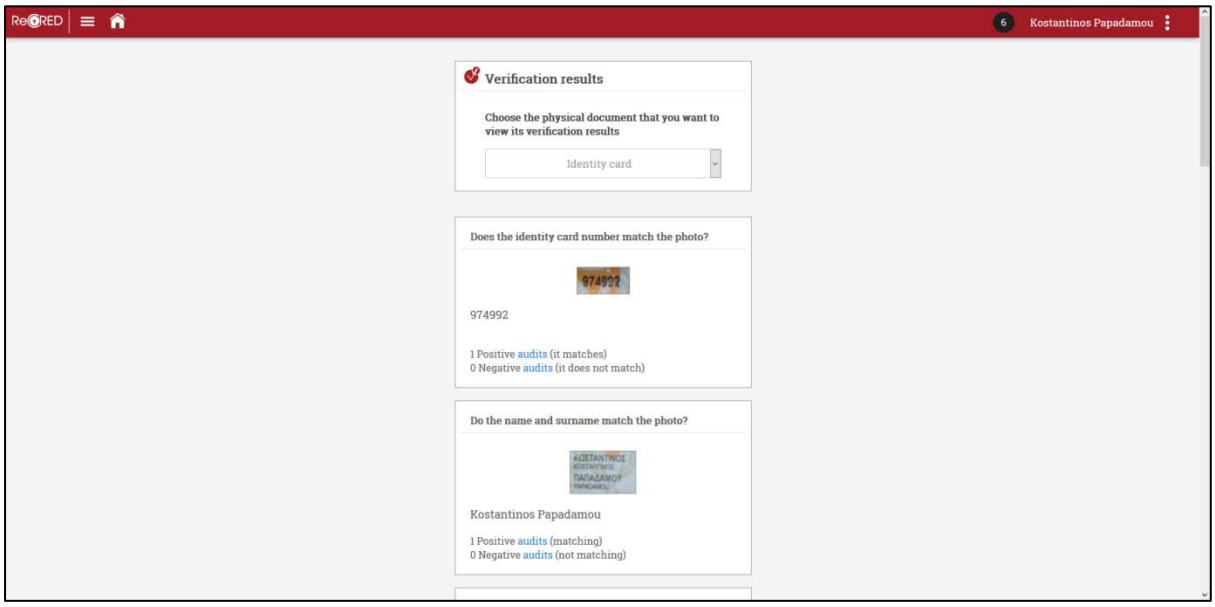

Figure 118. Peer-to-peer verification results page

8.2.4 Address Verification

In the identity acquisition section above we described the two methods that the Physical Identity Acquisition offers to the users to declare their addresses. In this subsection we describe the methods that the Physical Identity Acquisition module offers to the users in order to verify their declared addresses. These verification methods are:

1. Location tracking
2. Utility bill verification

Figure 113 below shows the two address verification methods that the Physical Identity Acquisition application offers.

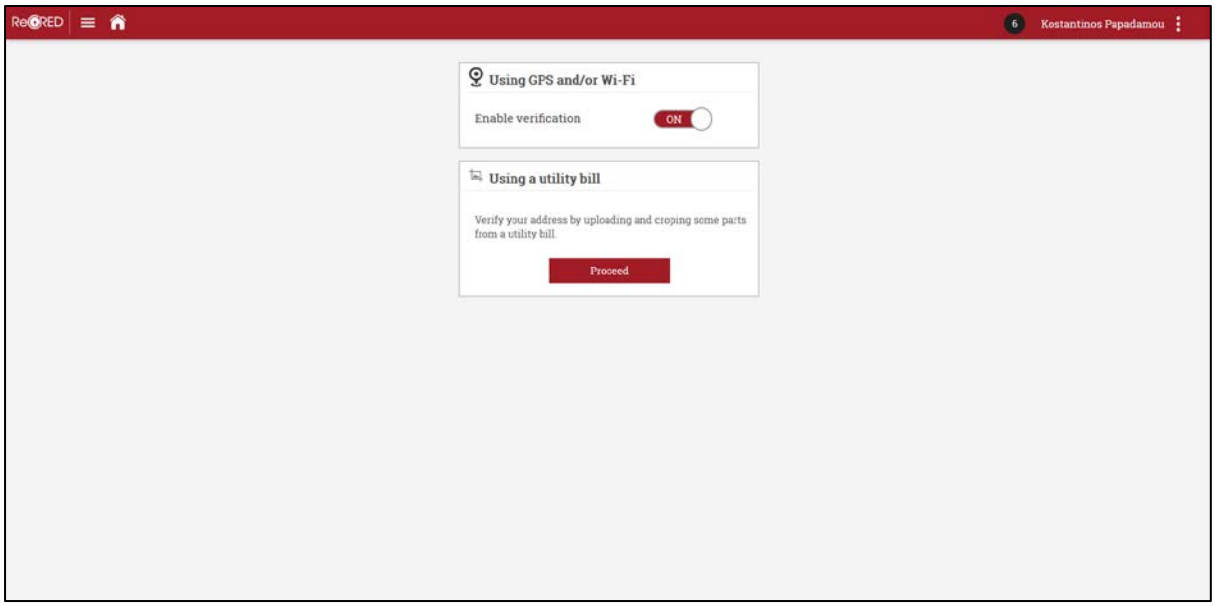

Figure 119. Physical Identity Acquisition. Address verification methods offered to the users

8.2.4.1 Address verification with location tracking

Location tracking is the one of the two methods offer to the users in order to verify their declared addresses. As soon as a user has declared one of his addresses, the location tracking option for address verification is available to be enabled.

When the user enables location, the physical identity acquisition application starts capturing his location for fifteen consecutive days. The application captures the coordinates of the user’s location from the HTML5 Geolocation API [28] functionality and the captured coordinates are asynchronously stored to the Identity Repository through the Storage API. When the location tracking 15-days period is passed, in the back-end of the Physical Identity Acquisition we calculate, for each declared address, a percentage that represents the number of the captured location’s coordinates that match with the addresses coordinates out of the total number of captured locations for the specific user.

8.2.4.2 Address verification using utility bill

The second method that our application offers to the users to verify their declared address is using a utility bill. For this the user is initially requested to choose the address that he wants to verify and also declare company that issued the utility bill that he is willing to submit. Then he is requested to capture and upload a photo of a utility bill that contains his name, his surname and the details of the address that he wants to verify. This page is shown in the figure below.

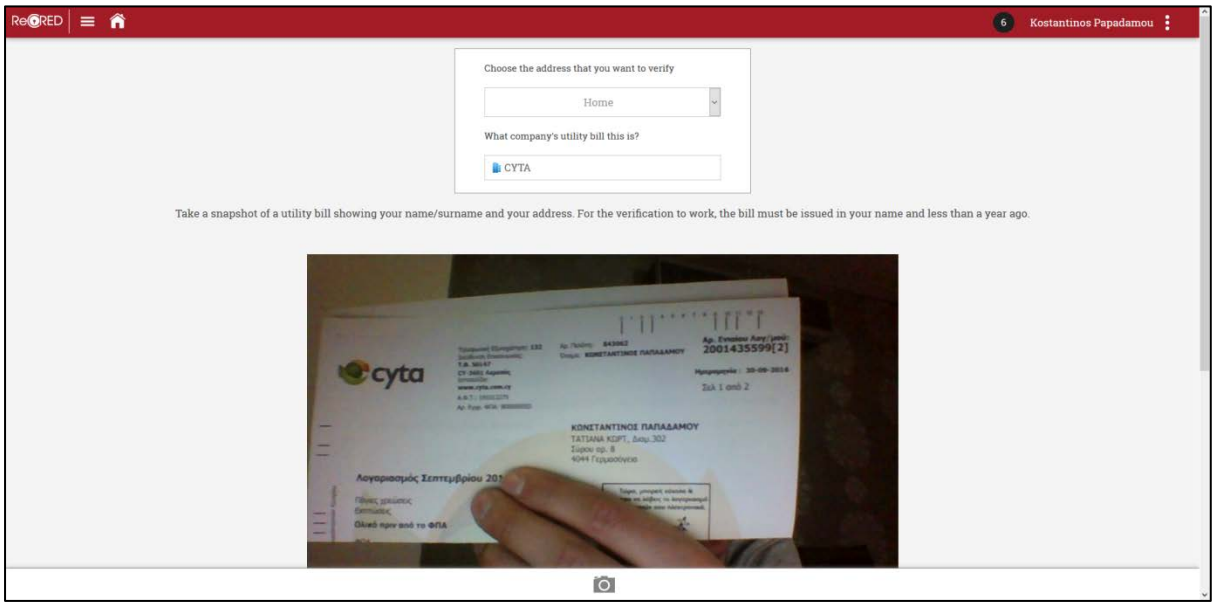

Figure 120. User captures a photo of his utility bill for address verification

After capturing and uploading a photo of his utility bill, the user is requested to crop various parts of this photo. These parts are his name and surname, the address details, the date that the utility bill was issued, and the logo and/or the name of the company that issued the utility bill. Figure 115 below shows an example of a user cropping his address details from his uploaded utility bill photo.

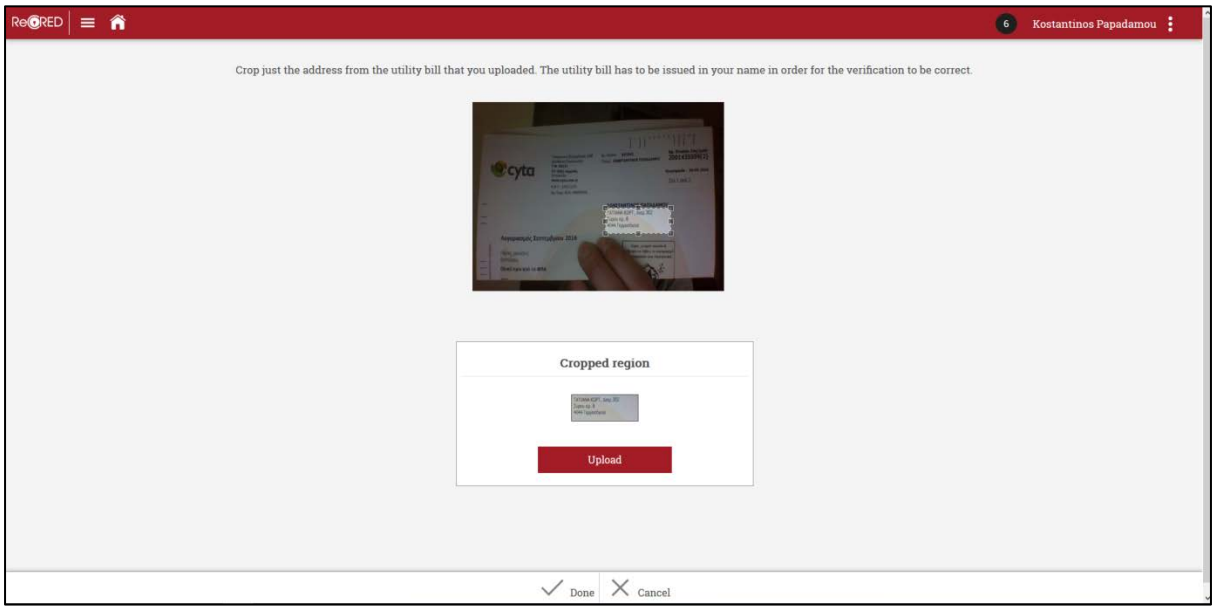

Figure 121. User is cropping the address details from his uploaded utility bill photo

For the verification of the utility bill related photos we have implemented crowdsourced audits, same as with the physical identity documents, and we assign them to randomly selected auditors when the user has uploaded all the required photos. Out of this audits we intent to verify the utility bill that the user has uploaded and also to verify whether the address details included in the uploaded utility matches the details of the address that the user declared and chooses to verify with the utility bill. An example of this audit is shown in the figure below.

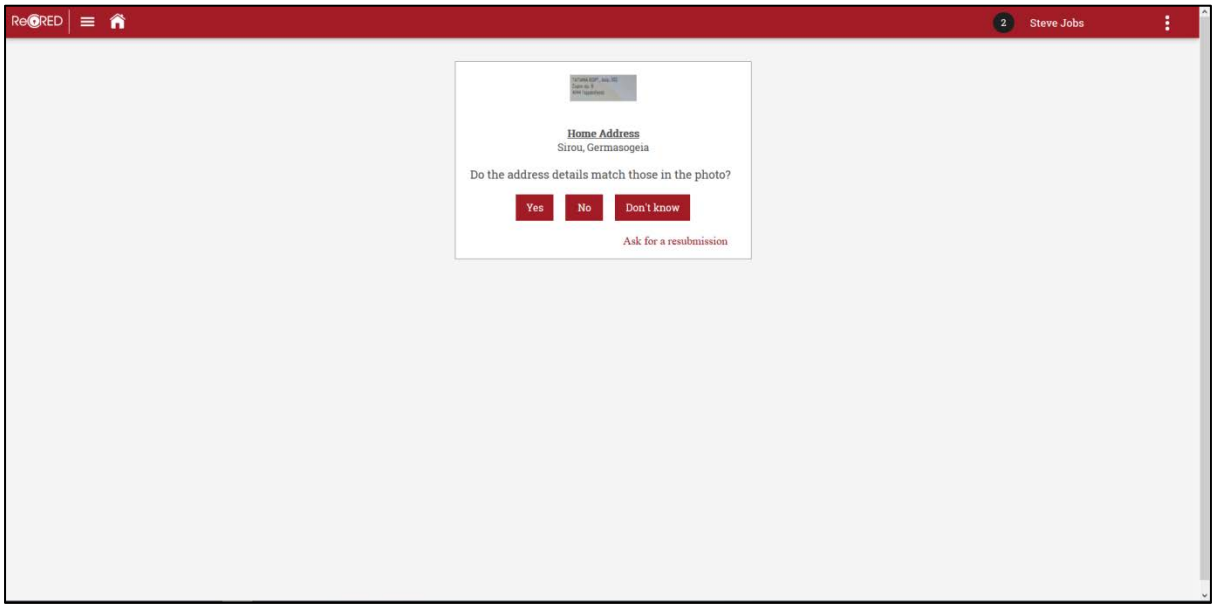

Figure 122. Peer-to-peer audit that compares the address details on the utility bill with the declared address details

8.2.5 Face-to-Face Identity Document verification by trained auditors

The last identity document verification method that we have implemented as part of the Physical Identity Acquisition module is the face-to-face (f2f) identity document verification by trained

auditors. This method is an extra verification of a user’s identity in which a user physically meets an appropriate trained auditor and presents his physical identity document to him for validation.

In order the trained auditor be able to declare the results of a face-to-face verification we have implemented a console in our web application. This console is shown in the figure below.

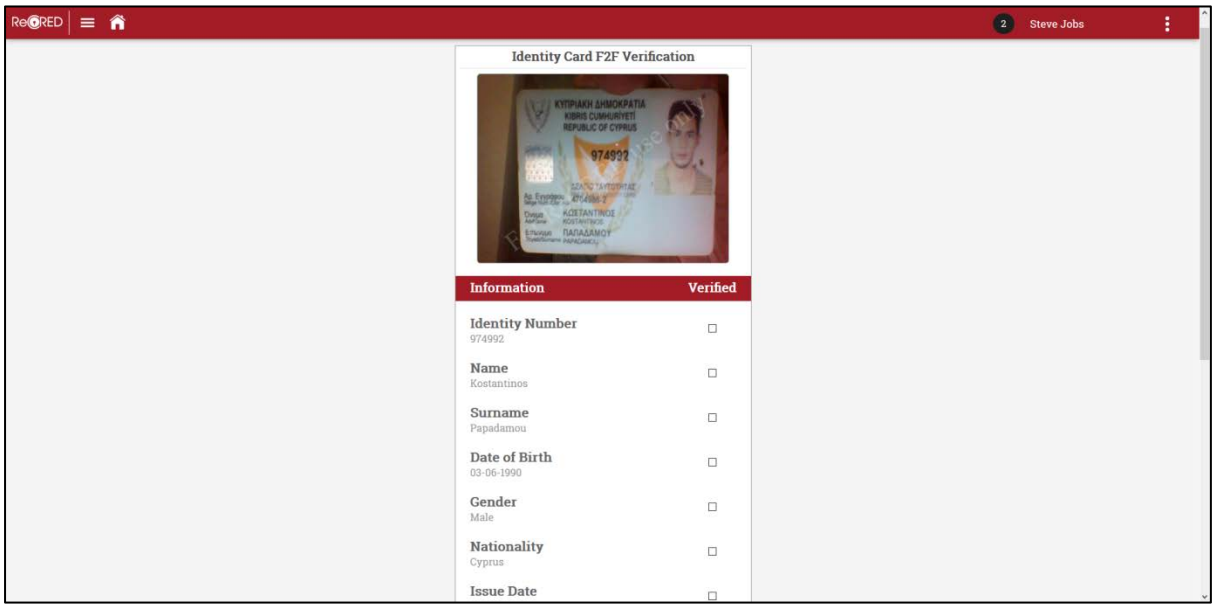

Figure 123. Face-to-face identity verification console for f2f auditors

First, the auditor has to declare the email of the user that he wants to verify and the he is requested to choose the identity document that audited user wants to verify. Then, the auditor is presented with all the identity information that the audited user declared to the Physical Identity Acquisition application and the auditor is able to verify each identity attribute separately. When a face-to-face verification has failed the auditor is able to add a description with the reason that led him to not verify the specified identity document.

8.3 Physical Identity Acquisition mobile application

Besides the Physical Identity Acquisition web application, we have also implemented an Android application, which runs on any android device with Android 5.0 and over. This application supports the same functionalities as the Physical Identity Acquisition web application. Figure 118 below shows the main page of the Physical Identity Acquisition android application.

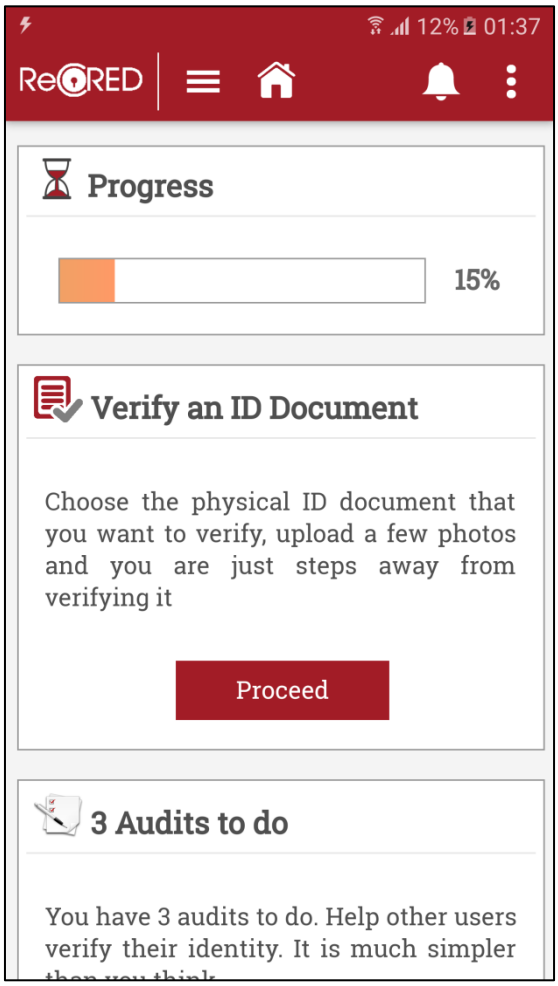

Figure 124. Physical Identity Acquisition mobile application mane page

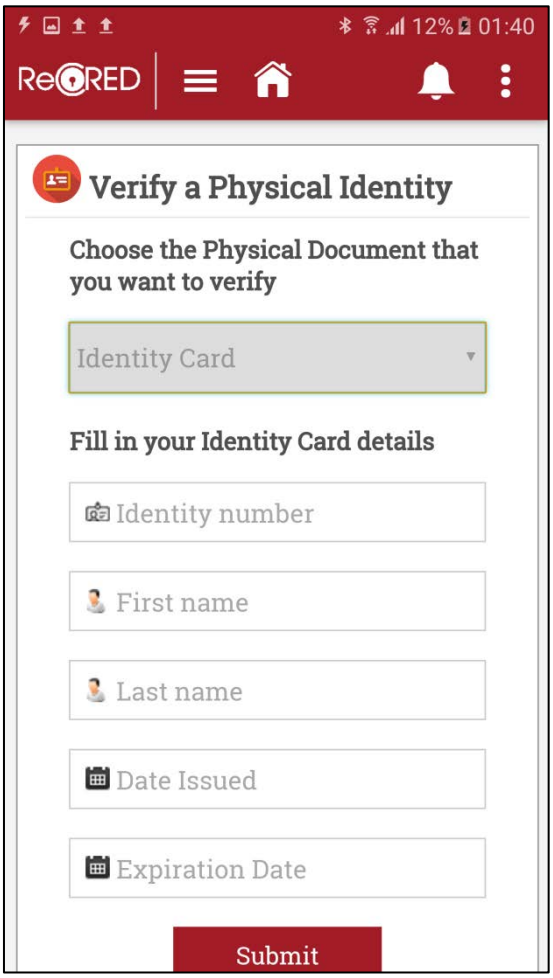

Figure 125. Physical Identity Acquisition mobile application: user declares the details of the identity document that he wants to verify

For the implementation of the Physical Identity Acquisition android application exploited the capabilities of the mobile devices in order to securely capture the identity information and physical characteristics of the users. Since in our application we are using all the available sensors on the devices (such as camera, GPS, etc.), a user has to authorize the application with the appropriate permissions to access theses sensors.

Furthermore, in order to enable the secure capturing of photos in our application and ensure that the submitted identity documentation photos have been captured from the device’s camera we utilize the Trusted Execution Environment (TEE) offered in commodity Android devices.

All the identity information acquired from the Physical Identity Acquisition mobile application is stored to the Identity Repository using the Storage API. The description for the supported functionalities of the Physical Identity Acquisition mobile application is included in the Physical Identity Acquisition and the Physical Identity Verification subsections above.

8.4 Physical Identity Acquisition Internal REST API

Besides the functionalities described above, the Physical Identity Acquisition module offers the Physical Identity Acquisition internal API, which is used by all the other modules of the Identity Consolidator platform when this is needed.

For the implementation of the Physical Identity Acquisition Internal API we use Slim [33], a PHP micro framework for the development of web applications and REST APIs. The REST operations that the current version of the API supports are described below.

8.4.1 User’s general identity verification status

Description: Provides information about the general identity verification status of a user.

Operation: GET /physical_id_acquisition_module/api/v1/general_verification_status/{user_id}

Request:

```
GET /physical_id_acquisition_module/api/v1/general_verification_status/{user_id}
Accept: application/json
```

Response:

```
200
Content-Type: application/json

{
        "user_id":"1",
        "verified":"true",
        "general_verification_progress":"90%",
        "verified_physical_identity_documents":"identity_card, passport"
}
```

Description of Elements in Request URI:

Element	Description	Valid Value
user_id	User’s unique Identifier	An integer up to 5 digits

Description of Elements in Response Body

Element	Description	Required	Valid Value
user_id	User’s unique Identifier	Yes	An integer up to 5 digits
verified	Declares whether the user has successfully verified one of his basic physical identity documents (identity card or passport)	Yes	TRUE or FALSE
general_verification_progress	Percentage that represents the general verification progress of the user	Yes	Float
verified_physical_identity_documents	A list of strings with all the verified physical identity documents	No	A list of strings

8.4.2 Physical Identity Document verification status

Description: Provides information about the verification status for a specific identity document of a user.

Operation: GET /physical_id_acquisition_module/api/v1/document_verification_status/{user_id}/{document_type}

Request:

```
GET
/physical_id_acquisition_module/api/v1/document_verification_status/{user_id}/{document_type}
Accept: application/json
```

Response:

```
200
Content-Type: application/json

{
        "user_id":"1",
        "document_type":"identity_name",
        "verified":"true",
        "verification_progress":"50%"
}
```

Description of Elements in Request URI:

Element	Description	Valid Value
user_id	User’s unique Identifier	An integer up to 5 digits
document_type	Physical Identity document type	identity, passport, profession_certificate, driving_license

Description of Elements in Response Body

Element	Description	Required	Valid Value
user_id	User’s unique Identifier	Yes	An integer up to 5 digits
document_type	The type of the physical identity document specified in the request	Yes	identity, passport, profession_certificate, driving_license
verified	Declares whether the user has successfully verified one of his basic physical identity documents (identity card or passport)	Yes	TRUE or FALSE
verification_progress	A percentage that represents the progress of the user on successfully verifying the specified identity document	Yes	Float

8.4.3 Identity Attribute verification status

Description: Provides information about the verification status for the specified identity attribute of a user.

Operation: GET
/physical_id_acquisition_module/api/v1/identity_attribute_status/{user_id}/{document_type}

Request:

```
GET /physical_id_acquisition_module/api/v1/identity_attribute_status/{user_id}/{attribute}
Accept: application/json
```

Response:

```
200
Content-Type: application/json

{
         "user_id":"1",
         "identity_attribute":"identity_name_surname",
         "verified":"true"
}
```

Description of Elements in Request URI:

Element	Description	Valid Value
user_id	User’s unique Identifier	An integer up to 5 digits
identity_attribute	The identity attribute that you want the verification status	String

Description of Elements in Response Body

Element	Description	Required	Valid Value
user_id	User’s unique Identifier	Yes	An integer up to 5 digits
identity_attribute	The identity attribute specified in the request	Yes	String
verified	Declares whether the specified identity attribute of the given user has successfully verified or not	Yes	TRUE or FALSE

8.4.4 Identity attribute peer-to-peer verification results

Description: Provides information about the peer-to-peer verification results for the specified identity attribute of a user.

Operation: GET
/physical_id_acquisition_module/api/v1/attribute_peertopeer_status/{user_id}/{document_type}

Request:

```
GET
/physical_id_acquisition_module/api/v1/attribute_peertopeer_status/{user_id}/{attribute}
Accept: application/json
```

Response:

```
200
Content-Type: application/json

{
          "user_id":"1",
          "identity_attribute":"identity_name_surname",
          "verification_status":"success",
          "verified":"true",
          "total_audits":"5",
          "positive_audits":"5",
          "negative_audtis":"0"
}
```

Description of Elements in Request URI:

Element	Description	Valid Value
user_id	User’s unique Identifier	An integer up to 5 digits
identity_attribute	The identity attribute that you want the peer-to-peer verification status	String

Description of Elements in Response Body

Element	Description	Required	Valid Value
user_id	User’s unique Identifier	Yes	An integer up to 5 digits
identity_attribute	The identity attribute specified in the request	Yes	String
verification_status	The verification status for the specified identity attribute	Yes	not_started, on_process, success, failed
verified	Declares whether the specified identity attribute of the given user has successfully verified or not	Yes	TRUE or FALSE
total_audits	The total number of audits that are assigned and related to the given identity attribute	No	Integer
positive_audits	The number of positive audits that the given user received and are related to the given identity attribute	No	Integer
negative_audits	The number of negative audits that the given user received and are related to the given identity attribute	No	Integer

9 Storage API

This section provides the description of the Storage API. This API defines a language- and platform-neutral protocol for Consumers to request, store and modify information of a user identity profile in the Identity Repository of the Identity Consolidator. This information contains identity attributes of the user and proofs of account ownership as well as other personal information.

Part of this specification is based on the model proposed by Portable Contacts [29] and OpenSocial [30]. This specification has been highly adapted to ReCRED requirements.

9.1 REST Services

ReCRED’s Storage API consist of collections of REST-accessible resources that can be accessed and modified using the basic set of HTTP request methods as defined by [I-D.ietf-httpbis-p2-semantics]. Such services have to be used by ReCRED modules or other third parties to access identity data hosted and managed by the ID Consolidator.

In general, for all REST Services:

- HTTP GET method is used to retrieve representations of the current state of any given resource
- PUT and PATCH are used to modify the current state
- DELETE is used to delete the resource
- POST is used to either create new resources or to perform other types of operations that do not fit within the scope of the other core HTTP methods

Implementations are free to support additional HTTP methods but their use is considered to be outside the scope of this specification.

Resources made available via a REST Service can represent individual objects (e.g. a person, an email information) or collections of objects (e.g. a friends list, a listing of user’s phones). Every resource is identified by a distinct URI to which the various HTTP methods are to be sent.

When a client application needs to communicate with a ReCRED server via an intermediary that restricts the use of certain standard and extension HTTP Methods (e.g., PUT, DELETE, and PATCH), the client COULD utilize the "X-HTTP-Method-Override" HTTP Request Header mechanism in a POST request. The "X-HTTP-Method-Override" header MUST NOT be used to send HTTP GET requests.

Responses to all requests specify an appropriate HTTP Status Code indicating the status of the response. All REST Services share a common, basic URI Structure that MAY be extended on a case-by-case basis. This common structure helps to ensure that all interactions remain as consistent as possible across multiple REST Services while allowing individual service-specific and implementation specific behaviors to be supported.

9.2 Header Fields

ReCRED services MAY use the request and response headers as defined at OData specification. In particular, it is highly recommended to support the following header elements:

DataServiceVersion: Clients MAY use the DataServiceVersion header on a request to specify the version of the protocol used to generate the request.

Content-Type: The format of an individual request or response body MUST be specified in the Content-Type header of the request or response.

Accept: As specified in [RFC2616], the client MAY specify the set of accepted formats through the use of the Accept Header.

If-Match: A client MAY include an If-Match header in a request to GET, PUT, MERGE, PATCH or DELETE an entity or entity property, or to invoke an action bound to an entity. The value of the If-Match request header MUST be an ETag value previously retrieved for the entity.

If specified, the request MUST only be invoked if the specified value matches the current ETag value of the entity. If the value does not match the current ETag value of the entity for a Data Modification or Action request, the service MUST respond with ‘412 Precondition Failed’ and MUST ensure that no data is modified as a result of the request.

ETag: A request that returns an individual entity MAY include an ETag header. The value specified in the ETag header may be specified in the If-Match (or If-None-Match) header of a subsequent Data Modification or Action request in order to apply optimistic concurrency in updating, deleting, or invoking the action bound to, the entity.

9.3 CREATE, UPDATE, and DELETE operations

A ReCRED service MUST support Create, Update, and Delete operations for all of the Identity elements that it exposes. A successfully completed Data Modification request must not violate the integrity of the data.

A client may request whether content be returned from a Create, Update, or Delete request, or the invocation of an Action, by specifying the Prefer Header.

9.3.1 Creation of information

To create an entity in a collection, send a POST request to that collection’s URL. The POST body MUST contain a single valid entity representation.

Upon successful completion, the response MUST contain a Location header that contains the edit URL of the created entity.

Upon successful completion the service MUST respond with either 201 Created, or ‘204 No Content’ if the request included a Prefer header with a value of "return-no-content’.

9.3.2 Conditional Requests

A client MAY include an ETag value in the if-match or if-none-match request header of a Data Modification or Action request. If specified, the operation MUST only be invoked if the if-match or if-none-match condition is satisfied.

The ETag value specified in the if-match or if-none-match request header may be obtained from an ETag header of a request for an individual entity, or may be included for an individual entry in a format-specific manner.

Note that the Entity Tag specified in the response is generally specific to the actual payload included in the response. If a REST service supports multiple representation formats for a single resource, such as offering multiple data format options or modified views of the resource tailored to the authentication credentials included in the request, the Entity Tag can vary for each specific response, regardless of whether the actual state of the resource on the server has changed. Therefore, for any single resource, multiple Entity Tags can potentially represent the current state of the resource.

9.3.3 Full Vs Partial Modification

Some update requests support two types of update: replace and merge. The client chooses which to execute by which HTTP verb it sends in the request. The current state of a resource may be modified in part or in full using either the PATCH [RFC5789] or PUT HTTP methods, respectively.

A PUT request indicates a replacement update. Given a URI that represents a resource, the current state of that resource can be modified in full by sending an HTTP PUT request to the URI. The payload of the PUT request is considered to be a replacement for the identified resource, although the server is free to determine exactly how the resource is to be modified.

Alternatively, the application can use a PATCH request to perform a partial modification of the resource. A PATCH or MERGE indicates a differential update. The service MUST replace exactly those property values that are specified in the request body. Missing properties, including dynamic properties, MUST NOT be altered. The semantics of PATCH are defined in [RFC 5789]. The service MUST be compliant with that definition. Support for the PATCH method to perform partial modifications of resource is optional.

Assuming the change is successful, the server would respond with an appropriate code status, and MAY include the updated representation of the resource.

9.3.4 Delete information

To delete an existing entity, send a DELETE request to that entity’s edit URL. The request body SHOULD be empty.

On success, the response MUST be 204 No Content.

9.4 Query Invocations

All requests to the ID Repository are made as HTTP GET operations on a URL deriving from the specified Base URL. Consumers MAY append additional path information and/or query string parameters to the Base URL as part of the request, as specified in Section 9.7.4. Additionally, authentication information MAY be sent via POST data or additional HTTP headers in the request, as specified in Section 9.4.1. Responses are returned in the body of the HTTP response, formatted as JSON or XML, depending on what is requested. Response and error codes SHOULD be transmitted via the HTTP status code of the response (if possible), and SHOULD also be specified in the body of the response, as described in Section 9.8 and Section 9.9. Since the API endpoint is dynamic (and does not serve static content), Consumers MUST NOT interpret any cache headers in the response as having meaning because the same URL request might return a different response upon subsequent invocation.

9.4.1 Authentication and Authorization

The data returned by an Identity Consolidator endpoint, through the Storage API, MAY contain public data, or it MAY contain private data. If the data returned is public, no authentication or authorization is required. Typically, this is done by Consumers of the Storage API obtaining either Direct Authorization, Basic Authorization with a username and password or Secured Authentication with an access token obtained out-of-band by the user, and given to the Consumer to present as part of the request. ReCRED specifies standard mechanisms for both types of authorization, so that Consumers may be able to obtain private data on a user's behalf from ID Providers in an automated

and consistent fashion. Regardless of the Authorization method used, the context of the request (i.e. the user for whom data is being requested) MUST be inferred by ID Providers from the Base URL and the authorization credentials provided. If public data is being accessed (and no authorization is provided), the Base URL MUST contain enough information for Identity Providers to know which data to return, but if private data is being accessed (and authorization is provided), the same Base URL MAY return information for different users depending on the authorization credentials provided.

9.4.1.1 Direct Authorization

IDC wishing to provide Direct Authorization MUST support HTTP Basic Access Authentication [RFC2617], and also support additional Direct Authorization mechanisms, if they choose. In addition to being a well-established mechanism for Direct Authorization, HTTP Basic has the added benefit of being understood by most Web Browsers, and can prompt users to enter their credentials as part of accessing a resource protected in this manner.

9.4.1.2 Available Authorization methods

IDC that provide access to private data MAY choose not to support either Direct Authorization, depending on their security requirements, but they MUST support either OAuth or HTTP Basic Authorization if they require any Authorization. When accessing a ReCRED endpoint, if sufficient authorization credentials are not provided, the IDC SHOULD return a 401 Unauthorized response, and SHOULD provide the available Authorization mechanisms available by including WWW-Authenticate headers in the response for each type of Authorization method supported (as defined in [RFC2616], section 14.47. Consumers will then be able to recognize that the Storage API is a protected resource and initiate the proper Authorization process needed to obtain the appropriate credentials.

9.4.2 Additional Path Information

A request using the Base URL alone MUST yield a result, assuming that adequate authorization credentials are provided. In addition, Consumers MAY append additional path information to the Base URL to request more specific information. Identity Providers MUST recognize the following additional path information when appended to the Base URL, and MUST return the corresponding data:

- /<path>/ Return all contact info (equivalent to providing no additional path info)
  e.g., http://consolidator.recred.eu:5000/ReCRED/StorageAPI/open/Providers
- /<path>({id}) Only return contact info for the contact whose id value is equal to the provided {id}, if such a contact exists. In this case, the response format is the same as when requesting all contacts, but any contacts not matching the requested ID MUST be filtered out of the result list by the Identity Provider.
  e.g., http://consolidator.recred.eu:5000/ReCRED/StorageAPI/open/Providers(1)

9.4.3 Query Parameters

Storage API defines a standard set of operations that can be used to filter, sort, and paginate response results. The operations are specified by adding query parameter to the Base URL, either in the query string or as HTTP POST data. ID Providers MAY support additional query parameters not specified here, and Providers SHOULD ignore any query parameters they don't recognize.

In particular, we use OData (http://www.odata.org/). The OData Protocol is an application-level protocol for interacting with data via RESTful interfaces. The protocol supports the description of

data models and the editing and querying of data according to those models. It provides facilities for:

- Metadata: a machine-readable description of the data model exposed by a particular data provider. http://consolidator.recred.eu:5000/ReCRED/StorageAPI/open/$metadata
- Data: sets of data entities and the relationships between them.
- Querying: requesting that the service performs a set of filtering and other transformations to its data, then returns the results.
- Editing: creating, updating, and deleting data.
- Operations: invoking custom logic
- Vocabularies: attaching custom semantics

The path of the URL specifies the target of the request (for example; the collection of entities, entity, navigation property, structural property, or operation). Additional query operators, such as filter, sort, page, and projection operations are specified through query options.

9.4.4 Filtering

Filtering is used to limit the request results to Contacts that match given criteria. The $filter system query option restricts the set of items returned.

ReCRED uses OData protocol and therefore the following operations and query filters are recommended. Providers MAY support a part of these methods. ID Providers MAY support additional filter operations if they choose. ID Providers MUST decline to filter results if the specified filter operation is not recognized (as per Section 6.3.5).

Built-in Filter Operations

OData supports a set of built-in filter operations, as described in this section. For a full description of the syntax used when building requests, see OData documentation [32].

The operators supported by the Storage API implementation in the expression language are shown in the following table.

Operator	Description	Example
Logical Operators		
Eq	Equal	/User_Accounts?$filter=Identifier eq 1
Ne	Not equal	/User_Accounts?$filter=Identifier ne 1
Gt	Greater than	/User_Accounts?$filter=Identifier gt 10
Ge	Greater than or equal	/User_Accounts?$filter=Identifier ge 10
Lt	Less than	/User_Accounts?$filter=Identifier lt 10
Le	Less than or equal	/User_Accounts?$filter=Identifier le 10
And	Logical and	/User_Accounts?$filter=Identifier le 10 and Identifier gt 3

Or	Logical or	/User_Accounts?$filter=Identifier le 3 and Identifier gt 10
Not	Logical negation	/User_Accounts?$filter=not endswith(Email,'com')

Built-in Query Functions

OData supports a set of built-in functions that can be used within $filter operations. The following table lists the available functions. For a full description of the syntax used when building requests, see OData documentation [32].

OData does not define an ISNULL or COALESCE operator. Instead, there is a null literal that can be used in comparisons.

Function	Example
String Functions	
bool substringof(string po, string p1)	/User_Accounts?$filter=substringof('hills', Email) eq true
bool endswith(string p0, string p1)	/User_Accounts?$filter=endswith(Email,'com') eq true
bool startswith(string p0, string p1)	/User_Accounts?$filter= startswith(Email, 'hills') eq true
string tolower(string p0)	/User_Accounts?$filter=tolower(DisplayName) eq ‘louisa’

Here are a few illustrative examples of filtering matches with OData. In each case, assume the following two users would be returned if no filtering parameters were provided:


```
User 1: {
        "AccessFailedCount": 0,
        "AccountCanceled": false,
        "AccountType": 1,
        "DisplayName": "Sam",
        "Email": "test@user2.com",
        "EmailConfirmUrl": "thisisatest.com",
        "Identifier": 1,
        "Language": "E",
        "LocationTrackingEnabled": 3,
        "LocationTrackingFinished": 3,
        "PhoneNumber": "234543214",
        "PhysicalIdentityVerificationInProgress": "",
        "RandomMove": 3,
        "RegisterDate": "/Date(1485890418000)/",
        "VerifiedEmail": false,
        "VerifiedPhoneNumber": false
},
User 2: {
        "AccessFailedCount": 0,
```

```
    "AccountCanceled": true,
    "AccountType": 1,
    "DisplayName": "Fay",
    "Email": "zora23@example.com",
    "EmailConfirmUrl": "3dd86bed5058326b99c9d4fff4",
    "Identifier": 2,
    "Language": "en",
    "LocationTrackingEnabled": 0,
    "LocationTrackingFinished": 0,
    "PhoneNumber": "+57(5)6491065489",
    "PhysicalIdentityVerificationInProgress": "profession_certificate",
    "RandomMove": -1,
    "RegisterDate": "/Date(337594339000)/",
    "VerifiedEmail": true,
    "VerifiedPhoneNumber": true
}
```


Given the parameters $filter=startswith(DisplayName, 'Sam'), only the first user (with Identifier=1) would match and be returned. Given the parameters $filter=substringof('zora', Email), only the second user (with Identifier =2) would match.

9.4.4.1 Sorting

Sorting allows requests to specify the order in which contacts are returned. The $orderby System Query option specifies the order in which items are returned from the service.

The value of the $orderby System Query option contains a comma-separated list of expressions whose primitive result values are used to sort the items. A special case of such an expression is a property path terminating on a primitive property. A type cast using the qualified entity type name is required to order by a property defined on a derived type.

The expression can include the suffix asc for ascending or desc for descending, separated from the property name by one or more spaces. If asc or desc is not specified, the service MUST order by the specified property in ascending order.

Null values come before non-null values when sorting in ascending order and after non-null values when sorting in descending order.

Items are sorted by the result values of the first expression, and then items with the same value for the first expression are sorted by the result value of the second expression, and so on.

9.4.4.2 Pagination

The pagination parameters can be used together to "page through" a large number of results in manageable chunks. The $top system query option specifies a non-negative integer n that limits the number of items returned from a collection. The service returns the number of available items up to but not greater than the specified value n. The $skip system query option specifies a non-negative integer n that excludes the first n items of the queried collection from the result. The service returns items starting at position n+1.

If no unique ordering is imposed through an $orderby query option, the service MUST impose a stable ordering across requests that include $top. Where $top and $skip are used together, $skip MUST be applied before $top, regardless of the order in which they appear in the request.

The $count system query option with a value of `true` specifies that the total count of items within a collection matching the request be returned along with the result.

For instance, on an initial query, specifying $skip=0&$top=10 will return only the first 10 results. The total number of possible results is indicated by $count=true, so the client knows how many "pages" of results exist. A subsequent query of $skip=10&$top=10 will return the next 10 results, and so on.

9.4.4.3 Presentation

Presentation controls the format, makeup, and delivery mechanism for returning the requested result set. The $select system query option requests that the service return only the properties, dynamic properties, actions and functions explicitly requested by the client. The service returns the specified content, if available, along with any available expanded navigation properties, and MAY return additional information.

The value of the $select query option is a comma-separated list of properties, qualified action names, qualified function names, the star operator (*), or the star operator prefixed with the namespace or alias of the schema in order to specify all operations defined in the schema.

If the $select query option is not specified, the service returns the full set of properties or a default set of properties. The default set of properties MUST include all key properties. If the service returns less than the full set of properties, either because the client specified a select or because the service returned a subset of properties in the absence of a select, the context URL MUST reflect the set of selected properties and expanded navigation properties.

The $format system query option specifies the media type of the response. The $format query option, if present in a request, MUST take precedence over the value(s) specified in the Accept request header. The value of the $format query option is a valid internet media type, optionally including parameters. Identity Providers MUST support the values json for JSON (http://json.org) and xml for XML (http://www.w3.org/XML/) and MAY support additional formats if desired.

9.4.4.4 Declining to honor query parameters

Providers SHOULD honor all filtering, sorting, and pagination requests specified via Query Parameters. However, in some instances it may be too burdensome to comply with a particular request, e.g. because the Provider does not have an efficient database index set up for a given field that is requested for filtering or sorting, and is unable to efficiently fetch all data and post-process the results to honor the request before returning the response. In such cases, Providers MAY decline to honor the request (or specific pieces of the request).

9.5 Response Format

The structure of the response object returned from a successful request MUST follows the OData specification. OData defines semantics around the following request and response headers. Additional headers may be specified, but have no unique semantics defined in OData.

9.6 Error Codes

The Identity Provider MUST returns a response code with every response. Response codes are numeric and conform to existing HTTP response codes where possible, as defined in OData. In addition to the response code, Identity Providers SHOULD also provide a human-readable reason

that explains the reason for the response code. This message SHOULD be intelligible to developers, but MAY be unsuitable for display to end-users. Clients SHOULD provide their own appropriate error message to users when encountering an error response.

Identity Providers MAY return additional codes to indicate additional information, but are discouraged from doing so and should instead augment the reason text with existing codes, if possible.

9.7 Structure

Each field is defined as either a Singular Field, in which case there MUST NOT be more than one instance of that field per contact, or as a Plural Field, in which case any number of instances of that field MAY be present per user profile.

Identity information is formatted using labeled attributes with either structured or unstructured string data. Each attribute value consists of one of the following types:

Simple: A single string attribute which MAY specify a REQUIRED data format or allow any string. A simple field MAY contain Canonical Values specified, in which case Identity Providers SHOULD try to conform to those values if appropriate, but MAY provide alternate string values to represent additional values.

Boolean: A special case of a Simple Field with two legal values: true and false. Values are case-sensitive.

Complex: A multi-value attribute that contains any combination of other attributes. Complex attributes are defined by listing the child attributes and their types. For most Complex Fields, the value sub-field contains the Major Value of that field (i.e. the primary piece of contact information described by that field), and the other fields provide additional meta-data.

9.7.1 entry Element

Unless otherwise specified, all fields are optional and of type xs:string. Also, unless specified, all field values MUST NOT contain any newline characters (\r or \n).

9.7.2 Singular Fields

id: Unique identifier for the Contact. Each Contact returned MUST include a non-empty id value. This identifier MUST be unique across this user's entire set of Contacts, but MAY not be unique across multiple users' data. It MUST be a stable ID that does not change when the same contact is returned in subsequent requests. For instance, an e-mail address is not a good id, because the same person may use a different e-mail address in the future. Usually, in internal database ID will be the right choice here, e.g. "12345".

displayName: The name of this Contact, suitable for display to end-users. Each Contact returned MUST include a non-empty displayName value. The name SHOULD be the full name of the Contact being described if known (e.g. Joseph Smarr or Mr. Joseph Robert Smarr, Esq.), but MAY be a username or handle, if that is all that is available (e.g. jsmarr). The value provided SHOULD be the primary textual label by which this Contact is normally displayed by the Identity Provider when presenting it to end-users.

published: The date this Contact was first added to the user's address book or friends list (i.e. the creation date of this entry). The value MUST be a valid xs:dateTime (e.g. 2008-01-23T04:56:22Z).

updated: The most recent date the details of this Contact were updated (i.e. the modified date of this entry). The value MUST be a valid xd:dateTime (e.g. 2008-01-23T04:56:22Z). If this Contact has never been modified since its initial creation, the value MUST be the same as the value of published. Note the updatedSince Query Parameter described in Section 6.3 can be used to select only contacts whose updated value is equal to or more recent than a given xs:dateTime. This enables Consumers to repeatedly access a user's data and only request newly added or updated contacts since the last access time.

birthday: The birthday of this contact. The value MUST be a valid xs:date (e.g. 1975-02-14. The year value MAY be set to 0000 when the age of the Contact is private or the year is not available.

anniversary: The wedding anniversary of this contact. The value MUST be a valid xs:date (e.g. 1975-02-14. The year value MAY be set to 0000 when the year is not available.

gender: The gender of this contact. Identity Providers SHOULD return one of the following Canonical Values, if appropriate: male, female, or undisclosed, and MAY return a different value if it is not covered by one of these Canonical Values.

note: Notes about this contact, with an unspecified meaning or usage (normally contact notes by the user about this contact). This field MAY contain newlines.

utcOffset: The offset from UTC of this Contact's current time zone, as of the time this response was returned. The value MUST conform to the offset portion of xs:dateTime, e.g. -08:00. Note that this value MAY change over time due to daylight saving time, and is thus meant to signify only the current value of the user's timezone offset.

9.7.3 Plural Fields

Within this specification, a "plural-field" is a property whose value consists of zero or more alternative choices represented as individual elements (phones, emails, etc.). Unless specified otherwise, all Plural Fields have the same three standard sub-fields:

value: The primary value of this field, e.g. the actual e-mail address, phone number, or URL. When specifying a sortBy field in the Query Parameters for a Plural Field, the default meaning is to sort based on this value sub-field. Each non-empty Plural Field value MUST contain at least the value sub-field, but all other sub-fields are optional.

type: The type of field for this instance, usually used to label the preferred function of the given contact information. Unless otherwise specified, this string value specifies Canonical Values of work, home, and other.

primary: A Boolean value indicating whether this instance of the Plural Field is the primary or preferred value of for this field, e.g. the preferred mailing address or primary e-mail address. Identity Providers MUST NOT mark more than one instance of the same Plural Field as primary="true", and MAY choose not to mark any fields as primary, if this information is not available. For efficiency,

Identity Providers SHOULD NOT mark all non-primary fields with primary="false", but should instead omit this sub-field for all non-primary instances.

The example below shows a Data Object with a single plural-field contact phone numbers for an individual.

```
User_Phones_Infos: {
        Info1: {
                "ConfidenceScore": 75,
                "LevelAssurance": 3,
                "PhoneNumber": "543984372",
                "Type": "work",
                "Updated": "/Date(1425370639000)/",
                "UserAccount": 1,
        },
        Info2: {
                "ConfidenceScore": 100,
                "LevelAssurance": 4,
                "PhoneNumber": "23453423",
                "Type": "home",
                "Updated": "/Date(1396426350000)/",
                "UserAccount": 1,
        }
}
```

When returning Plural Fields, the Identity Providers SHOULD canonicalize the returned value, if this is appropriate (e.g., e-mail addresses and URLs). Providers MAY return the same value more than once with different types (e.g. the same e-mail address may used for work and home), but SHOULD NOT return the same (type, value) combination more than once per Plural Field, as this complicates processing by the Consumer.

emails: E-mail address for this Contact. The value SHOULD be canonicalized by the Identity Provider, e.g. joseph@plaxo.com instead of joseph@PLAXO.COM.

urls: URL of a web page relating to this Contact. The value SHOULD be canonicalized by the Identity Provider, e.g. http://josephsmarr.com/about/ instead of JOSEPHSMARR.COM/about/. In addition to the standard Canonical Values for type, this field also defines the additional Canonical Values blog and profile.

phoneNumbers: Phone number for this Contact. No canonical value is assumed here. In addition to the standard Canonical Values for type, this field also defines the additional Canonical Values mobile, fax, and pager.

photos: URL of a photo of this contact. The value SHOULD be a canonicalized URL, and MUST point to an actual image file (e.g. a GIF, JPEG, or PNG image file) rather than to a web page containing an image. Identity Providers MAY return the same image at different sizes, though it is recognized that no standard for describing images of various sizes currently exists. Note that this field SHOULD NOT be used to send down arbitrary photos taken by this user, but specifically profile photos of the contact suitable for display when describing the contact.

relationships: A bi-directionally asserted relationship type that was established between the user and this contact by the Identity Provider. The value SHOULD conform to one of the XFN relationship

values (e.g. kin, friend, contact, etc.) if appropriate, but MAY be an alternative value if needed. Note this field is a parallel set of category labels to the tags field, but relationships MUST have been bi-directionally confirmed, whereas tags are asserted by the user without acknowledgment by this Contact. Note that this field is a Simple Field, meaning each instance consists only of a string value.

addresses: A physical mailing address for this Contact.

organizations: A current or past organizational affiliation of this Contact.

accounts: An online account held by this Contact.

9.7.4 Name Element

The components of the contact's real name. Providers MAY return just the full name as a single string in the formatted sub-field, or they MAY return just the individual component fields using the other sub-fields, or they MAY return both. If both variants are returned, they SHOULD be describing the same name, with the formatted name indicating how the component fields should be combined.

formatted: The full name, including all middle names, titles, and suffixes as appropriate, formatted for display (e.g. Mr. Joseph Robert Smarr, Esq.). This is the Primary Sub-Field for this field, for the purposes of sorting and filtering.

familyName: The family name of this Contact, or "Last Name" in most Western languages (e.g. Smarr given the full name Mr. Joseph Robert Smarr, Esq.).

givenName: The given name of this Contact, or "First Name" in most Western languages (e.g. Joseph given the full name Mr. Joseph Robert Smarr, Esq.).

middleName: The middle name(s) of this Contact (e.g. Robert given the full name Mr. Joseph Robert Smarr, Esq.).

honorificPrefix: The honorific prefix(es) of this Contact, or "Title" in most Western languages (e.g. Mr. given the full name Mr. Joseph Robert Smarr, Esq.).

honorificSuffix: The honorifix suffix(es) of this Contact, or "Suffix" in most Western languages (e.g. Esq. given the full name Mr. Joseph Robert Smarr, Esq.).

updated: The most recent date the details of this Person were updated (i.e. the modified date of this entry). The value MUST be a valid Date. If this Person has never been modified since its initial creation, the value MUST be the same as the value of published.

9.7.5 address Element

The components of a physical mailing address. Identity Providers MAY return just the full address as a single string in the formatted sub-field, or they MAY return just the individual component fields using the other sub-fields, or they MAY return both. If both variants are returned, they SHOULD be describing the same address, with the formatted address indicating how the component fields should be combined.

formatted: The full mailing address, formatted for display or use with a mailing label. This field MAY contain newlines. This is the Primary Sub-Field for this field, for the purposes of sorting and filtering.

streetAddress: The full street address component, which may include house number, street name, PO BOX, and multi-line extended street address information. This field MAY contain newlines.

locality: The city or locality component.

region: The state or region component.

postalCode: The zipcode or postal code component.

country: The country name component.

updated: The most recent date the details of this Person were updated (i.e. the modified date of this entry). The value MUST be a valid Date. If this Person has never been modified since its initial creation, the value MUST be the same as the value of published.

9.7.6 account Element

Describes an account held by this Contact, which MAY be on the Identity Provider's service, or MAY be on a different service. Consumers SHOULD NOT assume that this account has been verified by the Identity Provider to actually belong to this Contact. For each account, the domain is the top-most authoritative domain for this account, e.g. yahoo.com or reader.google.com, and MUST be non-empty. Each account must also contain a non-empty value for either username or userid, and MAY contain both, in which case the two values MUST be for the same account. These accounts can be used to determine if a user on one service is also known to be the same person on a different service, to facilitate connecting to people the user already knows on different services.

domain: The top-most authoritative domain for this account, e.g. "twitter.com". This is the Primary Sub-Field for this field, for the purposes of sorting and filtering.

username: An alphanumeric user name, usually chosen by the user, e.g. "jsmarr".

userid: A user ID number, usually chosen automatically, and usually numeric but sometimes alphanumeric, e.g. "12345" or "1Z425A".

updated: The most recent date the details of this Person were updated (i.e. the modified date of this entry). The value MUST be a valid Date. If this Person has never been modified since its initial creation, the value MUST be the same as the value of published.

Locked: Boolean reflecting whether the user account is locked for further actions (for instance, login in to that user account). This could happen for instance, if BAA informs to the ID Consolidator that user authentication has failed.

Trusted: Boolean reflecting whether the user trusts the Identity Consolidator to store the actual data.

9.7.7 verificationInfo Element

Describes additional information about the verification process on each record (single or plural) of this Contact, which MAY be by the Identity Provider's service, or MAY be by a different service (external or peer users). Consumers SHOULD NOT assume that this data has been verified by the Identity Provider.

status: The status of the verification process. Identity Provider SHOULD support at leat the following values: initiated, onProcess, verified.

startDate: The date this verification process has been initiated for the last time. The verification process could be a repeated one (periodically or by request). This information refers to the last time that this process has been initiated. This value SHOULD be a valid xs:date if possible, but MAY be an unformatted string, since it is recognized that this field is often presented as free-text.

endDate: The date this verification process has been completed for the last time. The verification process could be a repeated one (periodically or by request). This information refers to the last time that this process has been completed. This value SHOULD be a valid xs:date if possible, but MAY be an unformatted string, since it is recognized that this field is often presented as free-text.

verifiedBy: The name of the verification agent (e.g. company, user, Identity Provider or other organization). This field MUST have a non-empty value for each value returned.

description: A textual description of the verification process. This field MAY contain newlines.

9.7.8 mediaItem Element

MediaItem support collections of media items (video, image, sound) of the user.

id: Unique identifier for the media item.

title: The title of the media item.

description: Description of the media item

location: Location corresponding to the al media item.

thumbnailUrl: URL to a thumbnail cover of the media item.

mediaMimeType: String identifying the mime-types of media item in the media item.

9.7.9 urls Element

urls is the urls of blogs or webpages of the user.

id: Unique identifier for the url.

description: Description of the album

location: URL of the main page of the web site.

updated: The most recent date the details of this Person were updated (i.e. the modified date of this entry). The value MUST be a valid Date. If this Person has never been modified since its initial creation, the value MUST be the same as the value of published.

9.8 Example

Here is a sample request and response that illustrates much of Storage API.

Sample request (via HTTP GET):

http://consolidator.recred.eu:5000/ReCRED/StorageAPI/open/User_Accounts?$skip=10&$top=3&$orderby=DisplayName

Sample response (JSON):


```
{
 "d": {
  "results": [
   {
    "__metadata": {
     "id": "http://consolidator.recred.eu:5000/ReCRED/StorageAPI/open/User_Accounts(11)",
     "uri": "http://consolidator.recred.eu:5000/ReCRED/StorageAPI/open/User_Accounts(11)",
     "type": "StorageAPI.User_Account"
    },
    "AccessFailedCount": 4998,
    "AccountCanceled": true,
    "AccountType": 97,
    "DisplayName": "Keely",
    "Email": "llubowitz@example.net",
    "EmailConfirmUrl": "3fb09923fc35bb7c95cbd1ce8f",
    "Identifier": 11,
    "Language": "de",
    "LocationTrackingEnabled": 0,
    "LocationTrackingFinished": 0,
    "PhoneNumber": "1-814-793-2696x61816",
    "PhysicalIdentityVerificationInProgress": "driving_license",
    "RandomMove": -1,
    "RegisterDate": "/Date(1036322990000)/",
    "VerifiedEmail": false,
    "VerifiedPhoneNumber": true
   },
   {
    "__metadata": {
     "id": "http://consolidator.recred.eu:5000/ReCRED/StorageAPI/open/User_Accounts(15)",
     "uri": "http://consolidator.recred.eu:5000/ReCRED/StorageAPI/open/User_Accounts(15)",
     "type": "StorageAPI.User_Account"
    },
    "AccessFailedCount": 77775,
    "AccountCanceled": true,
    "AccountType": 869276,
    "DisplayName": "Kristy",
    "Email": "nbrakus@example.net",
    "EmailConfirmUrl": "bc8c4de1a626999f7b35b42968",
    "Identifier": 15,
    "Language": "es",
    "LocationTrackingEnabled": 0,
    "LocationTrackingFinished": 0,
    "PhoneNumber": "663.295.0681x0340",
```

```
    "PhysicalIdentityVerificationInProgress": "profession_certificate",
    "RandomMove": -1,
    "RegisterDate": "/Date(227776374000)/",
    "VerifiedEmail": false,
    "VerifiedPhoneNumber": true
   },
   {
    "__metadata": {
     "id": "http://consolidator.recred.eu:5000/ReCRED/StorageAPI/open/User_Accounts(19)",
     "uri": "http://consolidator.recred.eu:5000/ReCRED/StorageAPI/open/User_Accounts(19)",
     "type": "StorageAPI.User_Account"
    },
    "AccessFailedCount": 2150836,
    "AccountCanceled": true,
    "AccountType": 72750390,
    "DisplayName": "Laurel",
    "Email": "obie29@example.com",
    "EmailConfirmUrl": "a979e218d6a8bf86291614d12e",
    "Identifier": 19,
    "Language": "en",
    "LocationTrackingEnabled": 0,
    "LocationTrackingFinished": 0,
    "PhoneNumber": "06859269872",
    "PhysicalIdentityVerificationInProgress": "profession_certificate",
    "RandomMove": -1,
    "RegisterDate": "/Date(248140280000)/",
    "VerifiedEmail": false,
    "VerifiedPhoneNumber": false
   }
  ]
 }
}
```


9.9 Compatibility with Standards

This version of the Storage API is based on the Portable Contacts specification and is partially compatible with this specification. Portable Contaxts is currently (partially) compatible with the overlapping portion of the OpenSocial RESTful Protocol.

The main divergence between the Storage API, Portable Contacts and OpenSocial is the inclusion of OData for the Additional Path Information and Query Parameters. The Storage API has incorporated all the functionality defined by OData, enriching all the query mechanisms. In any case, this specification allows Identity Provider to implement Portable Contacts or OpenSocial mechanisms for query, sort or paging.

10 Identity Repository

This chapter defines the implemented Identity Repository data model. The designed schema is implemented using a traditional relational database MariaDB [31], a community-developed fork of the MySQL relational database. Note that this module serves as the central repository for the

Identity Consolidator. All the other modules of the Identity Consolidator use it through the Storage API to store and retrieve any information needed to develop their own functionalities.

10.1 Notation and Conventions

In this document we use the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" as described in [RFC2119]. Domain name examples use [RFC2606].

10.2 Definitions

Base URL: The root endpoint URL specified by the Online Service during Discovery and used to make requests. Consumers MAY append additional path information and query string parameters to this URL as part of the request.

Consumer: A website or application that uses the ReCRED protocol to request contacts managed by the Identity Provider.

Contact (or User): A record describing information about a particular person or entity, consisting of user information (e.g. name, e-mail addresses, phone numbers) and other descriptive information, as is typically found in address book and social networking applications.

Identity: A package of data about a user and their profile.

Identity Profile: The user data obtained from the Identity provider for authentication.

Identity (or Service) Provider: A web application that provides user information via the ReCRED protocol.

Portable Contacts: A standard protocol that provides users (and developers) a secure way to access their address books and friends lists without having to take their credentials or scrape their data.

Profile: Data associated with a user. Sometimes includes demographic or biometric information.

Verification Online Service: A web application that provides a verification service via the ReCRED protocol.

Singular Field: A contact field that can appear at most once per contact, e.g. displayName or gender.

Plural Field: A contact field that can appear multiple times per contact, e.g. emails or tags.

Simple Field: A Singular Field or Plural Field whose value is a single string attribute (see Section 9.5.1).

Complex Field: A Singular Field or Plural Field whose value is an object containing multiple sub-field attributes (see Section 9.5.1).

Canonical Value: String-valued contact fields that represent common values in a canonical form, e.g. "male" and "female" for gender. Identity Providers SHOULD conform to Canonical Values if appropriate, but MAY deviate if they need to represent additional values.

Primary Sub-Value: The sub-field in a Complex Field that should be used when sorting or filtering by that field. Unless otherwise specified, the value sub-field is always the Primary Sub-Field.

Account: A data record associated with one or more of a user’s identity.

10.3 Database design

The IDC serves as central repository which includes information about user identity, but also some other auxiliary information needed to allow the correct functionality of the different modules forming the Identity Consolidator.

The repository will store information using,

1. A user account table. This table will store information about the user account. This table will include a unique identification of the user (primary key) and other system related information like update date time, if the user has been validated or the preferred user profile information (display name, gender, etc.)
2. Several tables with plural information. Plural fields are information elements with any number of instances per contact. Plural fields include emails, phone numbers, urls, addresses, photos, videos or media items, positions, etc. These tables store consolidated information and the source of this could come from identity providers, users accounts or introduced directly by the user. All the plural information introduced directly by the user needs verification, so this information is connected directly with the Verification information.
3. Tables with single information, like the Physical Identity of a user, Service Providers or other information like biometrics and logs.
4. Information from external Identity Providers. Users have accounts or profiles in external providers such as an online service (e.g., Facebook), a bank or an operator. These tables store consolidated information as provided by external identity providers. All the providers are listed in a common Table Providers and then all the related information.

Figure X below represents this model.

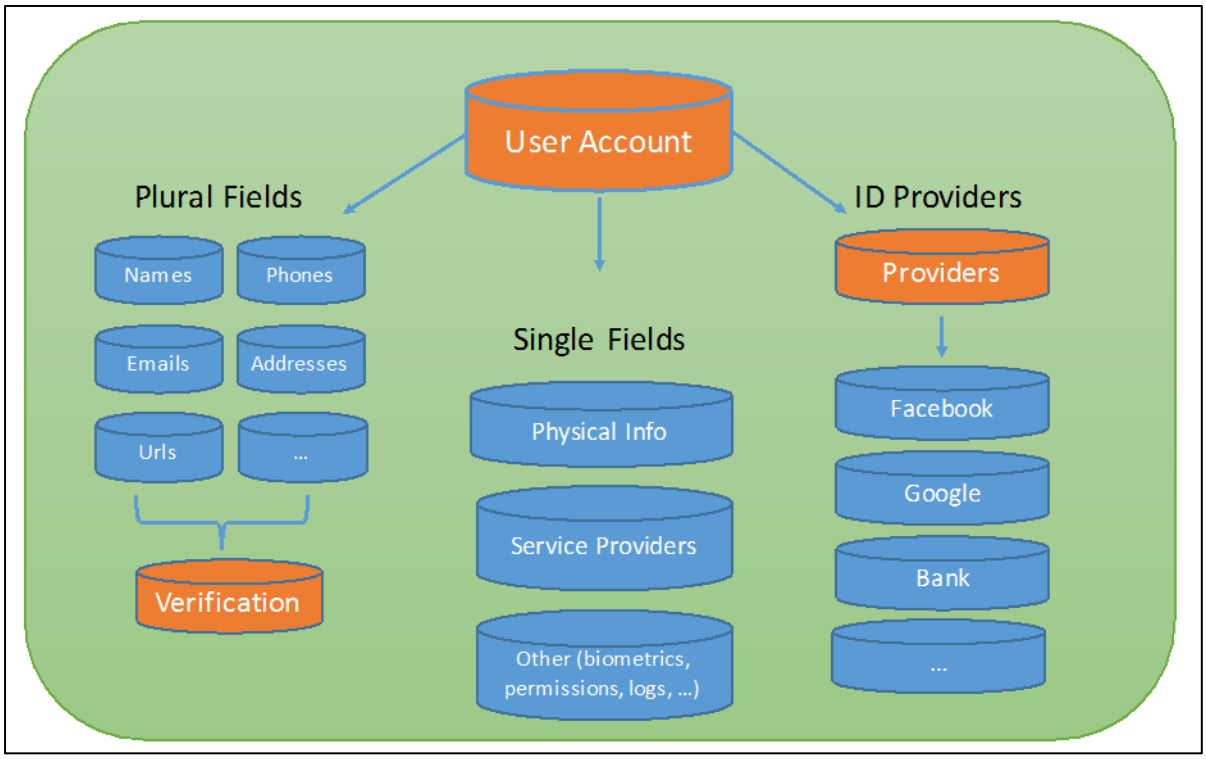

Figure 126. Storage model

10.4 User Account Table

This section outlines the fields in the central and preferred user profile structure. The Availability column shows whether or not the field is included in the profile data on public queries.

Key	Description	Availability
Identifier	The primary key of the user in the ReCRED database.	Guaranteed.
displayName	The name of this Contact, suitable for display to end-users. Each Contact returned MUST include a non-empty displayName value. The name SHOULD be the full name of the Contact being described if known (e.g. Joseph Smarr or Mr. Joseph Robert Smarr, Esq.), but MAY be a username or handle, if that is all that is available (e.g. jsmarr). The value provided SHOULD be the primary textual label by which this Contact is normally displayed by the Identity Provider when presenting it to end-users.	Guaranteed.
Email	An email address at which the person may be reached.	Available with user consent.
emailConfirmUrl	An url to verified the user email.	Available with user consent.
verifiedEmail	A boolean. True if the email has been validated.	Guaranteed.
phoneNumber	A phone number at which the person may be reached.	Available with user consent.
verifiedPhoneNumber	A boolean. True if the phone number has been	Guaranteed.

Key	Description	Availability
	validated.	
Language	The code of the language of the user, e.g. EN, ES	Guaranteed
physicalIdentityVerificationInProgress	The process of verifying (uploading photos) a specific physical identity document and which document is this none, identity, passport, driving_license, profession_certificate.	Guaranteed.
locationTrackingEnabled	Declares whether the user allowed the application to track his location for verifying his declared addresses.	Guaranteed.
locationTrackingFinished	1 if the location tracking period of the specific user has been passed. 0 if the user has not yet enabled the tracking of his location or it has been enabled but the 15 days period that we used to track the user's location has not passed.	Guaranteed.
randomMove	A randomly selected number for each user. This number declares a specific random move that the user is requested to do when capturing his front face photo in physical id acquisition application.	Guaranteed.
registerDate	The date of the user registration	Guaranteed.
accountType	The type of the user in the system. In Physical ID Acquisition module 0 means administrator, 1 means a typical user and 2 means that the user is a profession auditor in physical id acquisition platform.	Guaranteed.
accessFailedCount	The number of failed access.	Guaranteed.

Key	Description	Availability
accountCanceled	A Boolean. True if the account has been canceled	Guaranteed.

10.5 User Physical Identities Info Table

Key	Description	Availability
Identifier	Unique identifier for the Contact. Each Contact returned MUST include a non-empty id value. This identifier MUST be unique across this user's entire set of Contacts, but MAY not be unique across multiple users' data. It MUST be a stable ID that does not change when the same contact is returned in subsequent requests. For instance, an e-mail address is not a good id, because the same person may use a different e-mail address in the future. Usually, in internal database ID will be the right choice here, e.g. "12345"	Guaranteed.
primaryKey	The primary key of this entry in database.	Guaranteed.
documentType	The type of the physical documentation of the user. For instance, this filed will use values such as “Identity” or “Passport”.	Guaranteed.
documentNumber	The number of the user’s physical documentation. This field may be the number of his national identity or his passport number.	Available with user consent.
Gender	The gender of the user on his physical identity documentation.	Available with user consent.
dateOfBirth	The date of birth of the user on his physical identity	Available with

Key	Description	Availability
	documentation.	user consent.
Nationality	The nationality of the user on his physical identity documentation.	Available with user consent.
issueDate	The issued date of the user’s physical identity documentation.	Available with user consent.
expirationDate	The expiration date of the user’s physical identity documentation.	Available with user consent.
profession	The profession of the user.	Available with user consent.
namesInfo	The primaryKey of the namesInfo field in in the talbe User Names Info, related with the Physical ID information	Available with user consent.
verificationInfo	Describes additional information about the verification process on each record (single or plural) of this Contact, which MAY be by the Identity Provider's service, or MAY be by a different service (external or peer users). Consumers SHOULD NOT assume that this data has been verified by the Identity Provider. For more information, see Verification Info details.	Guaranteed.

10.6 Identity Providers data Table

This section outlines the fields in the normalized profile structure. The Availability column shows whether or not all providers include the field in their profile data responses.

Key	Description	Availability

Key	Description	Availability
Identifier	Unique identifier for the Contact. Each Contact returned MUST include a non-empty id value. This identifier MUST be unique across this user's entire set of Contacts, but MAY not be unique across multiple users' data. It MUST be a stable ID that does not change when the same contact is returned in subsequent requests. For instance, an e-mail address is not a good id, because the same person may use a different e-mail address in the future. Usually, in internal database ID will be the right choice here, e.g. "12345"	Guaranteed.
providerID	The ID which identified the provider.	Guaranteed.
primaryKey	Optional. A user ID number, usually chosen automatically by the Provider, and usually numeric but sometimes alphanumeric, e.g. "12345" or "1Z425A". This ID could be used as primary key of the user in the provider database.	Guaranteed.
formatted	The full name, including all middle names, titles, and suffixes as appropriate, formatted for display (e.g. Mr. Joseph Robert Smarr, Esq.). This is the Primary Sub-Field for this field, for the purposes of sorting and filtering.	Available, with user consent.
idp_formatted	A Boolean, specifies if the user give his consent to the IDP in such field.	Guaranteed.
familyName	The family name of this Contact, or "Last Name" in most Western languages (e.g. Smarr given the full name Mr. Joseph Robert Smarr, Esq.).	Available, with user consent.
idp_familyName	A Boolean, specifies if the user give his consent to the IDP in such field.	Guaranteed.

Key	Description	Availability
givenName	The given name of this Contact, or "First Name" in most Western languages (e.g. Joseph given the full name Mr. Joseph Robert Smarr, Esq.).	Available, with user consent.
idp_givenName	A Boolean, specifies if the user give his consent to the IDP in such field.	Guaranteed.
middleName	The middle name(s) of this Contact (e.g. Robert given the full name Mr. Joseph Robert Smarr, Esq.).	Available, with user consent.
ipd_middleName	A Boolean, specifies if the user give his consent to the IDP in such field.	Guaranteed.
Gender	The gender of this contact. Identity Providers SHOULD return one of the following Canonical Values, if appropriate: male, female, or undisclosed, and MAY return a different value if it is not covered by one of these Canonical Values.	Available from most providers, with user consent.
idp_Gender	A Boolean, specifies if the user give his consent to the IDP in such field.	Guaranteed.
Birthday	Date of birth in YYYY-MM-DD format. Year field may be 0000 if unavailable.	Available from most providers, with user consent.
idp_ Birthday	A Boolean, specifies if the user give his consent to the IDP in such field.	Guaranteed.
utcOffset	The offset from UTC of this contact’s current time zone, as of the time this response was returned. The value	Available from most providers, with user

Key	Description	Availability
	must conform to the offset portion of xs:dateTime, for example, -08:00. Note that this value may change over time due to daylight savings time, and is thus meant to signify only the current value of the user’s timezone offset.	consent.
idp_UtcOffset	A Boolean, specifies if the user give his consent to the IDP in such field.	Guaranteed.
Email	An email address at which the person may be reached.	Available from most providers, with user consent. Not available from Twitter, LinkedIn, and MySpace.
idp_Email	A Boolean, specifies if the user give his consent to the IDP in such field.	Guaranteed.
verifiedEmail	A Boolean, indicates whether the email has been verified or not.	Guaranteed.
URL	The URL of a webpage relating to this person.	Available from most providers, with user consent.
idp_URL	A Boolean, specifies if the user give his consent to the IDP in such field.	Guaranteed.
phoneNumber	A phone number at which the person may be reached.	Available from most providers, with user consent.

Key	Description	Availability
photo	The URL to a photo (GIF/JPG/PNG) of the person.	Available from most providers, with user consent.
idp_Photo	A Boolean, specifies if the user give his consent to the IDP in such field.	Guaranteed.
address1	See the address field section for details.	Available from most providers, with user consent.
idp_address1	A Boolean, specifies if the user give his consent to the IDP in such field.	Guaranteed.
address2	Aditicional addres field, in case the IDP has more than one address available.	Available from most providers, with user consent.
idp_address2	A Boolean, specifies if the user give his consent to the IDP in such field.	Guaranteed.
work	The work place of the user provide to the IDP.	Available from most providers, with user consent.
idp_Work	A Boolean, specifies if the user give his consent to the IDP in such field.	Guaranteed.
education	The education place of the user provide to the IDP.	Available from most providers, with user

Key	Description	Availability
		consent.
idp_Education	A Boolean, specifies if the user give his consent to the IDP in such field.	Guaranteed.
limitedData	A boolean value, true if social login was able to retrieve only limited public data from the user’s profile (for example, because the login session has expired or the user logged out from their account).	Provided by Facebook only.
Trusted	A boolean reflecting whether the user trusts the Identity Consolidator to store the actual data.	Guaranteed.
specificFields	Some Identity Providers return fields specific only to them. These fields will be present in a dictionary keyed by the provider field name. For instance, if provider is Linkedin, an entry with key “positions” have as a value a collection of positions each with isCurrent boolean and name (employer name) for each. As keys, we suggest to use string formats composing the name of the provider and the original name of the data. For instance, Facebook have the concept of games category and the key for that information will be represented as “Facebook:Games:Category”.	Available from some providers, with user consent.
idp_ SpecificFields	A Boolean, specifies if the user give his consent to the IDP in such field.	Guaranteed.
updated	The current timestamp when the data is saved for the first time or is updated.	Guaranteed.

10.7 User Names Info Table

This table contains the components of the contact's real name. Providers MAY return just the full name as a single string in the formatted sub-field, or they MAY return just the individual component fields using the other sub-fields, or they MAY return both. If both variants are returned, they SHOULD be describing the same name, with the formatted name indicating how the component fields should be combined.

Key	Description	Availability
identifier	Unique identifier for the Contact. Each Contact returned MUST include a non-empty id value. This identifier MUST be unique across this user's entire set of Contacts, but MAY not be unique across multiple users' data. It MUST be a stable ID that does not change when the same contact is returned in subsequent requests. For instance, an e-mail address is not a good id, because the same person may use a different e-mail address in the future. Usually, in internal database ID will be the right choice here, e.g. "12345"	Guaranteed.
primaryKey	The primary key of this entry in database.	Guaranteed.
formatted	The full name, including all middle names, titles, and suffixes as appropriate, formatted for display (e.g. Mr. Joseph Robert Smarr, Esq.). This is the Primary Sub-Field for this field, for the purposes of sorting and filtering.	Available, with user consent.
CS_formatted	The confidence score calculated for the formatted field.	Guaranteed
LoA_formateed	The level of assurance of the consolidated data for the formatted field.	Guaranteed
familyName	The family name of this Contact, or "Last Name" in most Western languages (e.g. Smarr given the full name Mr. Joseph Robert Smarr, Esq.).	Available, with user consent.

Key	Description	Availability
CS_familyName	The confidence score calculated for the familyName field.	Guaranteed
LoA_familyName	The level of assurance of the consolidated data for the familyName field.	Guaranteed
givenName	The given name of this Contact, or "First Name" in most Western languages (e.g. Joseph given the full name Mr. Joseph Robert Smarr, Esq.).	Available, with user consent.
CS_givenName	The confidence score calculated for the givenName field.	Guaranteed
LoA_givenName	The level of assurance of the consolidated data for the givenName field.	Guaranteed
middleName	The middle name(s) of this Contact (e.g. Robert given the full name Mr. Joseph Robert Smarr, Esq.).	Available, with user consent.
CS_middleName	The confidence score calculated for the middleName field.	Guaranteed
LoA_middleName	The level of assurance of the consolidated data for the middleName field.	Guaranteed
honorificPrefix	The honorific prefix(es) of this Contact, or "Title" in most Western languages (e.g. Mr. given the full name Mr. Joseph Robert Smarr, Esq.).	Available, with user consent.
CS_honorificPrefix	The confidence score calculated for the honorificPrefix field.	Guaranteed

Key	Description	Availability
LoA_honorificPrefix	The level of assurance of the consolidated data for the honorificPrefix field.	Guaranteed
honorificSuffix	The honorifix suffix(es) of this Contact, or "Suffix" in most Western languages (e.g. Esq. given the full name Mr. Joseph Robert Smarr, Esq.).	Available, with user consent.
CS_honorificSuffix	The confidence score calculated for the honorificSuffix field.	Guaranteed
LoA_honorificSuffix	The level of assurance of the consolidated data for the honorificSuffix field.	Guaranteed
updated	The most recent date the details of this Person were updated (i.e. the modified date of this entry). The value MUST be a valid Date. If this Person has never been modified since its initial creation, the value MUST be the same as the value of published.	Guaranteed.
verificationInfo	Describes additional information about the verification process on each record (single or plural) of this Contact, which MAY be by the Identity Provider's service, or MAY be by a different service (external or peer users). Consumers SHOULD NOT assume that this data has been verified by the Identity Provider. For more information, see Verification Info details.	Guaranteed.
moduleID	An identifier of the module who store the data in that table, in order to differentiate if the information has been provided by the end-user, the Physical Identity Acquisition Module or the Identity Integration Module or other	Guaranteed.

10.8 User Phones Info Table

This table contains Phone numbers for this Contact. No canonical value is assumed here. In addition to the standard Canonical Values for type, this field also defines the additional Canonical Values mobile, fax, and pager.

Key	Description	Availability
identifier	Unique identifier for the Contact. Each Contact returned MUST include a non-empty id value. This identifier MUST be unique across this user's entire set of Contacts, but MAY not be unique across multiple users' data. It MUST be a stable ID that does not change when the same contact is returned in subsequent requests. For instance, an e-mail address is not a good id, because the same person may use a different e-mail address in the future. Usually, in internal database ID will be the right choice here, e.g. "12345"	Guaranteed.
primaryKey	The primary key of this entry in database.	Guaranteed.
phoneNumber	A phone number at which the person may be reached.	Available, with user consent.
CS_phoneNumber	The confidence score calculated for the phoneNumber field.	Guaranteed
LoA_phoneNumber	The level of assurance of the consolidated data for the phoneNumber field.	Guaranteed
Type	The type of phone, with Canonical Values home, work, other. The list of Canonical Values could be extended in a future release.	Available, with user consent.
updated	The most recent date the details of this Person were updated (i.e. the modified date of this entry). The value MUST be a valid Date. If this Person has never been modified since its initial creation, the value MUST be the same as the	Guaranteed.

Key	Description	Availability
	value of published.	
verificationInfo	Describes additional information about the verification process on each record (single or plural) of this Contact, which MAY be by the Identity Provider's service, or MAY be by a different service (external or peer users). Consumers SHOULD NOT assume that this data has been verified by the Identity Provider. For more information, see Verification Info details.	Guaranteed.
moduleID	An identifier of the module who store the data in that table, in order to differentiate if the information has been provided by the end-user, the Physical Identity Acquisition Module or the Identity Integration Module or other	Guaranteed.

10.9 User Emails Info Table

Key	Description	Availability
identifier	Unique identifier for the Contact. Each Contact returned MUST include a non-empty id value. This identifier MUST be unique across this user's entire set of Contacts, but MAY not be unique across multiple users' data. It MUST be a stable ID that does not change when the same contact is returned in subsequent requests. For instance, an e-mail address is not a good id, because the same person may use a different e-mail address in the future. Usually, in internal database ID will be the right choice here, e.g. "12345"	Guaranteed.
primaryKey	The primary key of this entry in database.	Guaranteed.
email	An email address at which the person may be reached.	Available, with user consent.

Key	Description	Availability
CS_email	The confidence score calculated for the email field.	Guaranteed
LoA_email	The level of assurance of the consolidated data for the email field.	Guaranteed
type	The type of email, with Canonical Values home, work, other. The list of Canonical Values could be extended in a future release.	Available, with user consent.
updated	The most recent date the details of this Person were updated (i.e. the modified date of this entry). The value MUST be a valid Date. If this Person has never been modified since its initial creation, the value MUST be the same as the value of published.	Guaranteed.
verificationInfo	Describes additional information about the verification process on each record (single or plural) of this Contact, which MAY be by the Identity Provider's service, or MAY be by a different service (external or peer users). Consumers SHOULD NOT assume that this data has been verified by the Identity Provider. For more information, see Verification Info details.	Guaranteed.
moduleID	An identifier of the module who store the data in that table, in order to differentiate if the information has been provided by the end-user, the Physical Identity Acquisition Module or the Identity Integration Module or other	Guaranteed.

10.10 User Addresses Info Table

This table stores the components of a physical mailing address. Identity Providers MAY return just the full address as a single string in the formatted sub-field, or they MAY return just the individual component fields using the other sub-fields, or they MAY return both. If both variants are returned, they SHOULD be describing the same address, with the formatted address indicating how the component fields should be combined.

Key	Description	Availability
identifier	Unique identifier for the Contact. Each Contact returned MUST include a non-empty id value. This identifier MUST be unique across this user's entire set of Contacts, but MAY not be unique across multiple users' data. It MUST be a stable ID that does not change when the same contact is returned in subsequent requests. For instance, an e-mail address is not a good id, because the same person may use a different e-mail address in the future. Usually, in internal database ID will be the right choice here, e.g. "12345"	Guaranteed.
primaryKey	The primary key of this entry in database.	Guaranteed.
formatted	The full mailing address, formatted for display or use with a mailing label. This field MAY contain newlines. This is the Primary Sub-Field for this field, for the purposes of sorting and filtering.	Available, with user consent.
CS_formatted	The confidence score calculated for the formatted field.	Guaranteed.
LoA_formateed	The level of assurance of the consolidated data for the formatted field.	Guaranteed.
streetAddress	The full street address component, which may include house number, street name, PO BOX, and multi-line extended street address information. This field MAY contain newlines.	Available, with user consent.
CS_streetAddress	The confidence score calculated for the streetAddress field.	Guaranteed.
LoA_streetAddress	The level of assurance of the consolidated data for the streetAddress field.	Guaranteed.

Key	Description	Availability
city	The city or locality component.	Available, with user consent.
CS_city	The confidence score calculated for the city field.	Guaranteed.
LoA_city	The level of assurance of the consolidated data for the city field.	Guaranteed.
region	The state or region component.	Available, with user consent.
CS_region	The confidence score calculated for the region field.	Guaranteed.
LoA_region	The level of assurance of the consolidated data for the region field.	Guaranteed.
postalCode	The zipcode or postal code component.	Available, with user consent.
CS_postalCode	The confidence score calculated for the postalCode field.	Guaranteed.
LoA_postalCode	The level of assurance of the consolidated data for the postalCode field.	Guaranteed.
country	The country name component.	Available, with user consent.

Key	Description	Availability
CS_country	The confidence score calculated for the country field.	Guaranteed.
LoA_country	The level of assurance of the consolidated data for the country field.	Guaranteed.
type	The type of address, with Canonical Values home, work, other. The list of Canonical Values could be extended in a future release.	Available, with user consent.
latitude	The latitude value of the address	Available with user consent.
longtitude	The longtitude value of the address	Available with user consent.
updated	The most recent date the details of this Person were updated (i.e. the modified date of this entry). The value MUST be a valid Date. If this Person has never been modified since its initial creation, the value MUST be the same as the value of published.	Guaranteed.
verificationInfo	Describes additional information about the verification process on each record (single or plural) of this Contact, which MAY be by the Identity Provider's service, or MAY be by a different service (external or peer users). Consumers SHOULD NOT assume that this data has been verified by the Identity Provider. For more information, see Verification Info details.	Guaranteed.
moduleID	An identifier of the module who store the data in that table, in order to differentiate if the information has been provided by the end-user, the Physical Identity Acquisition Module or	Guaranteed.

Key	Description	Availability
	the Identity Integration Module or other	

10.11 User Acquired Locations Table

This table stores the location of the user that is acquired periodically from the device by the physical identity acquisition module. Those locations are then used for the verification of the user’s declared addresses.

Key	Description	Availability
identifier	Unique identifier for the Contact. Each Contact returned MUST include a non-empty id value. This identifier MUST be unique across this user's entire set of Contacts, but MAY not be unique across multiple users' data. It MUST be a stable ID that does not change when the same contact is returned in subsequent requests. For instance, an e-mail address is not a good id, because the same person may use a different e-mail address in the future. Usually, in internal database ID will be the right choice here, e.g. "12345"	Guaranteed.
primaryKey	The primary key of this entry in database.	Guaranteed.
latitude	The latitude value of the address	Available with user consent.
longitude	The longitude value of the address	Available with user consent.
capturedOn	The timestamp of the user’s captured location	Guaranteed.

Key	Description	Availability
partOfDayCaptured	Declares the part of the day that the specific location has been captured. We split the day in four parts (morning, noon, evening, night).	Guaranteed.
matchesAddressInfo	When the location tracking procedure has been finished we run a script that checks each captured location's coordinates of the user against the coordinates of his declared addresses and when they match we store in the matchesAddressInfo the primaryKey of the address in the User_Addresses_Info table.	
verificationInfo	Describes additional information about the verification process on each record (single or plural) of this Contact, which MAY be by the Identity Provider's service, or MAY be by a different service (external or peer users). Consumers SHOULD NOT assume that this data has been verified by the Identity Provider. For more information, see Verification Info details.	Guaranteed.

10.12 Urls Info Table

This table stores information about webpages or blogs of the user. It table has plural information about the contact or user.

Key	Description	Availability
identifier	Unique identifier for the Contact. Each Contact returned MUST include a non-empty id value. This identifier MUST be unique across this user's entire set of Contacts, but MAY not be unique across multiple users' data. It MUST be a stable ID that does not change when the same contact is returned in subsequent requests. For instance, an e-mail address is not a good id, because the same person may use a different e-mail address in the future. Usually, in internal database ID will be the right choice here, e.g. "12345"	Guaranteed.
primaryKey	The primary key of this entry in database.	Guaranteed.

Key	Description	Availability
url	An url with a WebPage, blog o similar element of the user.	Available, with user consent.
CS_url	The confidence score calculated for the url field.	Guaranteed
LoA_url	The level of assurance of the consolidated data for the url field.	Guaranteed
description	Description of the content of the page. This information is supplied by the user.	Available, with user consent.
updated	The most recent date the details of this Person were updated (i.e. the modified date of this entry). The value MUST be a valid Date. If this Person has never been modified since its initial creation, the value MUST be the same as the value of published.	Guaranteed.
verificationInfo	Describes additional information about the verification process on each record (single or plural) of this Contact, which MAY be by the Identity Provider's service, or MAY be by a different service (external or peer users). Consumers SHOULD NOT assume that this data has been verified by the Identity Provider. For more information, see Verification Info details.	Guaranteed.

10.13 MediaItem Table

This table stores information about media items (video, image, sound) of the user. It table has plural information about the contact or user.

Key	Description	Availability

Key	Description	Availability
identifier	Unique identifier for the Contact. Each Contact returned MUST include a non-empty id value. This identifier MUST be unique across this user's entire set of Contacts, but MAY not be unique across multiple users' data. It MUST be a stable ID that does not change when the same contact is returned in subsequent requests. For instance, an e-mail address is not a good id, because the same person may use a different e-mail address in the future. Usually, in internal database ID will be the right choice here, e.g. "12345"	Guaranteed.
primaryKey	The primary key of this entry in database.	Guaranteed.
documentCategory	Denotes the type of the physical document that the photo is, for example identity means that it is a photo related the identity card of the user The categories are: passport, identity, negative, driving_license, utility_bill, profession_certificate.	Guaranteed.
documentCategoryMediaType	Denotes the type of the picture: whole, redacted, name_surname, document_number, nationality, photo_in_document, date_of_birth, machine_readable_zone, face_photo, right_side_photo, left_side_photo, holding_id_document, profession, issued_date, address, other.	Guaranteed.
url	Location corresponding to this media item.	Available, with user consent.
title	The title of the media item. This information is supplied by the user.	Available, with user consent.
description	Description of the content of the media item. This	Available, with

Key	Description	Availability
	information is supplied by the user.	user consent.
thumbnailUrl	URL to a thumbnail cover of the media item.	Guaranteed.
mediaMimeType	String identifying the mime-types of media item in the media item.	Guaranteed.
updated	The most recent date the details of this Person were updated (i.e. the modified date of this entry). The value MUST be a valid Date. If this Person has never been modified since its initial creation, the value MUST be the same as the value of published.	Guaranteed.
verificationInfo	Describes additional information about the verification process on each record (single or plural) of this Contact, which MAY be by the Identity Provider's service, or MAY be by a different service (external or peer users). Consumers SHOULD NOT assume that this data has been verified by the Identity Provider. For more information, see Verification Info details.	Guaranteed.

10.14 Verification Table

Describes additional information about the verification process on each record (single or plural) of this Contact, which MAY be by the Identity Provider's service, or MAY be by a different service (external or peer users). Consumers SHOULD NOT assume that this data has been verified by the Identity Provider.

Key	Description	Availability
identifier	Unique identifier for the Contact. Each Contact returned MUST include a non-empty id value. This identifier MUST be unique across this user's entire set of Contacts, but MAY not be unique across	Guaranteed.

Key	Description	Availability
	multiple users' data. It MUST be a stable ID that does not change when the same contact is returned in subsequent requests. For instance, an e-mail address is not a good id, because the same person may use a different e-mail address in the future. Usually, in internal database ID will be the right choice here, e.g. "12345"	
primaryKey	The primary key of this entry in database.	Guaranteed.
status	The status of the verification process. Identity Provider SHOULD support at least the following values: initiated, onProcess, verified..	Guaranteed.
startDate	The date this verification process has been initiated for the last time. The verification process could be a repeated one (periodically or by request). This information refers to the last time that this process has been initiated. This value SHOULD be a valid xs:date if possible, but MAY be an unformatted string, since it is recognized that this field is often presented as free-text.	Guaranteed.
endDate	The date this verification process has been initiated for the last time. The verification process could be a repeated one (periodically or by request). This information refers to the last time that this process has been initiated. This value SHOULD be a valid xs:date if possible, but MAY be an unformatted string, since it is recognized that this field is often presented as free-text.	Guaranteed.
verifiedBy	The name of the verification agent (e.g. company, user, Identity Provider or other organization). This field MUST have a non-empty value for each value returned.	Guaranteed.
description	A textual description of the verification process. This field MAY contain newlines.	Guaranteed.

10.15 Audits Info Table

Key	Description	Availability
identifier	Unique identifier for the Contact. Each Contact returned MUST include a non-empty id value. This identifier MUST be unique across this user's entire set of Contacts, but MAY not be unique across multiple users' data. It MUST be a stable ID that does not change when the same contact is returned in subsequent requests. For instance, an e-mail address is not a good id, because the same person may use a different e-mail address in the future. Usually, in internal database ID will be the right choice here, e.g. "12345"	Guaranteed.
primaryKey	The primary key of this entry in database.	Guaranteed.
url	This is a random alphanumeric generated when the audit is assigned and will be used for implementation purposes. It MUST be unique for each entry in this table.	Guaranteed.
auditor	The user that is selected to perform this audit.	Guaranteed.
audited	The user that is being verified.	Guaranteed.
auditType	The type of the audit. This field can take integer values each one indicating a different type of audit.	Guaranteed.
auditResult	The result of the audit. The value of this field can take one the following values: assigned, positive, negative, and undetermined.	Guaranteed.
notifyResult	Denotes whether the user should be notified about the result of the specific verification (audit). True if he should be notified, False if he has already notified.	Guaranteed.
dateAssigned	The datetime that the audit has been assigned to the auditor.	Guaranteed.

10.16 Financial Information Table

This section outlines the fields for a banking loan origination scenario.

Key	Description	Availability
Identifier	The primary key of the user in the ReCRED database.	Guaranteed.
primaryKey	The primary key of this entry in database.	Guaranteed.
Income	The user’s income of last fiscal year in euro, data supplied by IRS	Available, with user consent.
idp_income	If the users relies in the IDP for such information	Guaranteed.
monthlyLoanPayments	Semicolon separated list of current monthly loan payments (decimal values, amount in euro), data supplied by National Banking Information System.	Available, with user consent.
idp_monthlyLoanPayments	If the users relies in the IDP for such information	Guaranteed.
overdueLoanPayments	total amount in euro of overdue loan payments, data supplied by National Banking Information System	Available, with user consent.
idp_overdueLoanPayments	If the users relies in the IDP for such information	Guaranteed.
Employed	employment status true/false, data supplied by social security organization.	Available, with user consent.

Key	Description	Availability
idp_employed	If the users relies in the IDP for such information	Guaranteed.
Name	The name of the user who provides the financial information.	Available, with user consent.
idp_name	If the users relies in the IDP for such information	Guaranteed.
phonenumber	The phone number of the user who provides the financial information.	Available, with user consent.
idp_phonenumber	If the users relies in the IDP for such information	Guaranteed.
Address	The address of the user who provides the financial information.	Available, with user consent.
idp_address	If the users relies in the IDP for such information	Guaranteed.
debtToState	The amount of money in euros that the user owes to the State.	Available, with user consent.
idp_debtToState	If the users relies in the IDP for such information	Guaranteed.
id_provider	A foreign key from table “Providers”, to indicate which Id Provider provides the source financial data	Guaranteed.

10.17 Cryptographic Credentials Table

Key	Description	Availability
identifier	Unique identifier for the Contact. Each Contact returned MUST include a non-empty id value. This identifier MUST be unique across this user's entire set of Contacts, but MAY not be unique across multiple users' data. It MUST be a stable ID that does not change when the same contact is returned in subsequent requests. For instance, an e-mail address is not a good id, because the same person may use a different e-mail address in the future. Usually, in internal database ID will be the right choice here, e.g. "12345"	Guaranteed.
primaryKey	The primary key of this entry in database.	Guaranteed.
cryptoCredential	The issued cryptographic credential.	
issueDatetime	The date and time that the cryptographic credential has been issued. This information is filled automatically when the credentials is stored.	
expirationDatetime	The date and time that the cryptographic credential will be expired. This information is provided by the user but it can be filled as NULL as soon as the user doesn’t specify it.	
Description	The description of the cryptographic credential. This information describes the identity attributes used to issue this cryptographic credential.	
included_attributes	The attributes included in the credential	
id_provider	A foreign key from table “Providers”, to indicate which Id Provider provides the source financial data	Guaranteed.

10.18 Behavioral Authentication Auths Table

Key	Description	Availability
Id	The primary key of this entry in database.	Guaranteed.
baaInfo	The information of the Behavioral Authentication Authority that holds the specific behavioral profile of the user.	Guaranteed.
behavioralProfileType	This information indicates the type of the behavioral profile of the user that the BAA knows. The value of this field can be gait, online behavior etc.	Guaranteed
active	A Boolean, if the information of the Behavioral Authentication Authority is active or not	Guaranteed.
updated	The most recent date the details of this entry were updated (i.e. the modified date of this entry). The value MUST be a valid Date. If this information has never been modified since its initial creation, the value MUST be the same as the value of published.	Guaranteed.

10.19 Behavioral Authentication Authorities Reference Table

Key	Description	Availability
identifier	Unique identifier for the Contact. Each Contact returned MUST include a non-empty id value. This identifier MUST be unique across this user's entire set of Contacts, but MAY not be unique across multiple users' data. It MUST be a stable ID that does not change when the same contact is returned in subsequent requests. For instance, an e-mail address is not a good id, because the same person may use a different e-mail address in the future. Usually, in internal database ID will be the right choice here, e.g. "12345"	Guaranteed.

Key	Description	Availability
primaryKey	The primary key of this entry in database.	Guaranteed.
Id	The identifier related with the Behavioral Authentication Auths table	Guaranteed.
updated	The most recent date the details of this entry were updated (i.e. the modified date of this entry). The value MUST be a valid Date. If this information has never been modified since its initial creation, the value MUST be the same as the value of published.	Guaranteed.

10.20 Service Providers Users Table

Key	Description	Availability
Identifier	The foreign key to the User Account identifier.	Guaranteed.
primaryKey	The primary key of this entry in database.	Guaranteed.
account_address	The URL of the 3rd party account	Guaranteed.
account_username	User’s account name in the 3rd party service. User’s real name could be “Danny Bush”, but on Facebook he would login as “dannybush”. A user MAY have more account usernames, which would result in separate rows in the table.	Guaranteed.
locked	A Boolean reflecting whether the user account is locked. This could be set by the user, or it could be set when BAA informs to the ID Consolidator that user authentication has failed.	Guaranteed.

Key	Description	Availability
approved	A Boolean indicates if the IDC has confirmed the validity of the Service Provider.	Guaranteed.
description	The description to the user account at this service provider (e.g.: e-banking account, shopping account)	Available, with user consent.
riskLevel	The risk level of the account, can be: Low, Medium or High.	Guaranteed.
userLocked	The status of the account defined by the user, the user can lock or unlock manually the account or set it to ‘Pol’ in order to follow the existing policies.	Guaranteed.
baaPolicy	A Boolean, indicates whether a user wants to be locked on BAA alerts.	Guaranteed.
baaLocked	A Boolean, if true indicates that a BAA has raised an alert for unusual behavior and the specific account is locked. Depends on the baaPolicy field value.	Guaranteed.

10.21 Identity Providers Users Table

Defines the locking policies for the Identity Provider accounts of the users.

Key	Description	Availability
Identifier	The foreign key to the User Account identifier.	Guaranteed.
primaryKey	The primary key of this entry in database.	Guaranteed.

Key	Description	Availability
providerID	The foreign key to the Service Provider entry.	Guaranteed.
locked	A Boolean, if the account is locked or unlocked by the application. This field is used if there are various policies will be enforced by the system using a daemon.	Guaranteed.
riskLevel	The risk level of the account, can be: Low, Medium or High.	Guaranteed.
userLocked	The status of the account defined by the user, the user can lock or unlock manually the account or set it to ‘Pol’ in order to follow the existing policies.	Guaranteed.
baaPolicy	A Boolean, indicates whether a user wants to be locked on BAA alerts.	Guaranteed.
baaLocked	A Boolean, if true indicates that a BAA has raised an alert for unusual behavior and the specific account is locked. Depends on the baaPolicy field value.	Guaranteed.

10.22 Account Locking Policies Table

The end-user is able to define locking policies for his/hers accounts in the Service Providers and ID Providers. User can define policies based on the day and time of the week and on the risk level of the accounts. The policy is defined per user and risk level and optionally per Service Provider account or Identity Provider account.

Key	Description	Availability
Id	The primary key of this entry in database.	Guaranteed.

Key	Description	Availability
description	The description of the locking policies for the user account in the Service Providers and ID Providers.	Available, with user consent.
active	A Boolean, indicates if the account is active or not.	Guaranteed.
userAccount	The foreign key to the User Account identifier.	Guaranteed
riskLevel	The risk level of the locking policies defined by the user. The risk can be: Low, Medium or High.	Guaranteed.
idProvider	The foreign key to the ID provider primary key. Identify the identity provider is involved in the locking policies described, if any.	Available, with user consent.
serviceProvider	The foreign key to the Service provider primary key. Identify the service provider is involved in the locking policies described, if any.	Available, with user consent.
monday_start	The time to start the policy on Monday, if it is required by the user	Available, with user consent.
monday_end	The time to end the policy on Monday, if it is required by the user	Available, with user consent.
tuesday _start	The time to start the policy on Tuesday, if it is required by the user	Available, with user consent.
tuesday _end	The time to end the policy on Tuesday, if it is required by the user	Available, with user consent.

Key	Description	Availability
wednesday _start	The time to start the policy on Wednesday, if it is required by the user	Available, with user consent.
wednesday _end	The time to end the policy on Wednesday, if it is required by the user	Available, with user consent.
thursday _start	The time to start the policy on Thursday, if it is required by the user	Available, with user consent.
thursday _end	The time to end the policy on Thursday, if it is required by the user	Available, with user consent.
friday _start	The time to start the policy on Friday, if it is required by the user	Available, with user consent.
friday _end	The time to end the policy on Friday, if it is required by the user	Available, with user consent.
saturday _start	The time to start the policy on Saturday, if it is required by the user	Available, with user consent.
saturday _end	The time to end the policy on Saturday, if it is required by the user	Available, with user consent.
sunday _start	The time to start the policy on Sunday, if it is required by the user	Available, with user consent.
sunday _end	The time to end the policy on Sunday, if it is required by the user	Available, with user consent.

10.23 Providers Table

Key	Description	Availability
primaryKey	The primary key of this entry in database.	Guaranteed.
providerID	Unique identifier for the Provider. Each Provider returned MUST include a non-empty id value. This identifier MUST be unique across this user's entire set of Contacts, but MAY not be unique across multiple users' data. It MUST be a stable ID that does not change when the same contact is returned in subsequent requests.	Guaranteed.
providerName	The name of the Provider ID	Guaranteed.
description	The description of the provider	Guaranteed.
levelAssurance	The level of assurance of the provider	Guaranteed.
supportedAttributes	List all the attributes that the IDP supports.	Guaranteed.
endpoints	List the endpoints that the IDP provides.	Guaranteed.

10.24 User Notifications Info Table

Key	Description	Availability
Identifier	The primary key of the user in the database.	Guaranteed.

Key	Description	Availability
primaryKey	The primary key of this entry in database.	Guaranteed.
applicationId	The identifier of the recred application that created this notification.	Guaranteed.
notificationText	The content of the notification.	Guaranteed.
notificationHref	The url that the user should be redirected when he clicks on the notification.	Guaranteed.
viewed	Declares whether the notification has been viewed by the user or not.	Guaranteed.
issuedOn	The timestamp of the notification.	Guaranteed.

10.25 User Physical Documents Cropped Regions Table

Key	Description	Availability
Identifier	The primary key of the user in the database.	Guaranteed.
primaryKey	The primary key of this entry in database.	Guaranteed.
xAxis	The start point of the cropped document at x axis.	Guaranteed.
yAxis	The start point of the cropped document at y axis.	Guaranteed.

Key	Description	Availability
width	The width of the cropped document.	Guaranteed.
height	The height of the cropped document.	Guaranteed.

10.26 User Relationships Info Table

This table stores the relations between users in the system. When the type of the relationship is set to audited the relation between the two users is in terms of peer-to-peer audits. Each time a crowdsourced audit for a user is assigned to another user in the system a relationship is stored between the two users so that in the future the selected auditor will not be assigned an audit for the same user.

Key	Description	Availability
primaryKey	The primary key of this entry in database.	Guaranteed.
Identifier1	The identifier of the user 1, related with user 2.	Guaranteed.
Identifier2	The identifier of the user 2, related with user 1.	Guaranteed.
relationshipType	The type of the relationship between the two users: 'audited','friendship','following','transaction'	Guaranteed

11 Identity Integration Module

This section provides the description and the implementation details of the Identity Integration module of the Identity Consolidator. Identity Integration module is responsible for aggregating and consolidating the identity information collected from multiple Identity Providers for a specific user. The identity integration module is also responsible for assigning confidence scores for the veracity of the attributes and for labelling identity attributes based on their origin. For example, the Identity consolidator should be able to tell verifiers that it has acquired the age information of user A

through a named Identity provider and let the verifier determine how much it trusts the identity information of that provider.

The identity information of a user is stored in the Identity Providers Data and Financial Information tables from the Identity Repository. Hence, the data is extracted from those tables. Once the information from each individual provider is available it must be integrated. Integration process first needs to map the identity attributes from different providers referring to the same piece of information. Once these mapping has been done, we have to define an algorithm to solve conflicts and chose one of the possible values for a given attribute. We use an algorithm that relies on the Level of Assurance associated to different Identity Providers. In short, our algorithm, in case conflict between two Identity Providers, selects the attribute associated to the ID Provider with higher Level of Assurance. Details of this algorithm are provided below in this section. Once all conflicts have been solved, the consolidated information for a user is stored in the Plural-Fields Tables of the Identity Repository described in Section 9.3, specifically in User Names Info Table, User Phones Info Table, User Emails Info Table and User Addresses Info Table.

11.1 Consolidation of identity information from multiple Identity Providers

This subsection describes the algorithm developed for the consolidation of identity information from multiple Identity Providers and for the resolution of conflicts. In the current implementation we consider N (4) data providers: Online Identity Providers (Facebook and Google), Banks, Telcos and Universities. 2. Each Data Provider has a list of predefined attributes and a predefined Level of Assurance (LoA). This information can be found in the Identity Repository.

Let's consider a given user *U*. For this user we have data from *M* different providers (*M* <= *N*), we proceed to the data consolidation as follows to generate an integrated profile:

A. If an attribute A is present only in 1 provider it passes immediately to the integrated profile of U with an assigned Confidence Score (CS) equal to the LoA of the provider, in the corresponding percentage, which provided that attribute.
B. If an attribute A is present in 2 or more providers belonging to different LoA
    i. If the value of the attribute matches in all LoA, the attributed is passed to the integrated profile and CS assigned is equal to the highest LoA, in the corresponding percentage. *Example: For a given attribute we have 3 providers (P1 with LoA = 4, P2 with LoA = 3 and P3 with LoA = 2). The CS will be 100 the maximum percentage corresponding for LoA 4.*
    ii. If the value of the attribute does not match between the different providers, the attribute associated to the provider with the highest LoA is selected and passed to the integrated profile. The CS is assigned as the Highest LoA - penalisation factor, in percentage. The penalisation factor (PF) increases with the number of mismatches in the attribute value, but will be in any case higher to the Highest CS - 1. *Example: For a given attribute we have 3 providers (P1 with LoA = 4, P2 with LoA = 3 and P3 with LoA = 2). The consolidated attribute will be the one provided by P1. If P1 mismatches with only P2 or P3 the CS will be 100 - PF1 (e.g., PF1 = 3.45 and CS = 96.55). If P1 mismatches with both P2 and P3 the CS will be 100 - PF2 (e.g., PF2 = 6.9 and CS = 93.1).*

C. If an attribute A is present in 2 or more providers belonging to the same LoA and there are discrepancies. There will be a predefined ranking between the providers belonging to the same LoA and the selected value of the attribute will be the one belonging to the highest ranked provided. For mismatches we will use a penalisation factor similarly as explained in step B.

In the end, the integrated profile will have a final list of attributed with a CS which is easily interpretable, integer numbers will indicate info coming from 1 of more providers from a given LoA without existing discrepancies. Whereas decimal numbers will reflect the level of assurance of the provider which provided the consolidated attribute as well the number of mismatches existing with other providers.

12 3rd Party REST API

The 3rd Party REST API that the Identity Consolidator platform is offers is used for communication purposes between the Identity Consolidator and external ReCRED entities (such as Service Providers) or other 3rd parties. In general, the 3rd Party API is used by any ReCRED component or online service who wants to interact with the Identity Consolidator platform. Additionally, 3rd Party API can be used Service Provider that wants to embed ReCRED’s Physical Identity Acquisition module in order to offer identity acquisition and verification to its users.

12.1 High Level Operations

The high level operations that the current version of the 3rd Party REST API supports can be classifier to the following categories:

1. Service Providers REST operations
    a. Transfer trust between Service Providers.
    b. Check status of a user’s account in a particular Service Provider
    c. Behavioral Authentication Authorities that have collected behavioral profiles for a user
2. Behavioral Authentication Authorities REST operations
    a. Inform the Identity Consolidator for signs of unusual behavior
    b. Inform the Identity Consolidator for BAAs authentication outcomes
    c. Inform the Identity Consolidator for the behavioral profiles stored by a Behavioral Authentication Authority
3. User’s Device REST Operations
    a. Cryptographic Credential Issuance from the Identity Consolidator to the User’s Device

12.2 Transfer trust between Service Providers

Description: This REST operation is used to transfer trust among Service Providers. For example, it can be used to transfer trust from eBay to Amazon.

Operation: POST /api/v1/transfer_trust/

Request:

```
POST /api/v1/transfer_trust/
Accept: application/json
```

Request Body:

```
{
        "user_id":721,
        "source_domain":"test1.com",
        "destination_domain":"test2.com"
}
```

Response:

```
200
Content-Type: application/json

{
        "user_id":721,
        "source_domain":"test1.com",
        "destination_domain":"test2.com",
        "attribute_description":"trust",
        "trust_value":"37",
        "transfer_datetime":"02/02/17 15:38"
}
```

Description of Elements in Request Body:

Element	Description	Valid Value
user_id	User’s unique Identifier	Integer
source_domain	The domain that is the holder of the trust value that we want to transfer	URL
destination_domain	The domain that will receive the trust value	URL

Description of Elements in Response Body

Element	Description	Required	Valid Value
user_id	User’s unique Identifier	Yes	Integer
source_domain	The domain that is the holder of the trust value that we want to transfer	Yes	URL
destination_domain	The domain that will receive the trust value	Yes	URL
attribute_description	The type of the identity attribute that we transfer. In this case this should be set to "trust"	Yes	"trust"
trust_value	The value of the trust of the user in the source domain	Yes	String
transfer_datetime	The date and time that the trust transferred.	Yes	DATETIME

12.3 User Account status in a particular Service Provider

Description: Provides information about the status (logged in or not) of a particular user’s account in a Service Provider.

Operation: GET /api/v1/user_account_status/service/{service}/user/{user_id}

Request:

```
GET /api/v1/user_account_status/ service/{service}/user/{user_id}
Accept: application/json
```

Response:

```
200
Content-Type: application/json

{
        "user_id":721,
        "service":"test.com",
        "account_status":"Logged"
}
```

Description of Elements in Request URI:

Element	Description	Valid Value
user_id	User’s unique Identifier	Integer
service	The domain name of the Service Provider that we are interested	URL

Description of Elements in Response Body

Element	Description	Required	Valid Value
user_id	User’s unique Identifier	Yes	Integer
service	The domain name of the Service Provider that we are interested	Yes	URL
account_status	Declares the status of the user’s account	Yes	Logged in or not

12.4 Behavioral Authentication Authorities that collected behavioral profiles for a user

Description: Provides information about the behavioral authentication authorities that have collected behavioral profiles for a user and can authenticate him. Used by Service Providers/Verifiers who wants to perform second-factor authentication against the user. Behavioral Authentication Authorities should make sure that the Identity Consolidator has always an updated reference regarding the behavioral profiles that they have stored about a user.

Operation: GET /api/v1/user_baas_reference_info/{user_id}

Request:

```
GET /api/v1/user_baas_reference_info/{user_id}
Accept: application/json
```

Response:

```
200
Content-Type: application/json

{
          "user_id":721,
          "behavioral_authentication_authorities_info":{
          "baa_name":"telefonica",
          "behavioral_profile_types":["browsing", “mobility”]
          },
          "updated":"02/02/17 15:39"
}
```

Description of Elements in Request URI:

Element	Description	Valid Value
user_id	User’s unique Identifier	Integer

Description of Elements in Response Body

Element	Description	Required	Valid Value
user_id	User’s unique Identifier	Yes	Integer
behavioral_authentication_authorities_info	A list with the information of all the Behavioral Authentication Authorities that stored behavioral profiles of a user	Yes	String
baa_name	The name of the Behavioral Authentication Authority	Yes	String
behavioral_profile_types	A list with the types of the behavioral profiles of the user that the BAA knows. (e.g., gait, online behavior etc.)	Yes	List of strings
updated	The most recent date the details of this entry were updated	Yes	DATETIME

12.5 Inform the Identity Consolidator for signs of unusual behavior

Description: This REST operation allows the Behavioral Authentication Authorities to contact and inform the Identity Consolidator about signs of unusual behavior. This will enable the Identity Consolidator to continuously authenticate users in the background. In the case of suspicious behaviour, the Identity Consolidator will also be timely informed to contact the Account management module in order to decide whether an account should be locked.

Operation: POST /api/v1/unusual_behaviour_report/

Request:

```
POST /api/v1/unusual_behavior_report/
Accept: application/json
```

Request Body:

```
{
   “user_id”: 721,
   “baa_name”: “telefonica”,
   “behavior_type”: “browsing”
}
```

Response:

```
200
Content-Type: application/json

{

}
```

Description of Elements in Request Body:

Element	Description	Valid Value
user_id	User’s unique Identifier	Integer
baa_name	The name of the Behavioral Authentication Authority who detected and reports the unusual behavior	String
behavior_type	The type of the behavioral profile that does not match with the usual behavior of the user	String

Description of Elements in Response Body

Element	Description	Required	Valid Value
-	-	-	-

12.6 Inform the Identity Consolidator for BAAs authentication outcomes

Description: This REST operation allows the Behavioral Authentication Authorities to contact and inform the Identity Consolidator about the outcome of authentication attempts. The received data will be used by the Identity Consolidator in conjunction with some pre-defined security policies to perform locks on some or all online accounts of a user.

Operation: POST /api/v1/authentication_outcome_results/

Request:

```
POST /api/v1/authentication_outcome_results/
Accept: application/json
```

Request Body:

```
{
   “user_id”: 721,
```

```
   “domain” : “test.com”
   “baa_name”: “telefonica”,
   “behavior_type”: “browsing”,
   “auth_outcome” : TRUE,
   “attempt_datetime” : “02/02/17 15:35”
}
```

Response:

```
200
Content-Type: application/json

{

}
```

Description of Elements in Request Body:

Element	Description	Valid Value
user_id	User’s unique Identifier	Integer
baa_name	The name of the Behavioural Authentication Authority who detected and reports the unusual behavior	String
domain	The domain of the Service Provider where the authentication attempt occurred.	URL
auth_outcome	If the authentication attempt was successful then the value is set to true. Otherwise the value is set to false	TRUE or FALSE
attempt_datetime	The date and time of the authentication attempt. This information is filled automatically when the REST operation is called	DATETIME

Description of Elements in Response Body

Element	Description	Required	Valid Value
-	-	-	-

12.7 Inform the Identity Consolidator for the behavioral profiles stored by a BAA

Description: This REST operation is used by the Behavioral Authentication Authorities that needs to update the list of the behavioral profiles of the user that it has.

Operation: POST /api/v1/update_baa_reference/

Request:

```
POST /api/v1/update_baa_reference/
Accept: application/json
```

Response:

```
200
Content-Type: application/json

{
“user_id” : 721,
“baa_name”: “telefonica”,
“behavioral_profiles”: [“browsing”],
“updated”: “02/02/17 15:32”
}
```

Description of Elements in Request Body:

Element	Description	Valid Value
user_id	User’s unique Identifier	Integer
baa_name	The name of the Behavioral Authentication Authority that requests the change	String
behavioral_profiles	A list of the behavioral profiles that the BAA collects about the specified user	List of strings
updated	The date and time that this change is requested	DATETIME

Description of Elements in Response Body

Element	Description	Required	Valid Value
-	-	-	-

12.8 Cryptographic Credential issuance to the User’s Device

Description: The user’s device is able to contact the Identity Consolidator and request the issuance of a cryptographic credential.

Operation: POST /api/v1/issue_cryptographic_credential/

Request:

```
POST /api/v1/issue_cryptographic_credential/
Accept: application/json
```

Request Body:

```
{
```

```
        "user_id":721,
        "identity_attributes [“age”],
        "idc_backup":TRUE
}
```

Response:

```
200
Content-Type: application/json

{
        "user_id":721,
        "cryptographic_credential":"sahdakjshdaskjhdskjashdkajsnfasnflas",
        "issued_datetime:”02/02/17 15:13",
        "expiration_datetime":"02/02/18 15:13",
        "description":"Age Credential"
}
```

Description of Elements in Request Body:

Element	Description	Valid Value
user_id	User’s unique Identifier	Integer
identity_attributes	The list of the identity attributes that should be used for the issuance of the cryptographic credential	List of strings
idc_backup	Declares whether the user wishes to backup the issued cryptographic credential to the Identity Consolidator	TRUE or FALSE

Description of Elements in Response Body

Element	Description	Required	Valid Value
user_id	User’s unique Identifier	Yes	Integer
cryptographic_credential	Declares whether the user has successfully verified one of his basic physical identity documents (identity card or passport)	Yes	TRUE or FALSE
Iisued_datetime	Percentage that represents the general verification progress of the user	Yes	DATETIME
expiration_datetime	A list of strings with all the verified physical identity documents	Yes	DATETIME
description	The description of the cryptographic credential. This information describes the identity attributes used to issue this cryptographic credential.	Yes	String

12.9 Physical Identity Acquisition Embed API

Besides the pre-described REST operations, 3rd Party API also offers the Physical Identity Acquisition embed API which enables any Service Provider/Verifier to embed the Physical Identity Acquisition module’s functionality to their website. Using this embed API, an Online Service can perform physical identity acquisition and verification for its users.

The embedding procedure is as easy as all the other available embed APIs (e.g., Google Maps Embed API). First the administrator of the interested Service Provider has to register as a developer from the Identity Consolidator’s page in order to acquire a unique API key. Each API key is bounded with the URL of the Service Provider. As soon as the Service Provider’s administrator has the API key, the only thing left to be done is to copy the following HTML code into the webpage of the service.

```
<iframe src=”//consolidator.recred.eu/
        &embed=true
        &type=physical_identity_acquisition
        &key=YOUR_API_KEY”
</iframe>
```

All the users’ acquired identity information are securely stored in the Identity Repository and maintained by the Identity Consolidator platform.

13 Conclusion

The Identity Consolidator Platform is the main component of the ReCRED architecture and takes place in most of the use cases of the ReCRED platform. This deliverable, “Full Identity Consolidator and Attributes Acquisition” describes the architectural design and provides the implementation details of the different modules that form the Identity Consolidator platform. Also, a description of the Identity Consolidator web application, which integrates all modules under one single application, is provided. The description APIs that the modules use to communicate with each other are included in this deliverable.

In this deliverable we also describe the Schema of the Identity Repository that is responsible to hold the collected identity information of the users along with a description of the Storage API, which is the way that the Identity Consolidator modules use for interacting with the Identity Repository.

Lastly, an integral part of the Identity Consolidator platform is the 3rd Party API which allows all the external ReCRED entities to communicate with the Identity Consolidator.

14 References

[1] OpenID Connect. http://openid.net/connect/
[2] The OAuth 1.0 Protocol. https://tools.ietf.org/html/rfc5849
[3] The OAuth 2.0 Authorization Framework. https://tools.ietf.org/html/rfc6749
[4] FiWARE. https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Privacy_Open_RESTful_API_Specification
[5] OpenAM. https://forgerock.org/openam/
[6] LATCH. https://www.elevenpaths.com/technology/latch/index.html
[7] Apache HTTP Server. https://httpd.apache.org/
[8] PHP. http://php.net/
[9] Laravel PHP Framework. https://laravel.com/
[10] WebRTC. https://webrtc.org/
[11] Near Field Communication (NFC). http://nearfieldcommunication.org/
[12] Jcrop API. http://jcrop.org/
[13] Google Maps JavaScript APi. https://developers.google.com/maps/documentation/javascript/
[14] PeerJS. http://peerjs.com/
[15] IRMA (I Reveal My Attributes). https://www.irmacard.org/
[16] Basic Access Control. https://en.wikipedia.org/wiki/Basic_access_control
[17] JMRTD library. http://jmrtd.org/about.shtml
[18] ICAO. http://www.icao.int/Pages/default.aspx
[19] RFC6749. https://tools.ietf.org/html/rfc6749
[20] Selenium WebDriver. http://www.seleniumhq.org/
[21] OCR. http://www.webopedia.com/TERM/O/optical_character_recognition.html
[22] phpOCR. https://sourceforge.net/projects/phpocr/
[23] Face detection. https://en.wikipedia.org/wiki/Face_detection
[24] OpenCV. http://opencv.org/
[25] PHP Facedetect. https://github.com/infusion/PHP-Facedetect
[26] Kairos SDK (PHP). https://github.com/infusion/PHP-Facedetect
[27] Kairos, Face Recognition API. https://www.kairos.com/face-recognition-api
[28] HTML5 Geolocation API. https://www.w3schools.com/html/html5_geolocation.asp
[29] Portable Contacts. http://portablecontacts.net/draft-spec.html#schema
[30] OpenSocial. https://en.wikipedia.org/wiki/OpenSocial
[31] MariaDB. https://mariadb.org/about/
[32] OData. http://www.odata.org/documentation/odata-version-2-0/uri-conventions/
[33] Slim. https://www.slimframework.com/
[34] Material UI. www.material-ui.com
[35] Moment.js http://momentjs.com
[36] Apache Olingo. http://olingo.apache.org
[37] ICAO Document 9303. https://www.icao.int/publications/pages/publication.aspx?docnum=9303