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 D4.1 "Identity Consolidator Baseline Platform "

Editor(s):	Rubén Cuevas (UC3M)
Author(s):	
Dissemination Level:	Public
Nature:	Report
Version:	5.~~3~~2

Formatted: Indent: Left: 0"

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 BY UTI	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	Spain
cnit	Consorzio Nazionale Interuniversitario per le Telecomunicazioni	Italy
BAKER & McKENZIE	Studio Professionale Associato a Baker & Mckenzie	Italy

Document History

AUTHORS
Rubén Cuevas (UC3M)
Kwstantinos Papadamou, Michael Sirivianos, Savvas Zannettou (CUT)
Agustín Santos Méndez (IMDEA)
Vangelis Bagiatis, Kostas Flokos, Lefteris Fanos (UPCOM)
Antonis Hatzikonstantinou, Yannis Katsaros (EXUS)
Alberto Caponi, Claudio Pisa and Luigi Stammati (CNIT)
Christoforos Ntantogian, Eleni Veroni, Faidon Lalagiannis, Nikolaos Vavoulas (UPRC)
Arsalan Najwani (VER)
Claudio Soriente, Michal Ficek (TID)
Evangelos Kotsifakos, Giorgos Gonidelis (WEDIA)

VERSIONS

Version	Date	Author	Remarks
0.10	2015-12-23	UC3M	Template and TOC creation
1.0		UC3M	Architecture diagram
1.1		IMDEA	Storage API description
1.2		IMDEA	Identity Repository Schema creation
2.1		IMDEA	Identity Integration Module Figures and description
2.2		IMDEA	Minor changes in Identity Repository
2.3		IMDEA	Minor changes in Storage API description
3.0		IMDEA, UC3M	Online Identity Acquisition Module
3.1 to 3.5		IMDEA	Minor changes in Storage API and Figures
3.6 to 3.9		IMDEA	Added new tables and minor DB schema changes, the Location table, Cryptographic Credential table
4.0		IMDEA, UC3M, CUT	Added content to Identity Consolidator Platform section, Introduction, Physical Identity Acquisition and 3rd Party API
4.1		EXUS, UPCOM	Added content to Identity Management Module
4.2		VERIZON, UPCOM	Added content to Authentication Management Module
4.3		TID	Added content to Account Management Module
4.4		CNIT	Added content to Credential Management Module
4.5		EXUS, UPCOM, CUT	Added content to Identity Management Module
4.6		UC3M, IMDEA, CUT	Added Introduction and Conclusion, general revision and several changes at Identity Repository Schema
4.7		CUT	Added content to Physical Identity

			Acquisition Module and 3rd Party API
4.8 to 5.1	2016-02-27	WEDIA, CUT	Comments and corrections
5.2	2016-02-29	WEDIA, CUT, UC3M	Proof reading – final version
5.3	2017-07-31	UC3M	Removed proprietary statement

Formatted: Left

Formatted: Font: (Default) +Body (Calibri), 11 pt, Font color: Auto

Formatted: Font: (Default) Times New Roman, 12 pt

Executive Summary

This document describes the design of the Identity Consolidator Platform. 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, authorization and authentication aspects.

The document presents a high overview of the architecture of the Identity Consolidator Platform and it carefully details the functionality and design of each one of the different components of the platform.

Table of Contents

List of Figures

1 Introduction

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 to perform horizontal binding of the online identities of a user and vertical binding of the real-word identities to independent verifiable identity attributes.

1.1 General Overview of the Identity Consolidation Platform

In general, the ID consolidator offers an identity management service that allows users to voluntarily submit identity attributes and proofs of account ownership. This service consists of the Physical Identity Acquisition module that is responsible for the acquirement and the verification of all the physical identity information of the user and also performs the vertical binding, and the Online Identity Acquisition Module that performs the horizontal binding. Those two modules are responsible to securely and verifiably collect identity attributes from the users. The Identity Repository in turn, is responsible of holding all the acquired identity information of the users. Additionally, the ID consolidator exchanges verified identity attributes with ID providers using federated login protocols (such as OpenID Connect, OAuth). The consolidator acts as a Relying Party when it collects the ID attributes from the ID providers. It also acts as ID provider when it proves to verifiers the collected ID attributes upon the user's request. The ID consolidator consists of the Authentication, Credential, Account and Identity Management modules which are described in detail in the rest of the document.

All of the aforementioned modules interact with each other using two well-defined REST APIs, the Integration and Storage APIs. All the other ReCRED entities, online services (verifiers) and Identity Providers that need to communicate with the Identity Consolidation Platform will use the 3rd Party API.

1.2 Structure of the document

The rest of this document is organized as follows. Section 2 describes the architecture of the Identity Consolidation platform and its components. Section 3 describes in detail the Authentication Management Module, Section 4 the Credential Management Module, Section 5 the Account Management Module and Section 6 the Identity Management Module and its sub-modules. In Section 7 we describe the Online Identity Acquisition Module and the methods that will be used for the obtainment of the different online identities of the users. Section 8 describes the Physical Identity Acquisition Module, the Identity Acquisition and verification processes. In Section 9 we provide a description of the Storage API, which defines a language protocol to store, request and modify the identity information stored in the Identity Repository while in Section 10 the Identity Repository that holds all the identity information is described. Section 10.17 describes in detail the Identity Integration Module and Section 12 the 3rd Party API that external entities will use to communicate with the Identity Consolidator. Finally, we conclude in Section 13.

2 Identity Consolidator Platform

Figure 1 shows the architecture of the Identity Consolidation platform. It is composed by the following modules:

- **Physical Identity Acquisition Module**: It is responsible for acquiring identity related information from the physical world (e.g., passport number)
- **Online Identity Acquisition Module**: it is responsible for acquiring the identity (i.e., validated account id) of a user in different online services (e.g., the user's validated account in Facebook, Twitter, etc.)
- **Identity Integration Module**: It is responsible to integrate the online identities of a user in different online services
- **Authentication Management Module**: It is responsible for the authentication of the user who wishes to gain access to the Identity Consolidator Platform for the purpose of managing his identity and his credentials. It is also responsible for the authentication of all the identity providers and BAAs that interact with the Identity Consolidator Platform.
- **Credential Management Module:** It provides the required functionality for the issuance and management of cryptographic credentials.
- **Identity Management Module:** This module is responsible for providing information to the user on which identity provider knows what about him. Moreover, this module allows users to define their consent to their various identity attributes and to transfer identity attributes from one identity provider to another.
- **Account Management Module:** It offers end-users user the ability to manage their user accounts with various Identity Providers and the BAAs, as well as to lock/unlock his accounts using the Latch functionality.
- **Identity Repository:** It is a central Database that stores all the information pertaining to user identity attributes as well as context related information (e.g., temporal validity of an attribute, its source and its confidence score). It serves as the central communication point for the interaction of the various modules.

In addition the Identity Consolidator Platform offers two APIs:

- **Storage API**: It allows the previously described modules to read, write and modify the information stored in the Identity Repository
- **3rd Party API**: It allows other external ReCRED entities and third parties (e.g., BAAs) to interact with the Identity Consolidator Platform. Additionally, it allows any online service that is interested to embed our Physical Identity Acquisition functionality to their webpage.

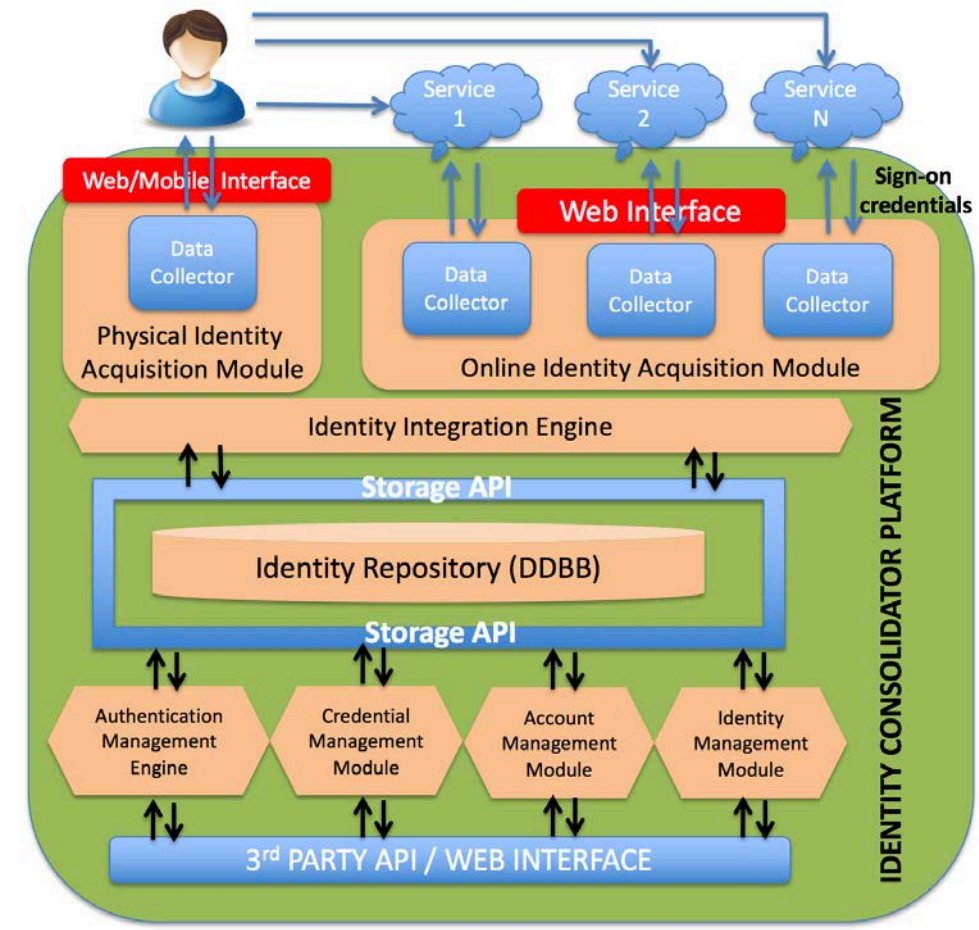


Figure 1 Identity Consolidator Platform

Comment [SZ1]: Updated after submission (Identity integration engine)

3 Authentication Management Module

We consider the ID Consolidator a highly critical component in the ReCRED architecture, since it is responsible for binding the online and real-world identities of the users, therefore can gather and store sensitive private information regarding its users. Each user's identity is composed of various attributes and different attributes can be classified into different sensitivity levels (public, personal, sensitive, etc.).

Moreover, each user can connect his ReCRED account to multiple Identity Providers (IDPs), such as a bank or a university, in order to transfer identity attributes from the IDP to the ID Consolidator. Different IDPs can require different levels of assurance, according to the sensitivity of the data that they provide. For example, the user can authenticate to his Facebook account by using a username and a password but usually additional factors are required for authenticating to a bank's e-banking system.

Taking these into account, the ID Consolidator offers a versatile Authentication Management Module (AMM), which supports multiple authentication methods, with each method offering a different Level of Assurance (LOA) and allowing access to different categories of the user's ID consolidator data and different actions on the platform. The supported authentication methods comply with the following LOAs defined by the National Institute of Standards and Technology (NIST), in terms of the likely consequences of an authentication error[1]. As the consequences of an authentication error become more serious, the required level of assurance increases. Of course, the token methods allowed for each given level are also allowed for all of its lower levels.

- **LOA 1**: At this level, the authentication mechanism provides some assurance that the same Claimant who participated in previous transactions is accessing the protected transaction or data. Successful authentication requires that the Claimant proves through a secure authentication protocol that he or she possesses and controls the token. Plaintext passwords or secrets are not transmitted across a network at LOA 1. However, this level does not require cryptographic methods that block offline attacks by eavesdroppers.
- **LOA 2**: At this level, single factor remote network authentication is provided and identity proofing requirements are introduced, requiring presentation of identifying materials or information. A wide range of available authentication technologies can be employed, such as Memorized Secret Tokens, Pre-Registered Knowledge Tokens, Look-up Secret Tokens, Out of Band Tokens and Single Factor One-Time Password Devices.
- **LOA 3**: At this level, multi-factor remote network authentication is provided, with at least two different authentication factors being required. In addition, identity proofing procedures require verification of identifying materials and information.
- **LOA 4**: This level is intended to provide the highest practical remote network authentication assurance. At this level, authentication is based on proof of possession of a key through a cryptographic protocol and in-person identity proofing is required. In essence, LOA 4 is similar to LOA 3, except that only "hard" cryptographic tokens are allowed.

The ID Consolidator offers a suite of web and mobile applications, through which the end-users have access to various functions, such as view and manage their identity attributes or verify their physical

[1] http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-2.pdf

identity. In order to grant access to the ID Consolidator's applications, the AMM provides the user with various authentication options. The first option is to access the Identity Consolidator by authenticating using its own authentication method (e.g., a very secure master password combined with hardware tokens or behavioral second factors). Another option is to allow access to IDC functionalities by authenticating the user through external affiliated IDPs that offers authentication through OpenID Connect. The user can then choose the IDP with which he wants to authenticate and one or more authentication options are provided (if the selected IDP supports more than one authentication options, i.e. both one-factor and multi-factor).

According to the chosen authentication method, the user is authenticated up to a certain LOA, which is decided by the IDP. Therefore, it is not the AMM that specifies the LOA but the Identity Consolidator or the IDP that was chosen. As soon as the user is authenticated and the LOA has been determined, then he can perform actions or view identity attributes associated with up to the current LOA.

For example, the user may choose to authenticate to the ID Consolidator (IDC) by signing in to his Facebook account (through a username / password challenge). In that case, the user is authenticated with LOA 2, therefore he can only perform actions or view identity attributes that are associates with LOA 1 and LOA 2 and his Facebook data. If the user wants to be granted access to additional functions and/or data, then he has to authenticate by choosing a method that offers a higher LOA or through the master authentication mechanism of the IDC or a different IDP. For example, he can choose to authenticate using Facebook's 2FA for LOA 3, or access his e-banking-related IDC stored data using a hard token for LOA 4. Nevertheless, the ID Consolidator and the IDPs may employ additional policies to regulate access to the identity attributes according to the authentication type (e.g., external IDP) and data sensitivity.

Comment [SZ2]: Added after submission

Furthermore, the AMM authenticates the users to the ID Consolidator's application through IDPs that offer OpenID Connect authentication. That means that the user can authenticate and provide consent on attributes transfer at the same time. More specifically, as soon as the user is authenticated to a certain IDP, the AMM asks the user to authorize the transfer of identity attributes from the IDP to the ID Consolidator (if not already authorized). In case the user consents, the attributes are transferred and stored in the Identity Repository. In this case the AMM communicates with the consent management submodule (Section 6) to ensure that the user- and ID Provider-defined policies are abided by.

The AMM provides some recovery mechanisms for cases where users are unable to log in to theirs ID Consolidator accounts (e.g., forgot their master password). Such mechanisms consist of a set of security questions that were defined during the registration with the ID Consolidator, SMS verification and prove of possession of online accounts (such as Facebook, Google, etc.,) via well know protocols (such as FacebookConnect, OpenID Connect, etc.).

Comment [SZ3]: Added after submission

Finally, the AMM is responsible for managing how the user interacts with online services (verifiers) that treat the IDC as an Identity Provider for the identity attributes collected by the IDC. To this end, the AMM invokes the 3rd party API to perform OpenID Connect/OAuth 2 with the verifiers and invokes the internal API to communicate with the consent management submodule (Section 6) to ensure that the user- and ID Provider-defined policies are abided by. To conclude, the AMM

implementation will be based on the OpenAM identity solution [OpenAM]. All the aforementioned features will be checked against the current implementation of OpenAM and we will provide extensions to cover all the scenarios that are required by ReCRED.

Comment [SZ4]: Added after submission

4 Credential Management Module

The Identity Consolidator (IDC) Credential Management (CM) module is responsible for the issuance of cryptographic credentials to the devices of the users. The main challenges on the design and the development of this module are: i) to securely map identity-attributes, acquired from consolidated identities, to cryptographic credentials to be issued to the user and ii) to provide interfaces and functionalities to make it possible to issue credentials from different sources (i.e. heterogeneous third-party Identity Providers). The CM module will initially integrate Idemix and U-prove cryptographic engines for credentials issuing and will extend the functionalities of the ReCRED credential issuance module running on the ReCRED Identity Providers. The credentials include a set of cryptographic attributes which can be either:

- stored on the ReCRED Identity Consolidator, i.e. extracted from already verified identity-attributes coming from a previous physical identity acquisition or online identity consolidation.

- managed by a third-party Identity Provider (e.g. University, Bank, Municipality, etc.) which:
    - **does not support** the issuance of cryptographic credentials and that relies on the Identity Consolidator credentials issuance capabilities.
    - **supports** the issuance of cryptographic credentials to which the user device is redirected (by the Identity Consolidator) when requesting credentials.

In order to allow the issuing of consistent credentials and the related definition of policies at the verifiers, the Identity Consolidator should allow to define (or acquire) a set of supported credential templates that define how the credentials are compiled (i.e. attributes required to fill-in the credentials) and how to treat them for policies definition. When the user interacts with the Identity Consolidator, she can select one of these formats and which of her acquired and verified identity-attributes are suitable to be included in the issued credential.

We assume that the user is already authenticated with their personal device which is in turn authenticated with the ReCRED Identity Consolidator. As an alternative, the user could authenticate and make use of the web interface to trigger credential issuance to its own personal device. The user should trust the Identity Consolidator and the appropriate identity-attribute matching checks are performed before the actual credential issuance process is started.

4.1 Credential Management Module Functionalities

As discussed above, the Credential Management module is designed to allow the user to receive cryptographic credentials for attribute-based access control by means of the Identity Consolidator.

To design a complete solution able to fulfill such requirement, the Credential Management module should implement functionalities that allow to:

1. Extend the Identity Consolidator functionalities in order to map acquired identity-attributes to cryptographic credentials.

2. Provide a trusted centralized component in the ReCRED architecture that provides support to the issuance procedure of third-party Identity Providers.

In any case the credentials are finally stored on the user device and are optionally (and based on user selection) backed-up into the Identity Consolidator. Indeed, the Credential Management module should support the secure backup (to restore in case of lost user device) of the user's owned credentials, by providing strong encryption that uses keys known only to/obtainable only by the user. In Figure 2~~Figure 2~~~~Figure 2~~ the high-level architecture of the Credential Manager module and the related interfaces to the involved components of the ReCRED framework are reported and discussed in the following sections.

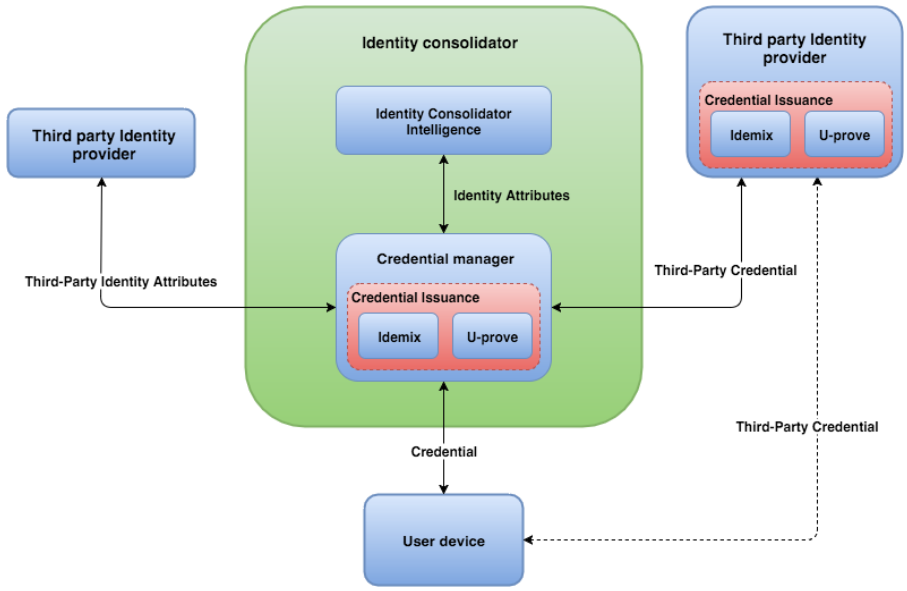


Figure 2 - Architecture and Interfaces for the Credential Issuance service provided by the Credential Manager module of the Identity Consolidator

4.2 Credential issuance from Identity-attributes

At the moment in which the user requests a credential, the Identity Consolidator has already acquired and stored the identity-attributes. As described in Sections 7 and 8, this attribute acquisition can happen either through user-provided physical items or through the retrieval of publicly available information from trusted resources. The ID Consolidator will be able to provide identity attributes to the Credential Management module through an API. Such identity attributes

will be used by the Credential Management module to compile related cryptographic credentials that the user requested for subsequent issuing to the user device.

4.3 Credential issuance from third-party Identity Providers

The ReCRED framework will provide a cryptographic issuance module to be run inside third-party identity providers in order to issue cryptographic credentials for attribute-based access control to users. Such module will allow the Identity Providers to independently run the credential issuance to the user's personal device without involving the Identity Consolidator in the issuance procedure. However, since the Identity Consolidator will be a central and trusted component in the ReCRED framework, it would be preferable to involve it in the credential issuance process.

The Credential Management module should extend the Identity Consolidator functionalities providing support for credential issuing from heterogeneous third party Identity Providers in different ways:

- The Credential Management module may serve as a trusted Identity Portal for External Identity Providers supporting ReCRED credentials issuance. Indeed, both the user requesting the issuance of a credential and the involved Identity Provider can exploit this functionality to request the Identity Consolidator to authenticate related parties. The Identity Provider and the Identity Consolidator exchange user authentication/sign-on information. Then the user is redirected to the Identity Provider, which does not require a new sign-on or identity verification and is thus able to issue directly (or by means of the Identity Consolidator) credentials to the user.

- The Credential Management module may provide to the user the functionality of trusted redirect to Identity Providers supporting ReCRED credentials issuance. The user contacts the Identity Consolidator in order to request a credential from an Identity Provider, as above. In this case the ID Consolidator does not directly interact with the Identity Provider but it simply redirects the user to the Identity Provider. Since there is no direct interaction between the Identity Provider and the Identity Consolidator, the user has first to authenticate to the Identity Provider. The issuance procedure then occurs directly between the Identity Provider running the ReCRED issuance module and the user.

- Identity Providers that do not support the ReCRED issuance module can take advantage of the Identity Consolidator proxy feature to issue credentials through the Credential Management module. In this case, the user contacts the Identity Consolidator and requests the issuance of a credential from an Identity Provider which does not support the ReCRED credential issuance module. The Identity Consolidator interacts with the external Identity Provider, acquires from it the appropriate attributes, and issues the requested credential through the Credential Management module on behalf of the third-party Identity Provider.

4.4 Credential Management Module Operations

The Credential Management module interacts with the Identity Consolidator Intelligence through REST APIs. These offer support for:

- Cryptographic credential issuance: the Identity Consolidator Intelligence can provide the Credential Management Module with user attributes to obtain a credential on their behalf
- Cryptographic credential backup: the Identity Consolidator Intelligence can take advantage of the backup feature of the Credential Management module by providing it with credentials to store. The Identity Consolidator Intelligence can then later enumerate and retrieve stored credentials for a specific user.

These interactions do not require access to the Identity Consolidator database from the Credential Management module. The Credential Management module has its own separate database in which credentials are stored. However, the user ids employed in the Credential Management module database are the same as the ones in the Identity Consolidator database. This allows the ID Consolidator Intelligence to use the user ids as an initial handle for credential management.

4.5 Supported Functionalities

This subsection describes the functionalities that the ReCRED platform offers to end-users, online service and ID provider administrators.

End-users:

- Issue cryptographic credentials: The user is presented with a list of supported credential templates and he can issue a new cryptographic credential based on the selected template. Cryptographic credentials can be issued either directly by the IDC (as long as the user trusts the IDC) or directly by a selected IDP which can issue a credential for the selected template. Alternatively, a new cryptographic credential can be issued automatically, whenever required, in order to grant access to an online service, according to specific rules defined by the provider (e.g. a proof of age).
- List supported attributes per IDP: The user is presented with a list of all the IDPs and for each IDP a list of all the supported attributes.
- List issued cryptographic credentials: The user is presented with a list of all the cryptographic credentials that have been issued to their device.
- View issued cryptographic credentials' details: The user can select an issued cryptographic credential, in order to see extended details for the selected credential (issued date, expiration date, issuer, whether it has been backed up to the IDC, a history of the credential's uses, etc.).
- Reissue expired cryptographic credentials: The user can select an expired cryptographic credential, in order to reissue it. A new credential is issued, by the same authority and with a new expiration date. Alternatively, the user can activate an option (through the user settings) authorizing the IDC to automatically attempt to reissue expired certificates.
- Encrypt/Decrypt cryptographic credentials: The user can select one or more credentials, in order to encrypt them or decrypt them.
- Backup cryptographic credentials: The user can select one or more credentials that have not been backed up to the IDC, in order to back them up. All the selected credentials are transferred to the IDC through a secure channel. Alternatively, the user can activate an option (through the user settings) to automatically backup issued credentials to the IDC.

- Restore cryptographic credentials: The user can select to restore any cryptographic credentials that are backed up to the IDC and not stored in the user's device (useful when a device is reset or a new device is purchased).
- Erase cryptographic credentials: The user can select to erase some or all of the cryptographic credentials that have been issued. The user has the options to erase the credentials from the IDC, from her device or both.

Online Service administrator:

- Create policy: The Online service administrator can create a new policy in order to grant access to a specific resource (e.g. access to the liquor department of an online store is allowed only for UK citizens that can provide a valid credential proving they are above a certain age). A rule can include conditions on specific attributes (age > 21 AND country = "UK") and for each attribute a list of "trusted" IDPs. Upon successful creation, the new rule is registered with the IDC.
- List and manage applied policies: The Online Service Administrator is presented with a list of the created policies. A policy can be selected in order to be updated or deactivated.
- View statistics: The Online Service Administrator can see statistics regarding user access through cryptographic credentials issued by ReCRED (totals, successful / unsuccessful by resource, date, etc.).

ID Provider administrator:

- Manage supported identity attributes: The IDP Administrator is presented with all the identity attributes used by the various credential templates and they can select which of those attributes are supported by the IDP. An IDP can issue cryptographic credentials only for templates for which all the required attributes are supported.
- Issue cryptographic credentials: The IDP Administrator can manually issue a credential and send it to the end user's device (e.g. in the case of a university administrator who registers students and professors to the campus Wi-Fi). A manually issued credential may or may not have an expiration date.
- List issued cryptographic credentials: The IDP Administrator is presented with a list of all the cryptographic credentials that have been issued.
- Revoke cryptographic credentials: The IDP Administrator can select a specific issued credential, in order to revoke it. The credential is erased from the user's device and/or the IDC (or it is marked as "revoked").
- View statistics: The IDP Administrator can see statistics regarding cryptographic credentials issued by the IDP.

5 Account Management Module

5.1 Account Management Module Functionality

The ReCRED framework includes an additional security measure to protect user accounts from illegitimate login attempts. Each account at an online service that participates in the ReCRED framework will carry a global status label. The status, either *locked* or *unlocked*, defines whether the

online service accepts authentication attempts for that particular account. In other words, before a user can log into their email account, the status of the account must be *unlocked*. If the account status is *locked*, the email provider does not accept authentication attempts for that specific account. As a result, even if the login credentials of a user have been leaked, an adversary will not be able to enter the victim's account, if the status of that account is *locked*.

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. Figure 3 shows the message flow of a login attempt that involves the ID Consolidator (on behalf of the Account Management Module).

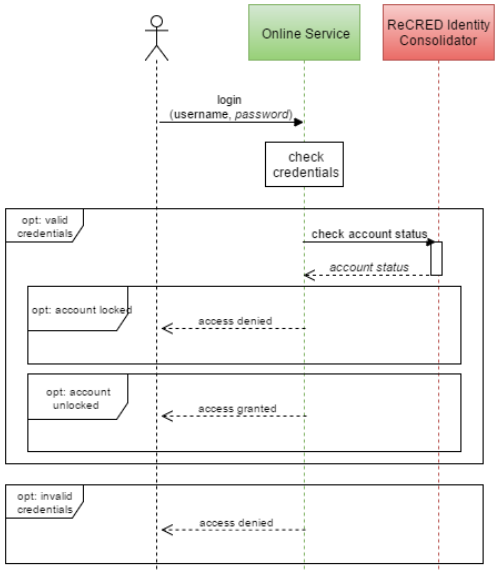


Figure 3 Message flow of a login attempt

When user *U* tries to log in at an online service, the latter will query the account management module for the status of *U*'s account. If the account is *locked*, the online service will reject the login, even if *U*'s credentials are valid. If the account is *unlocked*, the online service will grant the login, only if *U*'s credentials are valid. Note that Figure 3 depicts a login attempt via username and password. Nevertheless, the same procedure applies in scenarios where the user authenticates to the online service by other means (such as biometrics, FIDO, etc.).

Changing the status of *U*'s account can be either triggered by the legitimate account owner or by the ID consolidator, if it infers a high risk of account compromise. For example, a user can change the status of their account from *locked* to *unlocked* before logging in at the online service. A user can change the status of their account at any online service by logging into the account at the ID Consolidator. Similarly, a user may define arbitrary policies to automatically lock or unlock an account. For example, a user may define a policy to lock their corporate email account over weekends and to lock his e-banking account at night. ReCRED also allows the ID Consolidator to act on behalf of the user and lock his online accounts if the ID Consolidator detects a high risk of account compromise. For example, if the ID consolidator learns of a number of failed login attempts at *U*'s e-

banking account, it may decide to lock that account and also other accounts belonging to the same user. The ID consolidator may learn of failed login attempts via the Behavioral Authentication Authorities. The parameters to trigger account locking (e.g., number of failed authentication attempts, their frequencies, which other accounts to lock, etc.) may be chosen by the user.

The necessary attributes to support the Account Management module functionality are stored in User 3rd Party Accounts Table, described in Section 10.16.

The Account Management Module enables users to register a BAA or ID provider with the ID consolidator, and BAAs and ID provider admins to register their entities with the ID consolidator.

It is also responsible to act as a BAA discovery service for verifiers that require second factor device-to-service authentication.

Furthermore, the Account Management Module is considered as the main recovery mechanism for restoring users' accounts. Specifically, the Account Management Module stores information regarding the URL of a registered IDP and the respective username. The Account Management Module initiates the recovery procedure as described below:

- If the Credentials Management Module doesn't have the secret key for a respective IDP then it initiates the IDP-specific account recovery procedure.
- If the Credentials Management Module have the secret key for a respective IDP then the Account Management Module retrieves the secret key and it stores it to the new device. In this case, the IDP is agnostic of the recovery procedure.

Comment [SZ5]: Added after submission

5.2 Supported Functionalities

This subsection describes the functionalities that the ID Consolidator offers to end-users and are related to the Credential Management module.

End-users:

- Latch/ Unlatch online accounts: The user is presented with a list of all the connected online accounts and the status of each account (latched / unlatched – those terms could be replaced by locked / unlocked). The user can also manually latch an unlatched account or unlatch a latched one. Furthermore, he should be able to set policies to automatically lock and unlock accounts.
- View history of Latch changes: The user can see the history of all the latched / unlatched services. It includes both changes applied manually by the user or automatically by the IDC.
- Register Behavioral Authentication authority: The user can inform the ID consolidator which BAA authenticates him.
- Recover accounts: The user is presented with a list of the accounts that he have 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)

Comment [SZ6]: Added after submission

6 Identity Management Module

6.1 General Outline

The Identity Management module is common framework that serves as a standard for the definition and representation of user identity attributes within a given online service. Identity Management module is subdivided in two sub-modules, the identity 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. A representative use case is **reputation** in a certain online platform such as eBay, which we view as an attribute of that user. This identity attribute is currently used only by eBay and its users. 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.

The Identity Management module will be implemented as mobile and web interfaces.

6.2 Identity Management Module Architecture

As mentioned before, the Identity Management module is subdivided into two sub-modules, the Identity management sub-module and the consent management sub-module.

6.2.1 Identity profile m~~m~~anagement m~~sub-m~~odule

This ~~sub~~module provides the user's interface that allows the user to know and manage what each identity provider and verifier knows about them. It also enables the user to determine the risk of identity providers and verifiers inferring information about them that they didn't explicitly reveal to them. This information can leak by statistically analyzing correlations between identity attributes, thus the risk will be calculated by using similar techniques. Additionally, it allows a user to transfer identity attributes from one identity provider to another with respect to the policies defined in identity consent management module. It also allows the user to invoke the online ID acquisition module to transfer ID attributes from the ID providers to the IDC. Furthermore, the user has the ability to delete an identity attribute from an IDP. Lastly, this module provides the required functionality to the user to create and manage partial verifiable profiles.

6.2.2 Consent management ~~sub-~~module:

It allows users and IDPs to define their consent for their various identity attributes. Furthermore, the consent management module is subdivided into the following:

- User-defined policy for attribute transfer and proof: Such functionality is used when the user wants to define policies about to which IDPs and verifiers, their attributes should be revealed. For example the user may define that it does not wish to reveal their address to online social network services.

- 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. This is used, especially from ID Providers who do not wish certain attributes 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 (Section 4) is responsible for abiding by these set policies (for example not issue idemix credentials to the device if the IDP specified so). The ReCRED authentication app running on the device also has to abide by this policy. The profile management module also has to respect the IDP's and the user's policies regarding attribute transfer between IDPs.

Furthermore, as illustrated in Figure 1 the Identity Management module has access to the Identity Repository via the Storage API. It implements a REST API that allows users to manipulate their identity attributes maintained by the ID Consolidator. This API will be utilized both by a Web Application as well as a Mobile Application in order to accommodate a variety of User Interfaces and Devices (i.e. Desktop, Tablet, Smartphone).

6.3 Trust Modes

6.3.1 Fully Trusted

A user may choose to fully trust the Identity Consolidator. This means that all user attributes are stored along with their values in the Identity Repository. In this case whenever the user edits any of the stored attributes, Identity Management module stores all changes in the Identity Repository and contacts all the associated Identity Providers in order to have them update their local instances of the user’s attributes. In order to ensure consistency between the Identity Repository and the involved Identity Providers, the ID Management module will need to confirm the update and mark (in the Identity Repository) associated ID Provider records as “update confirmed” or not.

6.3.2 Not Trusted

Users may select to store only the associations between user attributes and ID Providers. In this case, the ID Management module will contact each of the associated ID Providers with the updated attribute values, and confirm that the update has been successful. It will also mark associated ID Provider records as “update confirmed” or not.

Note: Confirmation is the simple process of retrieving the user’s attributes from the Identity Provider and comparing them against the updated values.

6.4 Communication with Identity Providers

ID Providers incorporate in their software stack a ReCRED daemon that communicates with the corresponding ReCRED daemon running on the ID Consolidator in order to exchange identity attributes.

ReCRED daemons will leverage existing login protocols such as “OpenID Connect” and OAuth for the communication and exchange of data.

Therefore, ID Management module will get the base URL of each ID Provider from the Identity Repository, and will then have to get permission from the user to access each of the ID Providers. Then it will attempt an OpenID Connect/OAuth communication with the ID Providers in order to retrieve and/or update identity attributes for the requesting user.

6.5 Transfer of Identity Attributes between ID Providers

Transfer of attributes between ID Providers will be performed in the following way:

- The user specifies the attribute(s) and destination(s) ID Provider(s).
- The ID Management module retrieves from the Identity Repository the source ID Providers that maintain the attributes.
- It retrieves the requested attribute(s) from the source ID Provider(s) using the ReCRED daemon communication channel.
- It transfers the retrieved attribute(s) to the destination ID Provider(s) using the ReCRED daemon communication channel.
- It may display a “please confirm overwrite” prompt for destination ID Providers that indicate they already have an instance of this attribute for the user’s identity.
- It may then send a “overwrite confirmed” message to the ID Provider according to how the user answers the prompt.
- It confirms (i.e. retrieves and compares against the original) the stored attributes for each destination ID Provider.
- Finally, it stores in the Identity Repository an association between the retrieved attributes and the destination ID Providers.
- (Optional) In the case of Trusted Operation, the ID Management module will also store the retrieved attributes in the Identity Repository

Below is an outline of the REST API call that the ID Management module implements for Identity Attribute Transfer

Method	POST
URL	http://.../API/attribute/transfer
Content	{ “attributeId”:”a”, “sourceProviderId”:”x”, “destinationProviders”: [{“ProviderId”:”y”}, {“ProviderId”:”z”}, …] }

Response	{
	“attributeId”:”a”,
	“sourceProviderId”:“x”,
	“destinationProviders”:
	[
	{“ProviderId”:”y”, “verified”:true},
	{“ProviderId”:”z”, “verified”:false},
	…
	]
	}

6.6 Partial Verifiable Profiles

Users can group some of their attributes in profiles so that verifiers may be offered only a selection of verified attributes instead of the entire collection of the user’s stored identity. For example, a user may choose to only prove that he/she is a citizen of the European Union, or that a User is a registered professional with the corresponding national Professional Association.

The ID Management module implements a REST API to facilitate the creation and management of profiles:

Create:

Method	POST
URL	http://.../API/profile/user/1

Content	{
	“name”: ”My new partial profile”
	}

Response	{
	“id”:1234
	}

Retrieve:

Method	GET
URL	http://.../API/profile/1234

Response	{
	“id”:1234,
	“name”: ”My new partial profile”,
	“attributes”:
	[
	{“attributeID”:”x”}
	{“attributeID”:”y”}
	…
	]
	}

Add/Remove Attributes:

Method	PUT

URL	http://.../API/profile/1234

Content	{
	[
	{“attributeID”:”x”}
	{“attributeID”:”y”}
	…
	]
	}

Delete:

Method	DELETE
URL	http://.../API/profile/1234

6.7 Identity Management

The Identity Management module contains a web and mobile application, through which the users can view and manage their own identity data, as well as the identity attributes that each identity provider maintains for them or any online service knows. Therefore, the application contains two main interfaces:

1. **Identity Data Management**: 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 user-defined policy or the ID provider-defined policy for these attributes. For example, the user cannot change their name or age (especially if these have already been verified) but they can change their current job or address details. An update on some attributes may or may not trigger the initiation of an identity acquisition and verification process (e.g. a new proof of address). After the data is updated (and verified, if necessary), all the online services that maintain the updated identity attributes are notified and the values they hold are automatically updated as well.
2. **Shared Identity Attributes**: The user can view all of the identity attributes that are shared with various online services. This can be achieved in two alternative ways:
   a) The user selects an identity attribute and all the online services that maintain that attribute are fetched and displayed.
   b) The user selects an online service and all the identity attributes maintained by that service are fetched and displayed.

The application uses the Storage API in order to store and retrieve data from the Identity Repository. More specifically, the following methods are used:

- A GET method to retrieve the user’s identity data from the Identity Repository.
- A PUT method to update the user’s identity data.
- A GET method to search for identity providers that maintain data for a given identity attribute.
- A GET method to search for identity attributes that are maintained by a given identity provider.
- DELETE methods to:

- delete an attribute from the Identity Repository but let the associated ID Providers keep it (at least the source one).
- delete an attribute from an ID Provider but keep it in the Identity Repository.
- refresh the value of an identity attribute from the source ID Provider.
- toggle the trust mode of an identity attribute i.e., retrieve and store the value from the ID Provider in Full Trust and delete the value of the attribute from the Identity Repository in No Trust mode.

Example 1: Get the identity attributes for the user with id=1

Method	GET
URL	http://.../API/identity/user/1

Response	{
	“id”:1
	“lastName”:”Doe”
	“firstName”:”John”
	“birthdate”:”01-01-1980”
	…
	}

Example 2: Update the address details of the user with id=1

Method	PUT
URL	http://.../API/identity/user/1

Payload	{
	“address”:”10, Gloucester Road”
	“city”:”London”
	“country”:”UK”
	…
	}

Example 3: Get the online services that maintain the user’s street address attribute

Method	GET
URL	http://.../API/identity/serviceProvider?f=attribute=address

Response	[
	{
	“id”:1
	“serviceName”:”Service1”
	“URL”:”http://...”
	…
	}
	{
	“id”:2
	“serviceName”:”Service2”
	“URL”:”http://...”
	…
	}
	…
	]

Example 4: Get the identity attributes that are maintained by the online service with id=1

Methodf	GET
URL	http://.../API/identity/attribute?f=providerId=1

Response	[
	{
	"attributeId":1
	"attributeName":"last name"
	…
	}
	{
	"attributeId":2
	"attributeName":"first name"
	…
	}
	
	…
	]

Example 5: Delete the identity attributes specified in the request. The ID Management module will also contact all associated ID Providers and request that they also delete the specified attributes.

Method	DELETE
URL	http://.../API/identity/attribute
Content	[{ "attributeId":1 }, { "attributeId":2 }, …]

6.8 Risk Management

The ID Management module will display to the user a risk figure indicating the possibility that an ID Provider or a verifier may infer the values of unknown user attributes based on the known user attributes that the ID Provider maintains for this user. The risk indicator is separate for each unknown attribute and ID Provider permutation.

Note: Risk can be calculated only for attributes with known values i.e. attributes whose values are stored in the Identity Repository. For attribute values stored only in ID Providers it is impossible to determine their distribution and hence cannot calculate a risk factor.

6.8.1 RISK API

Risk indicators will be offered via a simple REST API

Method	GET

URL	http://.../API/identity/risk
Content	[{ “attributeId”:1, “IDProviders”: [{“ProviderId”:”x”}, {“ProviderId”:”y”}, …] }, { “attributeId”:1, “IDProviders”: [{“ProviderId”:”x”}, {“ProviderId”:”y”}, …] }, …]
Response	[{ “attributeId”:1, “IDProviders”: [{“ProviderId”:”x”, “risk”:0.8}, {“ProviderId”:”y”, “risk”:0.5}, …] }, { “attributeId”:1, “IDProviders”: [{“ProviderId”:”x”, “risk”:0.2}, {“ProviderId”:”y”, “risk”:0.6}, …] }, …]

6.8.2 Risk Calculation

A simplistic approach to risk calculation is to assume that risk is essentially the size of the population of user identities sharing similar attribute values over the entire population of user identities. The following example illustrates this approach.

An ID Provider knows that user “x” has the following attributes:

- {“professional domain” : ”IT”}
- {“locality” : “EU Citizen”}

In the entire population of 230,000 user identities in the ReCRED Identity Repository 220,800 of the "IT", "EU Citizen" users also have "English" as one of their spoken languages.

Therefore there's a 96% probability (i.e. risk that the ID Provider can guess) that user "x" also speaks English.

This approach is based on the assumption that an ID Provider has a large enough sample of user identities so that statistical properties are valid.

6.9 Supported functionalities

This subsection describes the functionalities that the Identity Management module offers to end-users and ID provider administrators.

End-users:

- View and manage identity attributes: The user is presented with a list of all the identity attributes supported by the IDC. For each attribute, the user can see the filled-in value (if not blank) and whether it has been verified or not. The user can also update the value of an attribute (or fill-in a blank attribute) which may or may not trigger a verification process. In that case, an update request is sent to all the IDPs that maintain the updated identity attributes, so that the user data remains synchronized across different IDPs. The user can also delete the value of an identity attribute and a delete request is sent to all the IDPs that maintain the deleted attribute.
- View identity attributes shared with Online services: The user can select an identity attribute and see which online services have access to that attribute. Alternatively, the user can select an online service and see which identity attributes are shared with that service. In both cases, risk calculation is executed, so that the user can see the probability with which an online service can infer the value for an attribute, even if it hasn't been explicitly shared with it.
- View and manage identity attributes maintained by ID Providers: The user can select an identity attribute and see a list with the IDPs that maintain that attribute, as well as the value of the attribute on each IDP. Alternatively, the user can select an IDP and see which identity attributes are maintained by that IDP and with which values. In the latter case, the user can select one or more identity attributes, in order to execute the following actions:
    - transfer the values of those identity attributes from the selected IDP to the ID Consolidator.
    - transfer the values of those identity attributes from the selected IDP to other IDPs also taking into account any specific rules defined by the selected IDP).
    - delete the values of those identity attributes from the selected IDP.

This functionality is very useful and can be used to transfer the trust and reputation built about a user from one Identity Provider to another Identity Provider without the need for the user to prove his trustworthiness again to the second Identity Provider. The need for transferring trust among Identity Providers has also been pinpointed by Venkatadri et al. who proposed a framework for strengthening weak online identities through inter-domain trust transfer [GOC16]. The proposed framework enables the transfer of trust built around a

Formatted: Not Highlight

Formatted: Normal, Indent: Left: 0.5", No bullets or numbering

Formatted: Not Highlight

user's identity between domains (IdPs) and can be implemented without significant implementation overheads.

o

- Create partial verifiable profiles: The user can select a subset of identity attributes and create a partially verifiable profile with those attributes. The IDC creates and presents a short URL that links to the created profile.
- List partial verifiable profiles: The user is presented with a list of all the partially verifiable profiles that he/she has created.
- View and manage partial verifiable profiles: The user can view extensive details regarding a selected verifiable profile (included attributes, short URL, creation date, etc.) and can preview the public verifiable profile. The user can also manage (add / remove) the identity attributes that are associated with the selected verifiable profile or delete the profile altogether.

ID Provider Administrators:

- Manage rules for identity attributes transfer: The IDP Administrator can define specific rules for the transfer of attributes maintained by the IDP to other IDPs. E.g. a bank administrator can create a rule so that the customers' bank accounts and loan data can be transferred only to other verified banks.

7 Online Identity Acquisition Module

The Online Identity Acquisition module obtains identity information from various ID Providers, such as online social networks like Facebook or Twitter, using Facebook connect or OpenID Connect/OAuth2. The main challenge in developing this module is to acquire user information integrated from a user's online accounts that user wishes to connect to the IDC. Once the IDC obtains access, it can collect the identity from the various ID Providers and it processes this information using the identity integration module to determine their validity.

7.1 Acquisition process

This module is responsible for horizontally binding of all online user accounts. 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. Attributes such as date of birth, location, education, occupation, etc. will be retrieved and will be stored to the Identity Repository of the IDC. Figure 4 shows how the Online Identity Acquisition module retrieves identity information from the social accounts of the users.

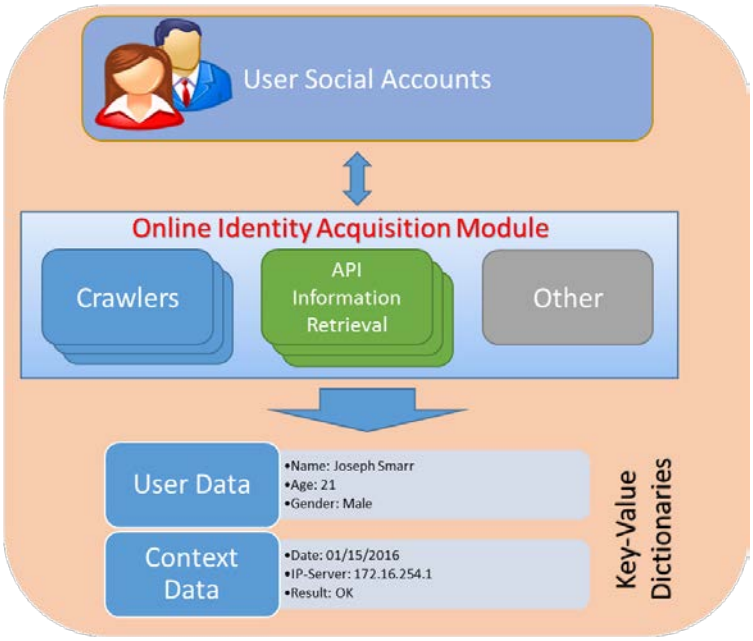


Figure 4 Online Identity Acquisition Module

7.2 Interaction with the Identity Repository

The process of collecting the identity information of a user from all her online accounts can be either for multiple online accounts or for a specific account upon user's request, depending on the authenticated service and the available information.

The first one is an automatic process, which is repeated periodically in order to update user's Identity Repository information from the ID providers. In the case of a specific account at user's request, the identity information is retrieved from the ID provider, once the user has been authenticated in the service.

The Identity Repository is updated when a user's identity attribute is obtained from an ID Provider either by updating directly the Repository, or by updating the Repository through the Identity Integration module that connects directly to the database.

7.3 Access to the Identity Providers

Access to id providers (online social networks for example) is integrated with the platform of each of the online social networks. In order to let users give their authorization to the IDC to access their online accounts' information, it is mandatory to include a social network login in the web interface of IDC. Once we provide such functionality, the users must be authenticated validating their credentials of the specified OSN and accepting the authorization request.

7.4 Authorization and Privacy management

The IDC asks the user for authorization to disclose personal information of their profile. The option or not to disclose information in each of the fields defined on the user's profile can be selected by a checkbox. If the user wants disclosure or not of his/her information about a specified field, as for example, information of his/her education, the user should decide what information is public or is hidden to the other users. By using this option the user can decide the privacy of his/her information.

7.5 Supported Functionalities

This subsection describes the functionalities that the ReCRED platform offers to end-users. All functionalities are related to the Online Identity Acquisition module.

End-users:

- Connect Online Account: The user can connect to the IDC an online account he owns, so that all his online accounts are eventually consolidated. An OAuth / OpenID Connect connection is established, in order for the user to be authenticated. The IDC issues a Proof of Account Ownership upon successful authentication to the service.
- 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 consent on the transfer of data from a selected connected online account to the IDC (through an OAuth / OpenID Connect connection).
- Disconnect Connected Account: The user can select a connected online account, in order to disconnect it from the IDC. Any data that has been transferred to the IDC must be deleted.
- Provide Proof of Account Ownership: The user can select a connected online account, in order to provide Proof of Account Ownership for that account. A cryptographic credential is issued, based on the Proof of Account Ownership, which can be used by the user in order to prove to others that he owns that specific account.

8 Physical Identity Acquisition Module

The Physical Identity Acquisition Module performs vertical identity binding. It is responsible for binding the real-world identity of a user to verifiable identity attributes. This module will be implemented as an application on smart trusted-computing-enabled devices and/or as a web application.

According to the reference architecture (Deliverable D2.3), the Physical Identity Acquisition Module consists of the Identity Acquisition Process and the Identity Verification Process. The Identity Acquisition Process involves the acquisition of physical characteristics of the users as well as physical identity documentation. This process uses trusted paths on the devices in order to securely capture images of a user and his physical characteristics (e.g., location). The identity acquisition module uses smart trusted-computing-enabled devices (e.g., mobile device) which will acquire the physical characteristics and identity documentation of the users. The devices also use trusted software paths in order to securely and verifiably capture through the device's camera images of a user and his documentation. Additionally, the physical characteristics of a user (such as location) will be extracted. All this information is then transferred into the identity verification process.

The Identity Verification Process uses crowdsourcing techniques and automated means in order to securely verify the acquired images and physical characteristics, while preserving the privacy of the users. Figure 5 shows how the physical identity acquisition process works and how a user declares and verifies attributes of his real-world documentation. The two processes of the Identity Acquisition Module are explained, in more detail, in subsequent sections.

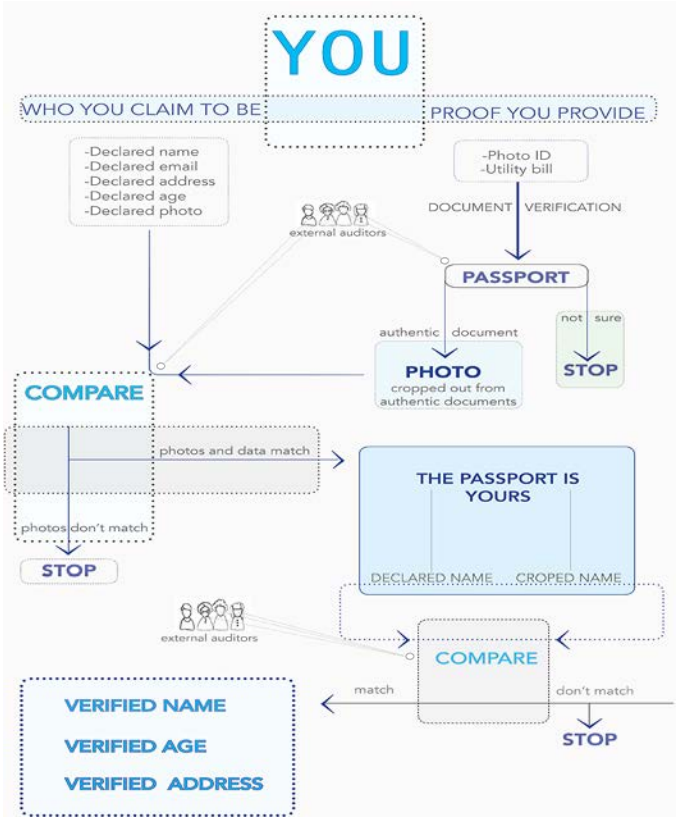


Figure 5. Physical Identity Acquisition and Verification process

8.1 Identity Acquisition Process

Initially, the Identity acquisition process requires from the user to declare their identity information like their name, surname, identity number, date of birth, etc. through a Web or mobile interface. The acquisition process also involves the acquirement of physical identity documentation of the user. The user has to capture, through his device's camera, images of their real-world identity and their face. Afterwards, the process enables the extraction and verification of the user's desired identity attributes (such as identity number, full name, date of birth). This is achieved, by requiring from the user to crop the corresponding parts from the captured image of his identity. Furthermore, this module enables the acquirement of additional physical characteristics of the users (e.g., his address). Some of those characteristics may require additional verification that will be described in the identity verification process.

The identity acquisition process periodically captures behavioral characteristics of the user (e.g., location) for verification purposes that will be explained further below.

8.2 Identity Verification Process

The Identity Verification process receives all the acquired identity information of the user from the identity acquisition process. This process uses peer-to-peer verification and automated means in order to securely verify identity information. Specifically, the process exploits the following well-known techniques:

- Face Detection and Recognition: this method is used for the verification of the captured images of the user. At first, face detection is used to verify whether a face is included in the acquired user photos, both documentation and face photos. As soon as a face is detected in the images, then we perform face recognition. Face recognition is used to check whether a user photo matches to his securely captured identity photo.
- Optical Character Recognition (OCR): this is performed on the acquired user photos. Specifically, it is performed to the cropped photos in order to first verify that they contain a text and then extract it from each photo. In the end, those texts are compared with the user's declared identity information.
- Peer-to-peer verification: crowdsourcing techniques are used to verify that the information on the acquired photos match the declared personal identity information and physical characteristics. The verification takes place via crowdsourcing it to other users of the platform. Confidentiality is preserved using watermarking and cropping techniques to ensure that nobody has access to reusable or forgeable photos of the users' physical documentations.

As soon as the verification has been completed, all the verified identity information is transformed into independent identity attributes that are stored into the Identity Repository of the Identity Consolidator platform.

8.3 Challenges

The Physical Identity Acquisition Module raises challenges both from the developer's and user's perspective. In such systems complexity, security and privacy increase the challenges for usability, which is very important for the adoption of the system. The system is more likely to be adopted if it is easy to use. However, due to the fact that we are dealing with sensitive information we should avoid compromising security and confidentiality.

The Physical Identity Acquisition Module has to collect a large amount of identity information from each user. Therefore, such a system should be developed in a way that it offers ease of use and at the same time earning the user's trust. In order to earn the trust of the user, it is important to preserve privacy and data confidentiality.

Another challenge, regarding the verification of the acquired identity information, is to deal with malicious users. In our case, a malicious user can be the provider of the identity information who aims to trick us and impersonate someone else. To deal with such users, we utilize enhanced state-of-the-art techniques to protect the ReCRED platform, and therefore 3rd parties.

8.4 Interaction with Identity Repository

The Physical Identity Acquisition Module interacts with the Identity Repository through a well-defined REST API (Storage API) so that obtained, verified identity attributes can be stored.

8.5 Supported Functionalities

This subsection describes the functionalities that the ReCRED platform offers to end-users. All functionalities are related to the Physical Identity Acquisition module.

End-users:

- Submit Identity Documents: The user can select to submit a specific document of a specific type (ID, Passport, Student Card, etc.). The acquisition process is triggered, through which the user must upload a photo of this document. After the document's acquisition is successfully completed, the identity verifications process is triggered as described above.
- List verified identity documents: The user is presented with a list of all the verified identity documents that they have previously submitted to the IDC.
- View & Manage verified identity documents: The user can select a submitted identity document, in order to see the data that is acquired for that document and the verification status. They also have the option to update the document (e.g. if they have changed his ID), triggering a new acquisition and verification process, or they can delete the submitted document altogether.
- View audit results/ Execute an audit: The user can see the audits that have been assigned to them and execute those audits. Additionally they can view the result of the audits that other users have performed about their submitted identity documents.

9 Storage API

This API defines a language- and platform-neutral protocol for Consumers to request, store and modify information of a user identity profile. This information contains identity attributes of the user and proofs of account ownership.

Part of this specification is based on the model proposed by Portable Contacts (http://portablecontacts.net/draft-spec.html#schema) and OpenSocial. This specification has been highly adapted to ReCRED requirements.

Note that this is a preliminary definition of the API, which will evolve during the execution of the project based on the modifications applied to the Storage Database.

9.1 Notation and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. Domain name examples use [RFC2606].

9.2 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 Fiel*d: 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.

9.3 Introduction

The ID Consolidation service is the second major component of the ReCRED architecture and it participates in most of the use cases of the ReCRED platform. It is responsible for horizontally binding the online identities of a user and vertically binding the real-world identities of a user to verifiable identity attributes. The ID consolidator exchanges verified identity attributes with the ID Providers using federated login protocols (such as OpenID Connect, OAuth).

The consolidator offers an identity management service which allows users to voluntarily submit identity attributes and proofs of account ownership. The user interacts with this service using his identity management application. This service contains the identity acquisition and verification components which are responsible for acquiring and verifying all the personal identity information of a user.

This API defines a language- and platform- neutral protocol for Consumers to request address book, profile, and friends-list information from Identity Providers. As a protocol, it is intended to be easy to understand and implement, either as an Identity Provider or Consumer, using any language or platform of choice. It is also intended to be implemented by both individuals and small services as well as large providers, in any case where a service contains data about who a user knows and wishes to make that information portable, under the user's control.

While there are currently standards for describing user info (such as vCard), these standards do not specify how to discover, access, and manipulate this information, and they do not capture the full range of information typically found in modern address book and social networking applications. Several large companies have also released their own non-standard APIs for accessing and interacting with contact information, increasing the burden on developers and Consumers who wish to support most or all Identity Providers. Nor do these APIs inform other providers as to how they should construct similar APIs. Thus ReCRED specification is an attempt to specify a complete, modern, and straightforward recipe for Identity Providers and Consumers of all sizes to make available and consume contact data in a standardized way.

9.3.1 Goals

The goal of this specification is to make it easier for developers to give their users a secure way to access the information they have built up all over the web. Specifically, we seek to create:

- A common access pattern and contact schema that any Online Service can implement
- Well-specified authorization and access rules
- Free and open source libraries in many languages for most popular platforms
- Community-sourced support, documentation, and collaborative tools
- Absolutely minimal complexity, with the lightest possible toolchain requirements for developers.

9.3.2 Approach

Our design is focused around ease of adoption, which means a few things:

- First, our emphasis is on simplicity of design and targeted use cases, keeping our scope intentionally narrow at the outset.
- Second, we're taking a modern approach to who-you-know data by unifying traditional contact information and social network data, in order to properly represent the current diversity of the social web ecosystem.
- Third, we're reusing existing standards wherever possible, including Portable Contacts, vCard, OpenSocial, XRDS-Simple, OAuth, etc.
- And lastly, we're designing something that should be easy for current identity providers to adopt. We started with a review of all the major existing contacts APIs and targeted common capabilities that they all share and provide. We believe this pragmatic balance is the best and quickest way to achieve our intended goal of widespread adoption.

9.3.3 Workflow Overview

A Consumer wishing to access a user's identity attributes via 3rd Party API must start with an Initial Identifier for the Identity Provider containing the user's data, usually provided by the user. In many cases, this may be the domain name of the ID Provider's web site, such as sample.site.org, but may be a more specific URL, such as the OpenID identifier of the user, if available. Consumers then perform Discovery on the Initial Identifier to determine where the ReCRED endpoint for this Identity Provider resides. If successful, the Consumer may then attempt to request information from that endpoint. If the endpoint contains private data, the Identity Provider will return an authorization challenge, and the Consumer must then guide the user through an appropriate authorization flow to obtain the credentials necessary to access this private data. Upon successful authorization, the Consumer may request data from the Portable Contacts endpoint using these authorization credentials. Whether accessing public or private data, Consumers may request a specific subset of the user's data using standard Query Parameters. Upon a successful request, the data is returned in the response, and the Consumer may then parse the response data and use it as desired. The following sections detail each of these steps.

9.4 Rest Services

ReCRED’s Storage API and 3rd Party 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 are intended to be used either by ReCRED modules or other third parties to access identity data hosted and managed by the ID Consolidator.

In general, for all REST Services:

- the 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,
- and 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 is required to communicate with a ReCRED server via an intermediary that restricts the use of certain standard and extension HTTP Methods (e.g. PUT, DELETE, 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 will 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.

9.5 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.

9.6 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.

9.6.1 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’.

9.6.2 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.

9.6.3 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.

9.6.4 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.

9.7 Query Invocations

All requests to the ID Provider 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 9.7.1. 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.

9.7.1 Authentication and Authorization

The data returned by a ID Consolidator endpoint MAY contain public data, or it MAY contain private data. If the data returned is public, no authentication or authorization is required. In most cases however, the data returned is not public, and ID Providers SHOULD ensure that the user has given prior consent, either explicitly or implicitly, for their information to be released by this API. Typically this is done by Consumers obtaining either Direct Authorization (with raw credentials, for example the user's username and password) or Delegated Authorization (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.

9.7.2 Delegated Authorization

ID Providers wishing to provide Delegated Authorization MUST support OpenID Connect and OAuth 2.0, and MAY also support additional Delegated Authorization mechanisms, if they choose.

OpenID Connect is a simple JSON/REST-based identity protocol built on top of the OAuth 2.0 and JWT (JSON Web Token) family of protocols. In particular, OpenID Connect is a simple identity layer on top of the OAuth 2.0 protocol, which allows computing clients to verify the identity of an end-user based on the authentication performed by an authorization server, as well as to obtain basic profile information about the end-user in an interoperable and REST-like manner.

9.7.2.1 Direct Authorization

ID Providers wishing to provide Direct Authorization MUST support HTTP Basic Access Authentication [RFC2617], and MAY 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. There are also convenient ways of providing and parsing HTTP Basic credentials in popular tools and libraries.

9.7.2.2 Available Authorization Methods

ID Providers that provide access to private data MAY choose not to support either Direct Authorization or Delegated Authorization, depending on their security requirements, but they MUST support either OAuth or HTTP Basic auth if they require any Authorization. When accessing a ReCRED endpoint, if sufficient authorization credentials are not provided, the Identity Provider 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 API is a protected resource and initiate the proper Authorization process needed to obtain the appropriate credentials. An example set of WWW-Authenticate headers returned by an Identity Provider that supports both OAuth and HTTP Basic might look like this. Note that the realm value is intended to be an opaque string that merely defines a shared label for resources that share the same authorization requirements.

WWW-Authenticate: OAuth realm="sample.site.org"
WWW-Authenticate: Basic realm="sample.site.org"

If ID Providers wish to make some response data publicly available and also provide additional info given the proper authorization credentials, they SHOULD provide a 200 OK response to requests without authorization with a WWW-Authenticate header in the response indicating that additional info is available via the specified authorization mechanisms.

9.7.3 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>/all Return all contact info (equivalent to providing no additional path info)
- /<path>/all/{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.

- /<path>/self Return contact info for the owner of this information, i.e. the user on whose behalf this request is being made. 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.

9.7.4 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 encourage the use of 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.
- 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.

9.7.5 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-URL].

Operator	Description	Example
Comparison Operators		
eq	Equal	`Address/City eq 'Redmond'`

ne	Not equal	Address/City ne 'London'
gt	Greater than	Price gt 20
ge	Greater than or equal	Price ge 10
lt	Less than	Price lt 20
le	Less than or equal	Price le 100
has	Has flags	Style has Sales.Color'Yellow'
Logical Operators		
and	Logical and	Price le 200 and Price gt 3.5
or	Logical or	Price le 3.5 or Price gt 200
not	Logical negation	not endswith(Description,'milk')
Arithmetic Operators		
add	Addition	Price add 5 gt 10
sub	Subtraction	Price sub 5 gt 10
mul	Multiplication	Price mul 2 gt 2000
div	Division	Price div 2 gt 4
mod	Modulo	Price mod 2 eq 0
Grouping Operators		
( )	Precedence grouping	(Price sub 5) gt 10

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-URL].

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	
contains	contains(CompanyName,'freds')
endswith	endswith(CompanyName,'Futterkiste')

startswith	startswith(CompanyName,'Alfr')
length	length(CompanyName) eq 19
indexof	indexof(CompanyName,'lfreds') eq 1
substring	substring(CompanyName,1) eq 'lfreds Futterkiste'
tolower	tolower(CompanyName) eq 'alfreds futterkiste'
toupper	toupper(CompanyName) eq 'ALFREDS FUTTERKISTE'
trim	trim(CompanyName) eq 'Alfreds Futterkiste'
concat	concat(concat(City,', '), Country) eq 'Berlin, Germany'
Date Functions	
year	year(BirthDate) eq 0
month	month(BirthDate) eq 12
day	day(StartTime) eq 8
hour	hour(StartTime) eq 1
minute	minute(StartTime) eq 0
second	second(StartTime) eq 0
fractionalseconds	second(StartTime) eq 0
date	date(StartTime) ne date(EndTime)
time	time(StartTime) le StartOfDay
totaloffsetminutes	totaloffsetminutes(StartTime) eq 60
now	StartTime ge now()
mindatetime	StartTime eq mindatetime()
maxdatetime	EndTime eq maxdatetime()
Math Functions	
round	round(Freight) eq 32
floor	floor(Freight) eq 32
ceiling	ceiling(Freight) eq 33
Type Functions	
cast	cast(ShipCountry,Edm.String)
isof	isof(NorthwindModel.Order)

`isof`	`isof(ShipCountry,Edm.String)`
Geo Functions	
`geo.distance`	`geo.distance(CurrentPosition,TargetPosition)`
`geo.length`	`geo.length(DirectRoute)`
`geo.intersects`	`geo.intersects(Position,TargetArea)`

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

```
{
  "id": "1",
  "displayName": "Chris Messina",
  "urls": [
    { "value": "http://factoryjoe.com/blog", "type": "blog" }
  ]
},
{
  "id": "2",
  "displayName": "Joseph Smarr",
  "emails": [
    { "value": "joseph@plaxo.com", "type": "work", "primary": "true" },
    { "value": "jsmarr@gmail.com", "type": "home" }
  ],
}
```

Given the parameters $filter=startswith(displayName, 'Chr'), only the first contact (with id=1) would match and be returned. However, with parameters $filter=length(displayName) gt 0, both contacts would be returned. Given the parameters $filter=contains(email,'plaxo.com'), only the second contact (with id=2) would match.

9.7.5.1 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.

9.7.5.2 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.

9.7.5.3 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.

9.7.5.4 *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). If any part of the request is declined, Providers MUST specify which part(s) of the request were declined in the response, using "sorted": false, "filtered": false, and/or "updatedSince": false as appropriate. For efficiency, Providers SHOULD omit these response fields if that part of the request was successfully performed, or if no such Query Parameter was specified in the request.

Note that since all of the filtering, sorting, and pagination operations are designed to reduce the amount of data returned, it is possible for Consumers to emulate these operations client-side when a Provider declines to perform them server-side. For instance, filtering can be accomplished by iterating through each entry returned and deleting those that do not match the filtering criteria. Thus Consumers can request these operations to be performed server-side, and Providers will honor them if possible, and otherwise indicate to Consumers that they need to be performed client-side, effectively "splitting the workload" while maintaining consistent semantics.

9.8 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. For more information, see chapters 8 to 10 in the OData Specification (http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-part1-protocol.html).

9.9 Error Codes

The Identity Provider MUST return 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.

For more information, see chapter 9 in the OData Specification (http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-part1-protocol.html).

9.10 Identity Schema

The Identity schema defines the containers and attributes used to deliver an individual user or a list of users as requested by the Consumer. The traditional identity info fields were taken directly from

the Portable Contact spec where possible. Even with some spelling changes, the field mappings remain equivalent, which means it should be easy to convert Portable Contacts data to and from ReCRED. By convention, Singular Fields have singular spelling (e.g. displayName) and plural fields have plural spelling (e.g. phoneNumbers) to make it easy to distinguish them.

Each contact returned MUST include the id and displayName fields with non-empty values, but all other fields are optional, and it is recognized that not all Identity Providers will be able to provide data for all the supported fields. The field list below is broad so that, for Identity Providers that do support any of these fields, there is a standard field name available.

9.11 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 containing 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.

9.11.1 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).

9.11.2 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.

name: The broken-out components and fully formatted version of the contact's real name, as described in Section 7.3.

nickname: The casual way to address this Contact in real life, e.g. "Bob" or "Bobby" instead of "Robert". This field SHOULD NOT be used to represent a user's username (e.g. jsmarr or daveman692); the latter should be represented by the preferredUsername field.

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.

preferredUsername: The preferred username of this contact on sites that ask for a username (e.g. jsmarr or daveman692). This field may be more useful for describing the owner (i.e. the value when /me/self is requested) than the user's contacts, e.g. Consumers MAY wish to use this value to pre-populate a username for this user when signing up for a new service.

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.

connected: Boolean value indicating whether the user and this Contact have established a bi-directionally asserted connection of some kind on the Identity Provider's service. The value MUST be either true or false. The value MUST be true if and only if there is at least one value for the relationship field, described below, and is thus intended as a summary value indicating that some

type of bi-directional relationship exists, for Consumers that aren't interested in the specific nature of that relationship. For traditional address books, in which a user stores information about other contacts without their explicit acknowledgment, or for services in which users choose to "follow" other users without requiring mutual consent, this value will always be false.

9.11.3 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 alternative contact phone numbers for an individual. Each is labeled, typed and a single number is marked as preferred:

```
"phoneNumbers": [
      {
        "value": "555-123-1234",
        "type": "home",
        "label": "Home"
      },
      {
        "value": "555-123-1235",
        "type": "work",
        "label": "Work",
        "primary": true
      },
      {
        "value": "555-123-1236",
        "type": "mobile",
        "label": "Mobile"
      }
    ]
```

When returning Plural Fields, Identity Providers SHOULD canonicalize the value returned, if appropriate (e.g. for 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.

ims: Instant messaging address for this Contact. No official canonicalization rules exist for all instant messaging addresses, but Identity Providers SHOULD remove all whitespace and convert the address to lowercase, if this is appropriate for the service this IM address is used for. Instead of the standard Canonical Values for type, this field defines the following Canonical Values to represent currently popular IM services: aim, gtalk, icq, xmpp, msn, skype, qq, and yahoo.

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.

tags: A user-defined category or label for this contact, e.g. "favorite" or "web20". These values SHOULD be case-insensitive, and there SHOULD NOT be multiple tags provided for a given contact that differ only in case. Note that this field is a Simple Field, meaning each instance consists only of a string value.

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.

9.11.4 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.

9.11.5 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.

9.11.6 organization Element

Describes a current or past organizational affiliation of this contact. Identity Providers that support only a single Company Name and Job Title field should represent them with a single organization element with name and title properties, respectively.

name: The name of the organization (e.g. company, school, or other organization). This field MUST have a non-empty value for each organization returned. This is the Primary Sub-Field for this field, for the purposes of sorting and filtering.

department: The department within this organization.

title: The job title or role within this organization.

type: The type of organization, with Canonical Values job and school.

startDate: The date this Contact joined this organization. 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 Contact left this organization or the role specified by title within this organization. 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.

location: The physical location of this organization. This may be a complete address, or an abbreviated location like "San Francisco".

description: A textual description of the role this Contact played in this organization. This field MAY contain newlines.

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.

9.11.7 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 happens 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.

9.11.8 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.

9.11.9 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.

9.11.10 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.

9.12 Example

Here is a sample request and response that illustrates much of Recred API. For simplicity, authorization information is not shown in the request.

Sample request (via HTTP GET):

```
http://sample.site.org/path/to/api/me/all?$skip=10&$top=10&$orderBy=displayName
```

Sample response (JSON):

```
{
 "startIndex": 10,
 "itemsPerPage": 10,
 "totalResults": 12,
 "entry": [
  {
   "id": "123",
   "displayName": "Minimal Contact"
  },
  {
   "id": "703887",
   "displayName": "Mork Hashimoto",
   "name": {
    "familyName": "Hashimoto",
    "givenName": "Mork"
   },
   "birthday": "0000-01-16",
   "gender": "male",
   "emails": [
    {
     "value": "mhashimoto-04@plaxo.com",
```

```
        "type": "work",
        "primary": "true"
      },
      {
        "value": "mhashimoto-04@plaxo.com",
        "type": "home"
      },
      {
        "value": "mhashimoto@plaxo.com",
        "type": "home"
      }
    ],
    "urls": [
      {
        "value": "http://www.seeyellow.com",
        "type": "work"
      },
      {
        "value": "http://www.angryalien.com",
        "type": "home"
      }
    ],
etc …
```


9.13 Compatibility with Standards

This version of the Recred API is based on the Portable Contacts specification and is partially compatible with it. This specification is currently (partially) compatible with the overlapping portion of the OpenSocial RESTful Protocol [OpenSocial].

The main divergence between Recred API and Portable Contacts and OpenSocial is the inclusion of OData for the Additional Path Information and Query Parameters. Recred 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

The aim of this chapter is to define the Identity Repository data model. Our goal is to define the conceptual model of the data associated to a profile account (contact or user). The design does not force a particular technology nor a database schema model. The data modeling techniques are basically implementation agnostic. Our intention is that this model could be implemented using traditional relational database (Oracle, MySQL, etc.) or NoSQL database (HBase, MonoDB, etc.). It does not directly deal with issues of performance, scalability, cluster distribution and management, etc.

The ReCRED servers manage normalized user profile data in standard format. This makes it easier to parse and use the profile data without having to learn about each provider’s data format.

10.1 Database design

The Identity Consolidator deals with user identity information. This data comes from different sources and ID providers. For instance, users could have a different username in Facebook or LinkedIn. The data base should be able to store and manage this diversity of data. It converts the data in one of the diverse data formats returned by the identity providers into a single standard format. We call this process normalization. In addition, a user could provide direct information to ReCRED platform. This information, when validated, should be marked as original or preferred information.

The system will store user information using,

1. A user account table. This table will store information about the user account. This table will include an unique identification of the user (primary key) and other system related information like update datetime, 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.
3. Several tables with (social) accounts information. Users have external accounts like Facebook, Instagram or Twitter. These tables store consolidated information as provided by this accounts or external identity providers.

The following figure represents this model:

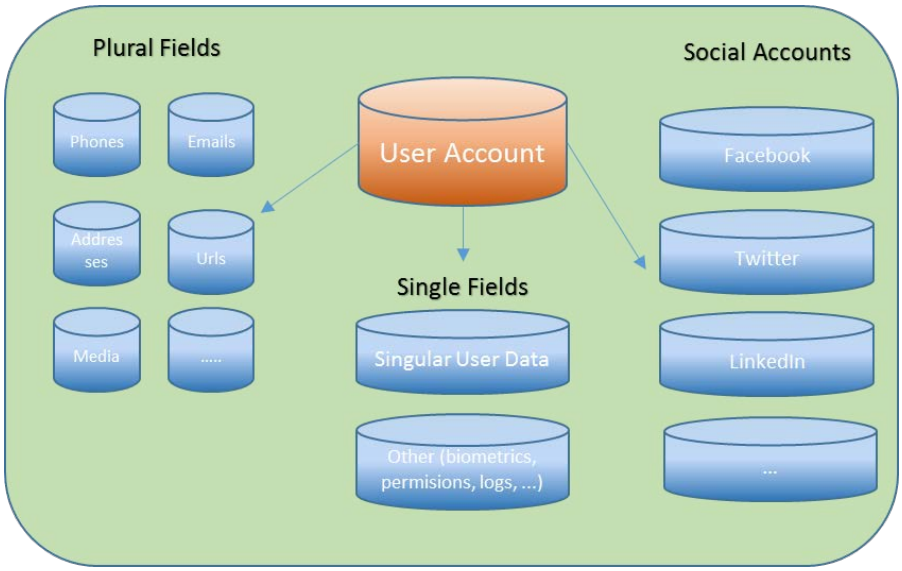


Figure 6 Storage model

10.2 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.
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 validated.	Guaranteed.
accessFailedCount	The number of failed access.	Guaranteed.
accountCanceled	A Boolean. True if the account has been canceled	Guaranteed.

10.3 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 documentation.	Available with user consent.
Nationality	The nationality of the user on his physical identity documentation.	Available with user consent.
expirationDate	The expiration date of the user’s physical identity	Available with

Key	Description	Availability
	documentation.	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.

10.4 Identity Providers Data Fields

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
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.
providerName	A human-readable name of the authentication provider that was used for this authentication. For instance, for well-known providers the integrator will use values such as “Google”, “Facebook”, and “MySpace”; “Other” is sent for other providers. New provider names are added over time. A particular case is “ReCRED”. This will be used to	Guaranteed.

Key	Description	Availability
	inform that the information comes directly from the ReCRED application and that the user has introduced it directly. This information should be considered as preferred information.	
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.
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.	Available from most providers, with user consent.
preferredUsername	The preferred username of this contact on sites that ask for a username.	Available from most providers, with user consent.
Name	A dictionary of name parts. See the name field section for details.	Available from most providers, with user consent. Yahoo! returns only a full name, not a first name or last name field.
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	Available from most providers, with user consent.

Key	Description	Availability
	these Canonical Values.	
Birthday	Date of birth in YYYY-MM-DD format. Year field may be 0000 if unavailable.	Available from most providers, with user consent.
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, 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.	Available from most providers, with user consent.
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.
verifiedEmail	A timestamp.	Available from Google, Facebook, Yahoo!, PayPal, Foursquare, and SalesForce.
URL	The URL of a webpage relating to this person.	Available from most providers, with user consent.
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.
address	See the address field section for details.	Available from most providers, with user consent.
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.

10.5 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.
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.
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.
middleName	The middle name(s) of this Contact (e.g. Robert given the full name Mr. Joseph Robert Smarr, Esq.).	Available, with user consent.
honorificPrefix	The honorific prefix(es) of this Contact, or "Title" in most Western languages (e.g. Mr. given the full name Mr. Joseph	Available, with user consent.

Key	Description	Availability
	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.).	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.

10.6 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	Guaranteed.

Key	Description	Availability
	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.
phoneNumber	A phone number at which the person may be reached.	Available, with user consent.
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 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.

10.7 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	Guaranteed.

Key	Description	Availability
	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"	
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.
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.

10.8 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.
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.
locality	The city or locality component.	Available, with user consent.
region	The state or region component.	Available, with user consent.
postalCode	The zipcode or postal code component.	Available, with user consent.
country	The country name component.	Available, with user consent.

Key	Description	Availability
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.

10.9 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	Guaranteed.

Key	Description	Availability
	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"	
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.
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.

10.10 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

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	An url with a WebPage, blog o similar element of the user.	Available, with user consent.
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.

10.11 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
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	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 information is supplied by the user.	Available, with 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	Guaranteed.

Key	Description	Availability
	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.

10.12 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 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.
status	The status of the verification process. Identity Provider SHOULD support at least the following values: initiated, onProcess, verified..	Guaranteed.

Key	Description	Availability
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.

10.12.1.1.1 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.

Key	Description	Availability
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.
dateAssigned	The datetime that the audit has been assigned to the auditor.	Guaranteed.

10.13 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.
IRS Certificate stating	IRS Certificate stating that the user does not have any overdue outstanding payments to the IRS.	Available, with user consent.

Key	Description	Availability
User Income	The user’s income of last fiscal year. Optional if it is included in the IRS Certificate.	Available, with user consent.
Social Security Certificate	Certificate from Social Security proving that the user is employed (self-employed or otherwise).	Available, with user consent.
Current Employer Certificate	Optional Certificate from current Employer verifying current employment status.	Available, with user consent.
NBIS Certificate	Certificate from the “National Bank Information System” (e.g. Tiresias) that the user is not blacklisted due to overdue outstanding loan payments to any national bank.	Available, with user consent.

10.14 Cryptographic Credential 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.

Key	Description	Availability
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.	

10.15 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.
primaryKey	The primary key of this entry in database.	Guaranteed.

Key	Description	Availability
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
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.

10.16 User 3rd Party Accounts Table

Key	Description	Availability
Identifier	The primary key of the user in the 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	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

10.17 Physical Identity Acquisition Embed API developers Table

Key	Description	Availability
Identifier	The primary key of this entry in the database.	Guaranteed.
developer_name	The name of the online service’s administrator that has created the developer account.	Guaranteed.
online_service_url	The URL address of the online service that the developer account has been created and that wants to use the Physical Identity Acquisition Embed API.	Guaranteed.
api_key	Alphanumeric randomly generated that is unique for each online service that uses the Physical Identity Acquisition Embed API.	Guaranteed

11 Identity Integration Module

This chapter describes the Identity Integration module. This module is responsible for verifying, standardizing and normalizing user information. It manages the spectrum of information elements that a user has on online social networks, blobs, portals or systems. A typical user has to deal with multiple online identities, which usually are stored in online social networks and managed independently of one another. Identity Integration is the process of providing a unified view of the data spread across different sources.

We have to recognize that existing online identity systems might be around for a long time, which leads us to find other solutions for resolving three key issues arising from managing data across disparate online identity systems:

1. **Duplication of information**. Identity information is often duplicated in multiple Identity Providers. For example, attributes such as addresses and phone numbers are often stored and managed in more than one system in an environment. When identity data is duplicated, it can easily get out of sync if updates are performed in one system but not the others.
2. **Lack of integration**. The complete view of a given user's attributes, credentials and privileges are often distributed across multiple identity providers, using heterogeneous protocols and technologies.
3. **Lack of veracity assessment and inference of missing attributes**. It is hard to determine the confidence in the veracity of identity information from a multitude of heterogeneous Identity sources.

The identity Integration module refers to the set of technologies that help ID Providers aggregate identity information from different identity sources (social networks or external applications), while reducing the complexity of data reconciliation, synchronization and integration. The efficiency of such a process is reliant on the effective semantic representation of the chosen data models, as well as the mapping relationships between the elements of the source data models.

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. The identity integration module can also infer missing ID attributes. The identity integration module is also responsible for assigning confidence scores for the veracity of the attributes and for labeling identity attributes based on their origin. For example, the Identity consolidator should be able to tell verifiers that user A is more than 18 years old with confidence 90%. Or alternatively the IDC 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 Integration process is a particular example of a more general concept: the data integration process. The concept of data integration has been studied by many research groups and from different perspectives.

The Identity Integrator module relies on the following principles:

1. An effective database solution should marry user data across disparate sources to provide a single, global view of a user. The ReCRED Identity Integrator follows this model. In our approach, we combine various data sources to form a uniform data repository capable of providing exact or maximally-contained answers.
2. Use a flexible and structured schema to store user profiles. User data needs structure in order to be actionable. For example, it can be a colossal challenge to query age ranges for users in a database when dates of birth are stored in inconsistent formats. The ReCRED database solution needs to normalize profile data that gets placed into the system, and enforce some structure to ensure that the data can be easily queried and applied for practical use.
3. Information should be validated and refreshed to keep the information stored accurate and up-to-date. The ReCRED approach should include information about the validation process and provides mechanisms to update the information as needed.

The systematic integration of the gathered data is hampered, because the underlying data stores of social networks are built with a focus on extension and flexibility, and thus, they often use so-called noSql databases. Such databases may store data in large tables without a traditional schema (e.g., HBase in Hadoop, used by Facebook and LinkedIn), or in schema-less multidimensional maps (e.g., Cassandra, used by Twitter). The resulting absence of explicit schema descriptions not only prevents the application of integration, but also hardens various data processing tasks, such as search, manipulation, optimization, translation, or evolution.

To integrate user profiles from different online social networks, the usual process consists of three major phases: data extraction, data transformation and storage. The following chart shows the flow of processes experienced by the user profile information since its capture. In this flow, the Identity

Integration Module plays an important role. Since the capture of information to the storage of standardized data, the data is processed and integrated to give a unified view.

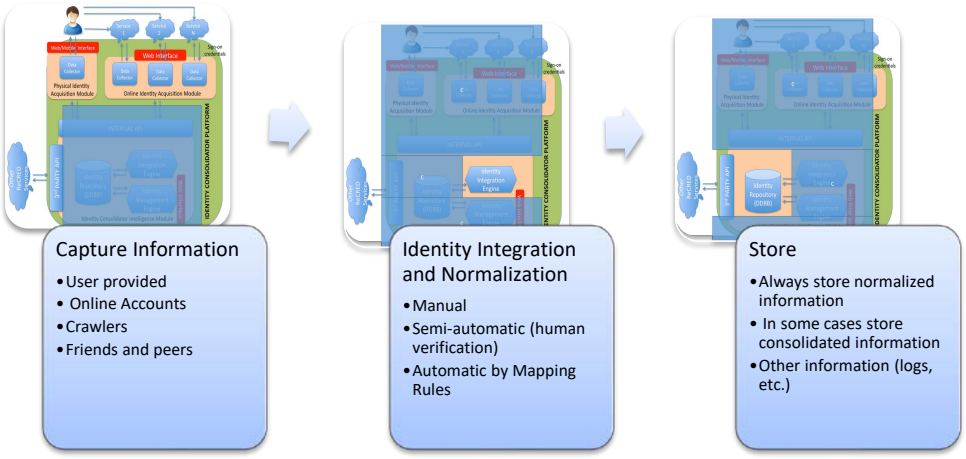


Figure 7 The flow of processes experienced by the user profile information since its capture

The Identity Integration and Normalization has two main sub-process: schema mapping and data transformation.

Identity user data may require any of the following basic transformation processes:

- **Canonization Process**. Some values should be canonicalized, e.g. http://josephsmarr.com/about/ instead of JOSEPHSMARR.COM/about/. Other examples are string-valued contact fields that represent common values in a canonical form, e.g. "male" and "female" for gender.
- **Aggregation or Disaggregation of information**. In some cases, the information is collected in separate fields and it is necessary to aggregate into a single information element, e.g. example, dates are obtained through its parts (day, month, and year). The opposite case is also possible. That is usually the case of names, e.g. a fullname can be divided into parts (first name, middle name, family name).
- **Mapping of Decoding**. When user data is represented by an external id, it could be necessary to translate it to an internal code (hash or string-value) that represent common values in a canonical form. An example of this is the Facebook id that should be mapped to ReCRED Id.
- **Sematic translation**. Some elements are too specific of the ID provider and cannot be stored directly into the Identity Repository, e.g. games information in Facebook. In that case, providers must decide whether to store the information and how. In that particular

example, providers could decide to store the information as a Media Item (like a photo or video) or ignore this piece of information.

- **Other ad-hoc transformations**. In general, ad-hoc transformations could be needed. For instance, statistical processing of logs or messages to form an approximation value, e.g. determine the city of a user based on its geolocation data.

All these transformation are specific of each provider and it is complex to give a general solution for every case. ID Providers should perform all this transformation when appropriate, but may delegate this process to the Identity Integration Module. In that case, the module will provide the ability to add specific plugins from each ID Provider.

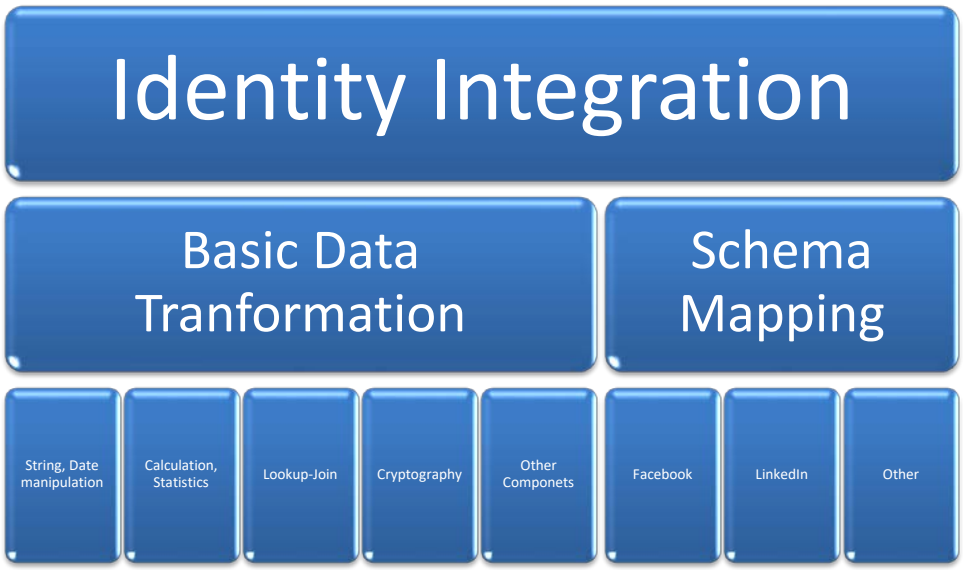


Figure 8 Identity integration

The architectural style we choose for implementing the Identity Integration Module is a service oriented, component based architecture. The reasons for using this style are usually:

1. Clean separation between interface and implementation. Allows us to even provide different implementations of the same service.
2. Loose coupling between components.
3. Easy integration. Both on the database level and when providing different Identity Providers.

This modular architecture allows three different modules or plugins depending on the phase of the information flow. The following figure 8 represents the overall process and some plugins examples.

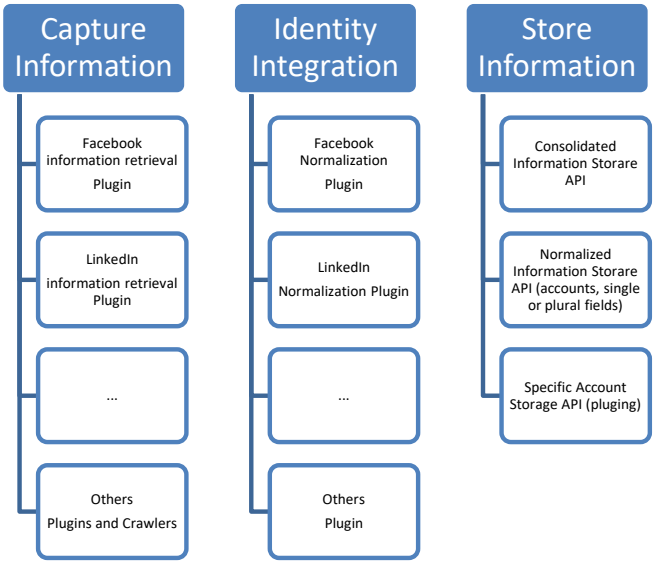


Figure 9. Overall process and plugin examples

At the capture Information level, we expect to have two different kinds of modules:

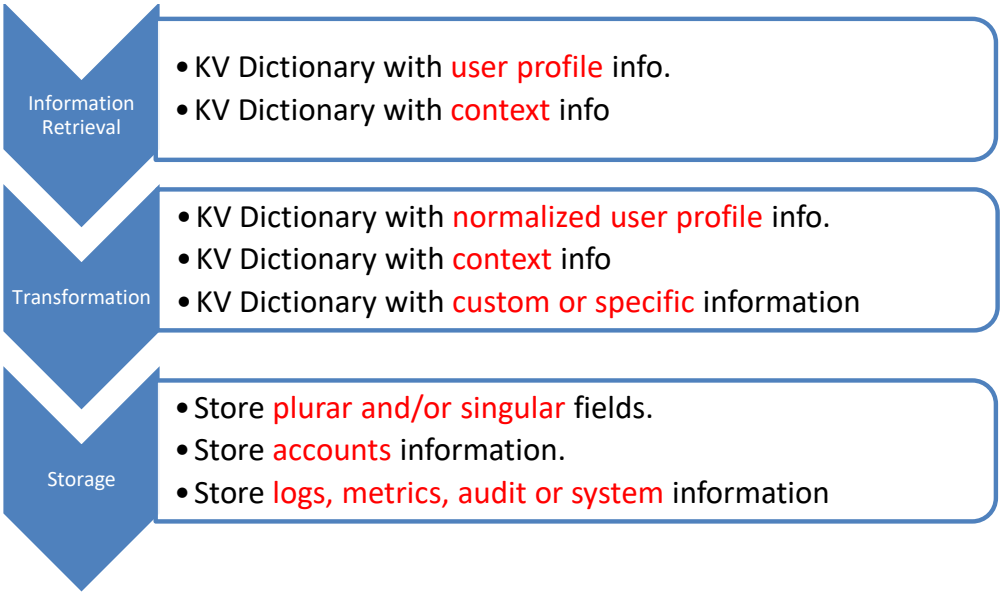
- Direct information retrieval. This module or plugin will extract information from social accounts directly from the user account, using the Identity Provider API, e.g. Facebook API.
- Crawler information retrieval. Some Identity Providers may not publish its internal database nor an API for its access. In some cases, it should be possible to retrieve information using crawling techniques. Although, most of the crawler modules have common parts (e.g. parsing HTML) each Identity Provider will have its own specific module.

In all these cases, information retrieval plugins should have a common set of functionalities like: start, stop, get the status of the process, etc. For each user profile, the result of the execution of the module will be a key-value dictionary with the user profile data. In addition to this data, each execution will also produce another key-value dictionary including information about the context of the retrieval process (date time, logs, errors, etc.). The advantage of using a Key-Value dictionary is two-fold. On the one hand, it is flexible and could be extending in the future. On the other hand, we expect to use JSON as external data representation, and a key-value data structure naturally fits with JSON.

At the Identity Integration level, we expect to have one single model for the construction of the plugin. Each plugin will take as input the two key-value dictionaries produced during the previous step (data profile information and context information). As a result, this plugin will produce three new key-value dictionaries. The first one will be a key-value dictionary with normalized information. Keys will be normalized using the schema of the Identity Repository, and values should be normalized and canonized. The second dictionary should be the context information. The third key-value dictionary will be a specific dictionary with profile data. This dictionary will include any data of

interest that does not fit in the normalized schema. The storage module will add this information on custom schemas for that particular Identity Provider. 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".

Finally, at the storage level, the Identity Repository will store all this information taking as input these three key-value dictionaries.



12 3rd Party API

The 3rd party API is used for communication purposes between the Identity Consolidator component of ReCRED architecture and other 3rd parties. Such parties include other ReCRED's components (e.g., Behavioral Authentication Authorities), Identity Providers and external online services/verifiers that want to interact with the Identity Consolidator platform or online services that want to embed ReCRED's physical identity verification module in order to offer an identity acquisition and verification functionality to their users.

The structure of the 3rd party API is similar with the one used for the Storage API (see Section 9). The subsections from "Rest Services" till "Plural Fields" also apply for the 3rd party API. The interested reader should refer to these subsections for the 3rd party API by visiting the Storage API's section.

12.1 High Level Operations

The 3rd party API offers, but is not limited to, the following operations:

- Verifiers should be able to interact with the Identity Consolidator component for the purpose of transferring trust between the services.
- Verifiers should be able to communicate with the Identity Consolidator component for the purpose of Two-Factor Authentication (2FA), if they desire. Specifically, the consolidator should be able to provide a reference to which Behavioral Authentication Authority (BAA) so that they can perform the additional authentication step.
- The BAA should be able to inform the Identity Consolidator component what behavioral aspects of each user they store. As a result, the consolidator always has a synchronized reference to the behavioral attributes that are stored in BAAs databases.
- The BAA should be able to inform the Identity Consolidator regarding the outcome (accept/reject) of an authentication attempt of a user to an online service. During this operation, the ID consolidator will check the defined securities policies and if are violated then it may lock some or all of user's account.
- An online service should be able to request the status of a user's account. Specifically, an online service should be able to query the Identity Consolidator which will reply with the account status (locked or unlocked).
- The ID consolidator should be able to issue cryptographic credentials directly to user's devices. Additionally, if the users desire so, the consolidator should be able to back up those issued credentials.
- Online services should be able to embed our Physical Identity Acquisition functionality to their web interface using the Physical Identity Acquisition Embed API, which is a part of the 3rd Party API.

12.2 Cryptographic Credentials Element

This element will be used when a user wants to issue and manage cryptographic credentials. During the issuance of a credential the user will be able to choose if the credential should be backed up at the ID consolidator. If the user chooses that, the Identity Consolidator will log all the necessary information about the issued credential.

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.

12.3 Behavioral Authentication Authorities Reference Element

Used when other ReCRED components need to communicate with the Identity Consolidator to access data regarding the BAAs reference. Specifically, when verifiers need to perform Two-Factor authentication they SHOULD be able to use this element to acquire information regarding the mapping of BAA, type of behavioral attribute and user. Furthermore, BAAs MUST make sure that the Identity Consolidator has always an updated reference regarding the behavioral attributes. For instance, if for whatever reason it is decided to change the BAA that hold the attributes of a specific user the BAA should use this element to update the reference in the Identity Consolidator.

baaInfo: The information of the Behavioral Authentication Authority that holds the specific behavioral profile of the user.

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.

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.

12.4 Account Element

This element is a limited version of the account's element that is described in the Storage API. The 3rd party API MUST be able to read all the information that is stored on this element but is able to modify only a subset of the attributes. The 3rd party API MUST be able to modify attributes in order to lock accounts of a user according to the BAAs recommendations. Below we describe the attributes and which ones can be modified by the 3rd party API.

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. **Read-only for 3rd party API**.

username: An alphanumeric user name, usually chosen by the user, e.g. "jsmarr". **Read-only for 3rd party API**.

userid: A user ID number, usually chosen automatically, and usually numeric but sometimes alphanumeric, e.g. "12345" or "1Z425A". **Read-only for 3rd party API**.

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. **Read-only for 3rd party API**.

Locked: Boolean reflecting whether the user account is locked for further actions (for instance, login in to that user account). This could happens for instance, if BAA informs to the ID Consolidator that user authentication has failed. **Can be modified by 3rd party API.**

Trusted: Boolean reflecting whether the user trusts the Identity Consolidator to store the actual data. **Can be modified by 3rd party API.**

12.5 Transfer attributes Element

This element is used to transfer verified identity attributes among online services. An example is when a user that has an account on Ebay wants to transfer his feedback to Amazon. These identity attributes MUST be verified in order to be able to be transferred among online services.

sourceDomain: The source domain of the identity attribute for this user account, e.g. "ebay.com". This is the Primary Sub-Field for this field, for the purposes of sorting and filtering.

destinationDomain: The domain which will receive the verified identity attribute for a user’s account e.g. “ amazon.com”.

attributeDescription: A representative description for the type of attribute.

attributeValue: The exact value of the attribute that will be transferred among the sourceDomain and the destinationDomain.

transferDatetime: The date and time of the attribute transfer. This information is filled automatically when the transfer is completed.

12.6 Authentication Outcomes Element

This element will be used so that BAAs can inform the Identity Consolidator regarding the outcomes of authentication attempts. This data will be used in conjunction with some pre-defined security policies to perform locks on some or all online accounts of a user.

Domain: The domain of the online service that the authentication attempt occurred.

Outcome: Boolean value. If the authentication attempt was successful then the value is set to true. Otherwise the value is set to false.

attemptDatetime: The date and time of the authentication attempt. This information is filled automatically when the BAA informs the Identity Consolidator.

12.7 Financial Information Element

This element is solely defined for the purpose of the loan origination pilot. Will be used when verifiers needs to access financial information regarding a user. This information should only be available upon user consent. Specifically, the user should be able to give consent for each attribute/certificate.

IRS certificate stating: IRS Certificate stating that the user does not have any overdue outstanding payments to the IRS.

User income: The user’s income of last fiscal year.

Social security certificate: Certificate from Social Security proving that the user is employed (self-employed or otherwise).

Current employer certificate: Certificate from current Employer verifying current employment status.

NBIS certificate: Certificate from the “National Bank Information System” (e.g. Tiresias) that the user is not blacklisted due to overdue outstanding loan payments to any national bank.

12.8 Physical Identity Acquisition Embed API

This Embed API is for any Online Service/Verifier that is interested in embedding our Physical Identity Acquisition module to their website. Using this API, an Online Service can perform physical identity acquisition and verification for its users.

The embedding procedure will be as easy all the other available embed APIs (e.g., Google Maps Embed API). First the administrator of the interested Online Service has to register as a developer in the Identity Consolidator web application in order to get a unique API key. Each API key will be bind with the URL of the Online Service. As soon as he has the API key for the Online Service, the only thing left to be done s copying the following HTML code into the webpage of the service.

```
<iframe src=”//www.recred.eu/idconsole/embed/physical-ID-acquisition
        &parameter_name=VALUE
        &key=YOUR_API_KEY”
</iframe>
```

All the acquired identity information will be stored in the Identity Repository of the Identity Consolidator platform and managed by the ReCRED platform.

13 Conclusion

The Identity Consolidation service is the central component in the ReCRED architecture. This deliverable defines the architecture and the basics of the Identity Consolidation service that is going to be implemented. A detailed description of all the modules that form the consolidator and the APIs that they use to communicate with each other is also provided.

Furthermore, in this deliverable we define the schema of the Identity Repository that is responsible to maintain most of the identity information of the users (e.g., identity attributes and cryptographic credentials).
Finally, an important part of the Identity Consolidation Platform is the 3rd Party API that we define in this deliverable. This API allows any external ReCRED entity, online services and Identity providers to communicate with the ID consolidation service.

Depending on further investigations, the ID consolidator will be revised and extended and we will describe the full ID Consolidation service in the next deliverable of this Work Package.

14 References

[GOC16] G. Venkatadri, O. Goga, C. Zhong, B. Viswanath, K. P. Gummadi, and N. Sastry. “Strengthening Weak Identities Through Inter-Domain Trust Transfer”, In Proceedings of WWW16, April 2016

[OAuth Core 1.0] OAuth Core Workgroup, “OAuth Core 1.0.”

[OAuth Discovery] Eran Hammer-Lahav, E., “OAuth Discovery 1.0.”

[OpenSearch] Clinton, D., “OpenSearch 1.1.”

[OpenSocial] Panzer, J., “OpenSocial RESTful Protocol Specification.”

[PortableContacts] Portable Contacts Specification, http://portablecontacts.net/draft-spec.html

[RFC2119] Bradner, B., “Key words for use in RFCs to Indicate Requirement Levels,” RFC 2119.

[RFC2425] Howes, T., “A MIME Content-Type for Directory Information,” RFC 2425.

[RFC2606] Eastlake, D. and A. Panitz, “Reserved Top Level DNS Names,” RFC 2606.

[RFC2616] Fielding, R., “Hypertext Transfer Protocol -- HTTP/1.1,” RFC 2616.

[RFC2617] Franks, J., “HTTP Authentication: Basic and Digest Access Authentication,” RFC 2617.

[XRDS-Simple] Hammer-Lahav, E., “XRDS-Simple 1.0.”

[OpenAM] OpenAM solution “https://forgerock.org/openam/”

Formatted: Font: Not Bold

Formatted: Font: Not Bold

Formatted: Font: Not Bold