Specific Targeted Research Projects (STReP)

SocIoTal

Creating a socially aware citizen-centric Internet of Things

FP7 Contract Number: 609112

WP1 – Socially-aware citizen centric architecture and community APIs

Deliverable report

Contractual date of delivery: 31/08/2016
Actual submission date: 31/08/2016

Deliverable ID:	**D1.3.3**
Deliverable Title:	**Final version of API specification**
Responsible beneficiary:	DNET
Contributing beneficiaries:	DNET, UC, CEA, CRS4, UNIS, UMU

Start Date of the Project: 1 September 2013 Duration: 36 Months

Revision: 1
Dissemination Level: Public

Document Information

Document ID: D1.3.3
Version: Final 1.0
Version Date: 31 August 2016
Authors: Nenad Gligoric, Srdjan Krco (DNET), Antonio Pintus, Alberto Serra (CRS4), Jorge Bernal Bernabé, Jose Luis Hernández (UMU), Ignacio Elicegui Maestro, Carmen López (UC), Colin O'Reilly, Niklas Palaghias (UNIS), Christine Hennebert, Etienne Gandrille, Levent Gurgen (CEA).
Security: **Public**

Approvals

	Name	Organization	Date	Visa
Project Management Team	K. MOESSNER	UNIS		
Internal Reviewer	Alberto Serra	CSR4	25/08/2016	

Document history

Revision	Date	Modification	Authors
Draft	18/17/2016	First Draft ToC	DNET
V0	3/08/2016	F2F enabler API	UNIS
V1	4/08/2016	Sociotal User Environment API	CRS4
V2	15/08/2016	Trust Manager API	DNET
V3	19/08/2016	Identity Manager, Group Manager, Authorization	UMU
V4	22/08/2016	Cotext and Community Manager	UC
Internal review	25/08/2016	Ready for internal review	CRS4

Content

Executive summary

This document specifies a set of APIs (Application Programming Interfaces) that exposes a complete set of required functionalities of SocIoTal system to application developers. Regularly updated online version of this document is available at the SocIoTal GitHub wiki [1]. The online version of the document has been used during API evaluation sessions, where developers used it to test and evaluate the APIs following a defined workflow, providing a valuable feedback for each API group, which produced this final updated version of the API specification. This document provides developers access to basic functionalities of the platform providing them means to completely reuse and employ lower-level infrastructure for their IoT deployment by using SocIoTal framework and APIs.

The deliverable is structured as follows: Section 1 proposes a semi-formal notation for the API specification description, which is carried out in Sections 2-10 grouped by envisioned modules of the SocIoTal architecture and those provided by a selected base platform as prerequisite for the SocIoTal.

Each API description includes the supported functionalities list (i.e., CRUD operations in case of a REST-based API), data types and their representation as well as returned data and parameters; moreover, they include the scope of the API, i.e.: internal or public, and the provider as the architecture module, WP or external tool/framework which expose the particular API. Section 11 concludes the API specification document.

Section 1 - Overview of SocIoTal API

Programming interfaces are a key element in a project with many components that aims at being open to third party applications developers. Thus, the function of programming interfaces is twofold; on one hand it allows to better define the interdependencies among different modules and work packages inside the SocIoTal project itself; on the other it allows to document with a concise set of entry points, the ways an external application could make use of SocIoTal functionalities.
The SocIoTal API are structured in three main macromodules grouped by their functionalities in the global architecture. These macro modules APIs are:

- **SocIoTal Context Manager API:** to provide access to context information and manage context entities (like devices).
- **Security & Privacy API:** to control the access to the system. It is a combination of different authorization, authentication and privacy control technologies and tools.
- **SocIoTal User Environment API:** to expose the User Environment Tools functionalities in the form of web APIs.

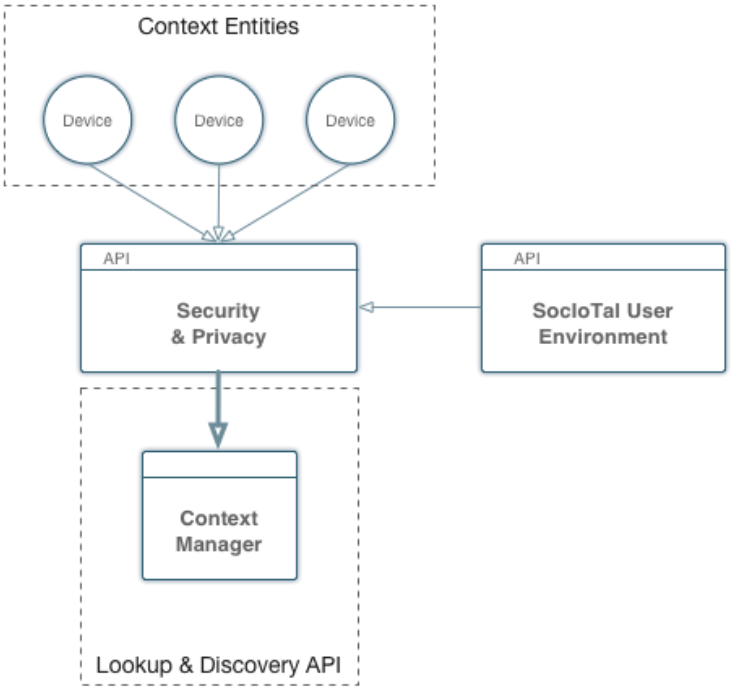

Figure 1 Overview of the API groups and their correlation

Section 2 - SocIoTal Context Manager

SocIoTal Context Manager (CM) [2] is part of the core blocks of the integrated SocIoTal Platform. As its main role, it provides (Figure 2) the SocIoTal entities directory and storage room for entities' context information so it is the context sink for context entities providers and the context source for context consumers. In order to manage the context information, the SocIoTal CM implements a complete NGSI9 and NGSI10 API rest interfaces, detailed (and updated) in SocIoTal Wiki plus an extended API with user functionalities in order to provide specific access and manage functionalities.

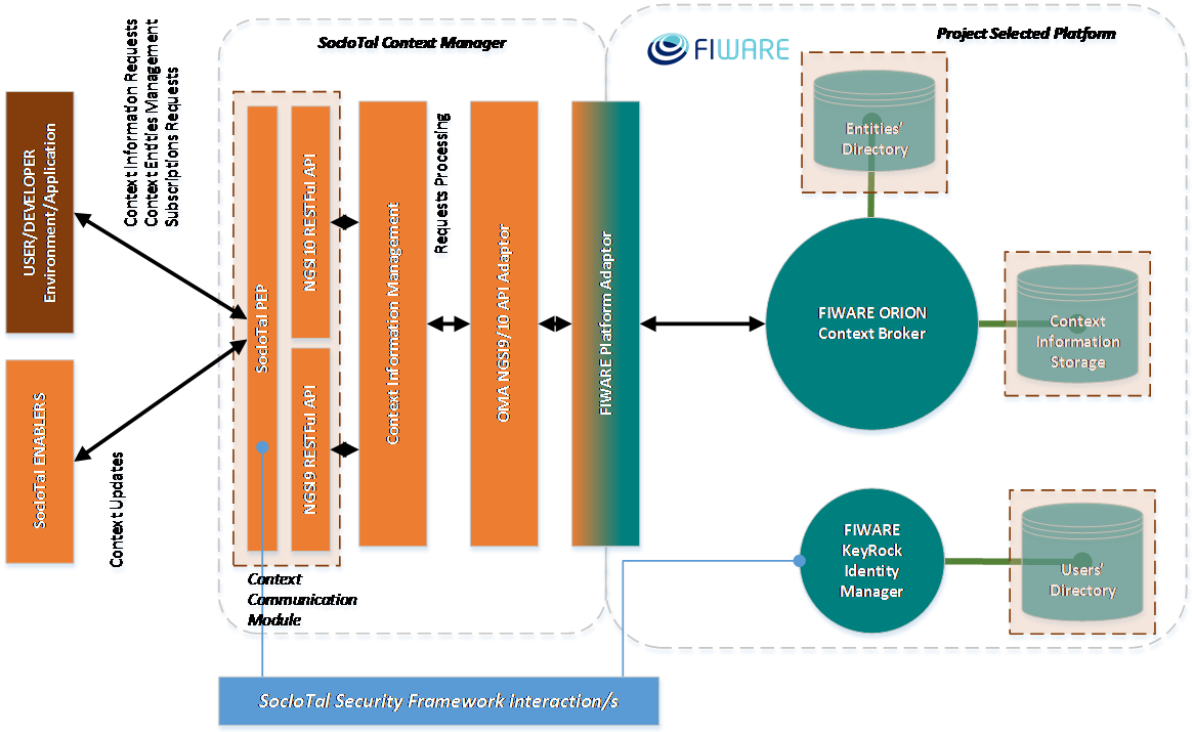

Figure 2 Current SocIoTal Context Manager implementation

Besides managing all context information related to entities, it also provides support to SocIoTal's Trust & Reputation mechanisms and SocIoTal's bubbles creation, keeping and supplying information related to scores and attributes. SocIoTal Context Manager is also integrated with the SocIoTal Security Framework by evaluating the Capability-Token and supports communities' management through the corresponding Community Token.

2.1 SocIoTal CM OMA NGSI9 API Implementation

This API [3] is intended to directly manage the SocIoTal's **Entities directory**. SocIoTal Context Manager NGSI9 API is represented by the URI:

/SocIoTal_CM_REST_V3/NGSI9_API/{method}

And provides this set of methods (further detailed and updated in SocIoTal's WiKi):

- **registerContext** [link]: registers new context entities or updates existing ones, including their attribute names and availability. This should be the first operation to be performed when creating a new Context Entity.

- **discoverContextAvailability** [link]: allows the synchronous discovery of context entities previously registered, providing access to their context structure (entities' ID, set of attributes and providing applications).
- **subscribeContextAvailability** [link]: allows the asynchronous discovery of the potential set of context entities belonging to an identified type and/or providing desired attributes. This is, by using this method, we will could receive (by POST method) an asynchronous notification (to the "reference" registered callback function) when a new entity is added to SocIoTal group (or an existing one changes), when a new device appears or when a new "AmbientTemperature" attribute is added (by a new entity or an existing one). Also, different options can be combined, in order to subscribe e.g. to devices registered within SocIoTal and providing "AmbientTemperature" information.
- **unsubscribeContextAvailability** [link]: just deletes an existing subscription to discover Context Information. The payload will only contain the "subscriptionId" returned by the "subscribeContextAvailability" method.
- **updateContextAvailabilitySubscription** [link]: this method updates an existing context availability subscription (performed using NGSI9 "subscribeContextAvailability). The request includes, in its payload, the "subscriptionId" element that identifies the subscription to modify and the set of elements to be updated.

2.2 SocIoTal CM OMA NGSI10 API Implementation

This API [4] is focused on directly **context information** management (get, update, discover and subscribe). SocIoTal CM's NGSI10 API is represented by the URI:

/SocIoTal_CM_REST_V3/NGSI10_API/{method}

And provides this set of methods (further detailed and updated in SocIoTal's WiKi):

- **updateContext** [link]: this is the most versatile method within SocIoTal CM API, providing 3 main functionalities:
  - APPEND → to create a new context entity and/or add new attributes to the context information provided
  - UPDATE → to replace existing attributes values (and metadata) of a given entity (or context element) with the current values
  - DELETE → removes an existing context entity (only at NGSI10 level. It won't remove the corresponding –if exists- NGSI9 instance)
- **queryContext** [link]: retrieval of Context Information. The requestor can specify restrictions on returned Context Information and perform context searching. The standard response includes all the attributes belonging to the given entities with their current values. If a list of attributes is included in the request (e.g. "AmbientTemperature") only those will be retrieved.
- **subscribeContext** [link]: this method provides asynchronous retrieval of specified context information. Based on NGSI, SocIoTal Context Manager will provide 2 different notification triggers:
  - ONTIMEINTERVAL: will send a notification when the time interval specified in the "notifyConditions" element is reached.
  - ONCHANGE: provides a notification every time a change in the specified context attributes happens
- **unsubscribeContext** [link]: this method deletes the context subscription pointed by the "subscriptionId" element of the payload.

- **updateContextSubscription** [link]: subscriptions can be updated using this method. The request includes a subscriptionId that identifies the subscription to modify, and the actual update payload, including the new values of the subscription the requester would like to change.

2.3 SocIoTal CM EXTENDED API Implementation

In addition to the pure NGSI9 and 10 interfaces, SocIoTal CM implements a set of extended functionalities to facilitate the use of its features. SocIoTal CM's Extended [5] methods API is represented by the URI:

/SocIoTal_CM_REST_V3/EXTENDED/{method}

- **createContextEntity** [6]: this method creates an entity and registers it through both NGSI9 and NGSI10 APIs. This way, a user can create an entity and publish its context information just in one call.
- **deleteContextEntity** [7]: this method deletes an entity in NGSI9 and/or NGSI10. The "id" will be the responsible to identify the entity to be deleted in NGSI10. On the other hand, the "registrationId" will point the entity to be deleted in NGSI9 and it can be obtained from the response when registering the entity through NGSI9 or using the extended method "createContextEntity".
- **queryContext by entityID** [8]: retrieval of Context Information. This method (not NGSI native) will retrieve the whole current context of the Entity ID passed as an http parameter.

Section 3 - SocIoTal Authorization manager

The SocIoTal access control system is designed as a combination of different authorization technologies and tools in order to enable a suitable solution for IoT environments. Such system is based on the use of XACML access control policies, which are employed to generate authorization credentials in the form of capability tokens. Then, such tokens are used by user to get access to entities, which are defined in the SocIoTal Context Manager. Additionally, it should be noticed that the resulting system has been integrated with the SocIoTal Identity Manager to get and evaluate capability tokens. Furthermore, it has been adapted to get a fine-grained authorization approach following NGSI-9/10 notation. The full specification for this component can be found on the wiki [9].

The SocIoTal Authorization scenario consists of the following main entities:

- **Capability Client**: performs requests to the Capability Manager to obtain capability tokens, which are used to perform actions over entities that are registered with the Context Manager.
- **Capability Evaluator**: responsible for evaluating capability tokens against a specific access request containing a capability token.
- **Capability Manager**: a server accepting requests for capability tokens generation. Additionally, this entity acts as a client requesting authorization decisions to the Policy Decision Point.
- **Policy Decision Point (PDP)**: It is a server that accepts XACML requests to make authorization decisions. The PDP is contacted by the Capability Manager before generating a capability token for the Capability Client.
- **Policy Administration Point (PAP)**: It is a web application responsible for defining and managing the access control policies. It provides the functionality so users can define XACML policies in a user-friendly way. The PAP has a GUI to facilitate the generation of policies.

3.1 Capability Client API

The *Capability Client* library [9] is a HTTPS client making requests to the *Capability Manager* to obtain capability tokens, which are used to get access to entities registered in the SocIoTal Context Manager.

3.2 Capability Evaluator API

This library [9] is intended to evaluate capability tokens against a specific access request. Indeed, it is employed by the SocIoTal Context Manager to evaluate capability tokens when clients try to perform a certain NGSI action over a specific entity.

The set of described methods has been developed and integrated in the scope of the main project scenarios. However, it should be noticed that these libraries have been also implemented by using CoAP-DTLS [10] protocols (instead HTTPS). Furthermore, an adapted version of the Capability Client API has been integrated to be employed by the Web User Environment that is enabled to get capability tokens on behalf of users.

Section 4 - SocIoTal Communities Manager

The concept of the SocIoTal Community is based on the types of relationships among smart objects and users. In this case, the usual driver of communities is the common interest relationship, putting together users and information related to a similar target or application. This way, SocIoTal defines a community as a secure cooperation between different producer users, making entities (devices, services, events, resources etc.) available to selected consumer users (community members), to achieving a common objective. SocIoTal Communities Manager [11] will provide a standard REST API to create and manage communities within the SocIoTal framework. From a high level point of view, this SocIoTal component will manage:

- **Users**: represent user entities within the SocIoTal Communities framework, keeping the credentials (names, passwords, roles and tokens) needed to be authenticated within the Communities Manager and the SocIoTal framework plus other extra information relevant for other SocIoTal applications (address, nicknames, organization, etc.).
- **Communities**: seen as groups of users and resources identified by a name and an id plus a description.
- **Domains**: as a group of isolated communities and users. Every user and community will be linked to only one domain, with the possibility to be duplicated under other different one.
- **Tokens**: provided by the Communities Manager and requested by a user, the Community-Tokens, represented by their corresponding UUID (Universal Unique Identifier) will identify the requestor user (previously registered within the Communities Manager), the community and the domain it belongs to, providing also extra related information such as the role the requestor has and its validity (as well as its expiring date).

The Communities Manager provides mechanisms to register users, manage communities and request and validate Community-Tokens. These methods collection is published and updated in the SocIoTal Wiki [11].

4.1 Users API

This set of methods groups the functionalities related to the users' creation and management. This set of functionalities are directly linked to SocIoTal IdM, using the

provided JAVA API and its SCIM compliant interface. This way, all SocIoTal components link to the same shared user's directory.

- **Register a user** [12]: this functionality creates a user and adds it to the SocIoTal IdM (Identity Manager). At the same time, a private community (for this new user purposes) is set up. When a user registers a new resource, it will be added by default to their private community and nobody will have access to the private community but the creator (and those who the creator provides access). The users and the private communities will be defined by names and ids.
- **Get user info by ID/Name** [13][14]allows the requester to get information about a user by providing their User-id/Username. Only users from the same community can get information about other users. The community will be pointed by the Community-Token provided in the request, so they will use the Community-Token of the same community they are requesting info.
- **Delete a user** [15]: this functionality deletes a user from the User Directory plus their private community. Only the correspondent user can remove itself.

4.2 Communities API

This API contains the methods to create (and manage) communities and assign users and roles. It is implemented through the Keystone V3 API provided by the Keyrock instance of SocIoTal, shared also by the SocIoTal IdM. This way, SocIoTal IdM will have access also to the domains/communities schema created by the Communities Manager.

- **Create a community** [16]: creates a community and assigns the "owner" role to the user which has request the petition. The tool will extract the information about who has made the request from the Community-Token attached in the headers.
- **List communities** [17]: provides a list of the communities of a domain (by default the SocIoTal domain). The user will be identified only with a Community-Token provided by the platform (it doesn't need to be related to a concrete community).
- **Get Community Info** [18]: this function allows obtaining further information about a concrete community by the community Name or the community ID. Within the body it will be defined the community name or the communityID, and within the headers there will be included a Community-Token from a community within the same domain of the requested community.
- **List available roles** [19]: list a set of roles to be applied to users when adding them to a community. The user will be identified only with a Community-Token provided by the platform (it doesn't need to be related to a concrete community.
- **Assign a role** [20]: this functionality adds a user to a community by assigning them the role to play within the community. Only an owner of the community is able to add a user to the community, and they will use their Community-Token of the community as a header of the request.
- **List users** [21]: lists users within a community and their corresponding roles. The community will be specified by the Community-Token heading the request, since this operation only can be done by a user belonging to the community.
- **Revoke a role** [22]: removes a user from a community. This is an action that can be only performed by the owner of the community (a role "owner" belonging to the referenced community is required), so the Community-Token for the heading will be the one from the owner (or an owner) of the community.
- **autoRemove from a community** [23]: the revoke role function allows the owner of a community to remove a user from the community, but if users want to remove themselves from a community they will be able to autoremove themselves from the community through this function. You must have in mind that if the user to be removed is the owner of the community, the community will remain without an owner, so most of the actions will not be performed by other users (unless the community

had more than one owner). The request doesn't require a body and the user and the community is obtained by the Community-Token of the user within the correspondent community.

- **Delete a community** [24]: this functionality deletes a community. Only an owner of the community is able to delete de community. The community to be deleted is pointed by adding the communityId to the URL.
- **List my communities** [25]: list the communities to which the users belongs to. The user will be identified by one of its Community-Token.

4.3 Community-Tokens API

This includes the request and retrieve community-tokens operations and the community-tokens validation. It is implemented over the same token schema SocIoTal IdM uses to validate users by user/password mechanism, so community-token can be also used to authenticate users.

- **Token request** [26]: the user will need a Community-Token for each community that they belong (including their private one). Every token request including a "communityName" will provide a token valid (if the user belongs to the community and it is properly authenticated) ONLY for the community pointed. A "communityName" is unique ONLY for a given domain, pointed here by the "domainName" element. If the user belongs to several communities, a different token, for each community should be requested. If no "communityName" is provided, the Community-Token retrieved will only authenticate the user and won't be valid to access any community. This request can be done by "username" [26], by "userID" [26] and extending a previous requested token (by tokenUUID) [26]. In any case, the Communities Manager will retrieve all the information linked to the requested token.
- **Simple token request** [27]: this call is a short version of the calls presented before and it has been designed for devices that want to avoid processing the response. The required payload is the same than the call presented before, but the response will return only the "Community-Token" as plain text, instead all the associated token information.
- **Validate a token** [28]: this functionality returns the information of the token provided as part of the URL. This action is performed by (and restricted to) those architectural components that need to validate if a user has the proper right to perform an action. The components will check and get the given Community-Token information (perform the validation of a token) whenever a user wants to perform and action in the architecture. The "Community-Token" parameter represents the UUID of the token (what we call Community-Token itself e.g. e515bb7ba9724270bec11a8b68de0fbf) and, in order to perform this call, the platform admin-token is required. In future releases, a special token for "apps" and developers will be included in order this Community-Token can be used by other components/apps outside of SocIoTal Integrated Platform.

Section 5 - SocIoTal Identity Manager

The Identity Management (IdM) system follows a **claim-based** approach with Attribute Based Credentials (ABC). The IdM relies on the **Idemix** cryptographic library from IBM, providing additional means to deal with IoT scenarios where consumers and providers' can be not only traditional computers, but also smart objects (e.g. smartphones). The IdM endows users and smart objects with means to control and manage their private data in their smartphone, defining partial identities over their whole identity, which is derived from the credential obtained from de Issuer. The usage of partial identities ensures a privacy-preserving solution with minimal disclosure of personal information. Unlike in traditional IdMs, the subject smart object is not redirected to its online Identity Provider (IdP) during the

transaction, so that the IdP is not involved when the target device verifies the smart object's attributes.

The SocIoTal IdM has been integrated with Fi-Ware **Keyrock IdM** to support traditional and basic, but necessary, Identity management operations in scenarios where claim-based accesses are not needed. Full documentation of IdM can be found online [29].
Keyrock IdM provides mechanisms such as secure and private authentication from users to devices, networks and services, authorization & trust management, user profile management, privacy-preserving disposition of personal data, Single Sign-On (SSO) to service domains and Identity Federation towards applications.
The SocIoTal IdM is composed of five main components:

- **SocIoTal-IdM-Android-Client:** It is an android application that allows obtaining Idemix credentials from the Issuer server. It also allows interact with the Verifier server which can validate the partial identity derived from the credential.
- **SocIoTal-Issuer-Server:** It is a web application implemented with Java servlets and XML-RPC which allows generating Idemix credentials for clients. Communications are done by https. The client must be authenticated against the Issuer using a valid certificate. The Issuer also support the verification functionality.
- **SocIoTal-Verifier-Server:** It is a web application, also implemented with Java servlets and XML-RPC, which is able to validate partial identities presented by the client application.
- **SocIoTal-IdM-Enabled-Capability Manager:** The IdM-Enabled-Capability-Manager is a web application that allows users to obtain capability tokens using their partial identities. In other words, it allows authenticating and demonstrating their attributes by means of Idemix proofs of having a valid credential issued by the Issuer.
- **SocIoTal IdM KeyRock Client:** The SocIoTal IdM KeyRock Client Java library provides a basic API for identity management by implementing a client to interact with the FIWARE KeyRock server. To carry out such communication, the SCIM 2.0 and Identity API v3 interfaces provided by this IdM are used.

The following figure shows the main components and their interactions.

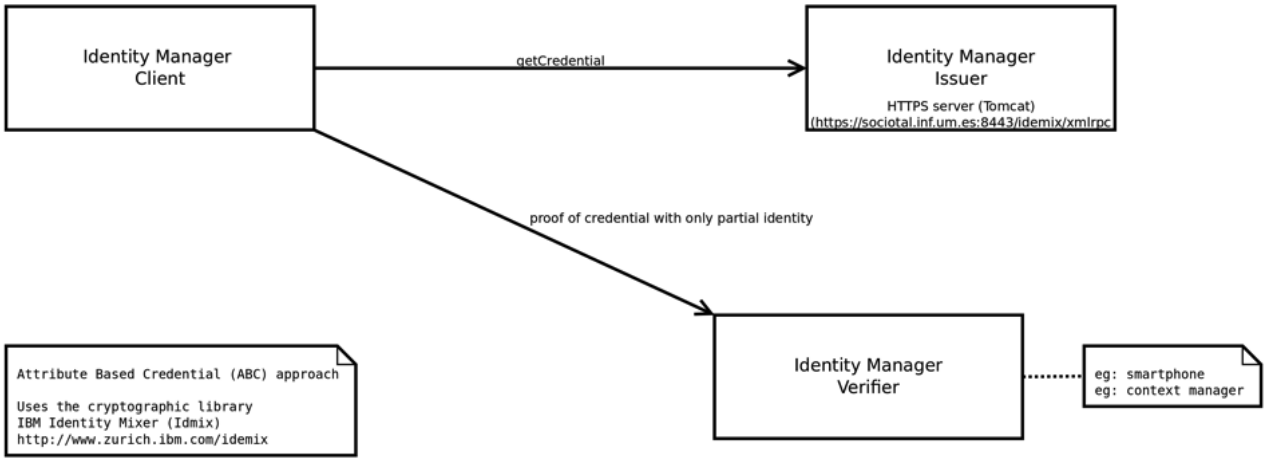

Figure 3 Identity Manager components and interactions

5.1 Client API

The Android app "SocIoTalIdMAndroidClient.apk" as well as the android library can be found in the SocIoTal repository [1]:

This first proof-of-concept android app provides three main functionalities:

- Obtain an Idemix credential from the Issuer.
- Validate a partial identity (Idemix proof) derived from the obtained Idemix credential
- Obtain a authz capability token from the capability manager using the partial identity

The application requires a proper certificate signed by the CA. The certificate is used to authenticate the user against the Issuer Server. The communication with the Issuer is done by HTTPs. The app already includes a valid certificate for testing purposes. Notice that the Issuer web container (i.e. Apache Tomcat) is configured to trust the certificate signed by the CA.

Idemix cryptographic library requires the recipient (the client user) to share the same Idemix system parameters and group parameters with the Issuer Server. These parameters are public and accessible by two configuration files. The android client is already configured to read from these two files to perform the cryptographic operations required in the issuance and proving Idemix operations.

SYSTEM_PARAM_LOCATION = "http://host:port/idemix/files/sp.xml";
GROUP_PARAM_LOCATION = "http://host:port/idemix/files/gp.xml";

The issuer public key location is available at: ISSUER_PUBLIC_KEY_LOCATION = "http://host:port/idemix/files/ipk.xml";

The SocIoTal IdM relies on the Idemix cryptographic library v2.3.0. For further information about Idemix and the library, please, refer to: http://www.zurich.ibm.com/idemix/downloads.html

Idemix cryptographic library requires the recipient (the client user) to share the same Idemix system parameters and group parameters with the Issuer Server. These parameters are public and accessible by two configuration files. The SocIoTal IdM relies on the Idemix cryptographic library v2.3.0. For further information about Idemix and the library, please, refer to [30].

Developers can take advantage of the java API to provide to their android applications a way of work with partial identities. The API provides two main classes to deal with the main functionalities. The IdemixManager and the CapabilityManager.

5.2 Issuer API

The Issuer server is a web application implemented with Java servlets and XML-RPC which allows generating Idemix credentials for clients. Communications are done by https. The Issuer requires of a Web container like Apache Tomcat. The Tomcat must be configured to trust the CA. The Issuer server is release as a java war application that has to be deployed in the /webapps folder of the Tomcat server for its installation.

The Issuer requires client authentication with a valid certificate signed by the CA. Idemix provides an XML schema to define the structure of the credential to be issued by the Issuer.

During the issuance protocol, the client must specify the particular credential specification that wants to obtain from the Issuer. That is, the identity attributes that are going to be included in the credential issued by the Issuer. The SocIoTal Issuer provides a predefined set of credential specifications, which are available at http:// plaltform.sociotal.eu:8443/idemix/files/.

The InitiateIssuanceCredential method below shows an Idemix credential structure example, with 13 identity attributes, corresponding to the 13 attributes supported by our Keyrock instance.

5.3 Verifier API

The verifier server is a web application based on XML-RPC that allows validating partial identities, i.e. validate Idemix proofs related to an Idemix credential obtained from the Issuer server.

Idemix requires agreeing a particular proof specification, between the client and the Verifier. In other words, both entities must agree on a specific structure of the partial identity, with the attributes from the full credential that are going to be shown and verified in the verification process.

5.4 SocIoTal IdM KeyRock Client

This Java library [29] provides a basic API for identity management by implementing a Admin client to interact directly with the FIWARE KeyRock server. To carry out such communication, the SCIM 2.0 and Identity API v3 interfaces provided by this IdM are used.
This means that this API is not aimed for final users, as the admin token required to interact with the library is supposed to be kept secretly by the administrator. Instead, final users can interact through the keyrock for authentication and attributes retrieval through the IdM client API explained in section 5.1.

The library offers the following functionality:

- Add a new entity (user or smart object) to IdM (SCIM 2.0 interface)
- Delete a registered entity in the IdM (SCIM 2.0 interface)
- Update the attributes of a registered entity in the IdM (SCIM 2.0 interface)
- See the attributes of a registered entity in the IdM (SCIM 2.0 interface)
- Authenticate a registered entity in the IdM (SCIM 2.0 and Identity API v3 interfaces)
- Validate a authentication of registered entity in the IdM (Identity API v3 interface)

Next, the implementation details and some aspects of interest on the deployed KeyRock instance are shown. A complete documentation is available online on the wiki [29].

Section 6 - SocIoTal Group Manager

The SocIoTal Group Manager [31] component is based on the CP-ABE cryptographic as a flexible scheme to enable a secure group data sharing mechanism. The functionality of this component is mainly split into two entities: the Group Manager Server or Attribute Authority (AA), and the Group Manager Client. The Group Manager client API allows obtaining cryptographic material, encrypt, decrypt as well as sharing encrypted information through the Context Manger. This library has been integrated to manage data sharing procedures within SocIoTal bubbles. A complete documentation is available online on the wiki [31].

Section 7 - SocIoTal Trust Manager

A Trust Manager [32] is a REST webservice with a logic that utilises set of different rules to build a reputation score. Generic model for rules enables mapping between provided JSON format and relational database for mining and extraction of rules previously added over a registration API. The crucial component that Trust Manager utilizes to continuously maintain the updated version of score in respect to last attribute value changes is the SocIoTal Context Manager. A complete documentation is available online .

Section 8 - SocIoTal F2F Interaction

The SocIoTal F2F Interaction consitutes the interface to retrieve information about users' behavioural traits including social interactions and social relation. In order to access this information there is a need to utilise an access_token in the access URL. The token could be retrieved from the Authorization manager presented in Section 3 - . The token allows the component to communicate with the Context Manager in order to create the context entity but also to update and query the Context Manager for the context entity of the F2F enabler. The information are represented through JSON structure as payload to the REST communication mean.

Following are the specification for creating-reading-updating-deleting context related to the F2F enabler. A complete documentation for the F2F api is available online on the wiki [33].

Section 9 - Sociotal User Environment

The SocIoTal User Environment [34] provides an API where every endpoint is authenticated by token through an *access_token* URL parameter. That token is unique, it is associated to a particular user and can be obtained from the user profile page in the Web User Environment website. Access token is represented by the AUTH_TOKEN placeholder in the API documentation that follows.

channelId is the id of the Channel. You can find it in the url of the channel details page.

For example: **<uri:port> /channels/<CHANNEL_ID>**

The supported format for entities representations is only JSON, for every endpoint.

Although the API endpoints can be reached also through HTTP protocol, **using HTTPS protocol is the recommended way**. Next versions of API could deprecate HTTP.

9.1 Channel Management API

A channel is an API connector that allows interfacing the UserEnv to a SocIoTal API (eg. A SocIoTal Context Manager) or to a 3rd party device API (eg. Xively). Each category of device or service to be managed in the UserEnv will have a corresponding Channel.

The Channel Management API provides the following set of methods (further detailed and updated in SocIoTal's WiKi [35]):

- **createChannel** [36]: creates a new Channel inside the User Environment. A Channel is created following a set of templates. One of available templates is the SocIoTal Channel.
- **updateChannel** [37]: updates a previously created channel inside the User Environment.
- **deleteChannel** [38]: removes a user's Channel from the User Environment.
- **getChannelList** [39]: returns the list of all the Channels belonging to a user.
- **getChannelInformation** [40]: returns a Channel data representation.
- **getChannelData** [41]: returns data stored in the channel or a list of channels belonging to a user.

9.2 Composition API

These APIs allow a user to create compositions between triggers and actions of different Channels, in other words it allows to connect to Channels. A **trigger**, registered in a business logic composition, is a condition to be evaluated on the incoming data on the source Channel. An **action** is the set of operations to be executed on the destination Channel, when the trigger is valid.

First three methods of this API aim to handle the management of Composition objects. A Composition object is composed by a trigger for the source Channel and an action for a destination Channel.

The Composition API provides the following set of methods (further detailed and updated in SocIoTal's WiKi [42]):

- **addComposition** [43]: to connect a source Channel that produces data to a target Channel able to receive and read the data. This data is the result of a triggered composition.
- **getCompositionslist** [44]: returns all the Compositions for a Channel
- **getComposition** [45]: returns a Composition representation given a Channel
- **updateComposition** [46]: updates a Composition for a Channel
- **deleteComposition** [47]: removes a Channel Composition from the User Environment.

9.3 Phone Management API

This API allows users to add a new virtual entity representing a smartphone and to send to it push notifications. Currently, only Android platform is supported.

The Phone Management API provides the following method (further detailed and updated in SocIoTal's WiKi [48]):

- **registerDevice** [49]: registers an Android smartphone device giving a GCM (Google Cloud Messaging) token. It is useful to send push notification from the User Environment.

The interaction between the Studio and the gateway relies on the HTTP protocol, using REST (REpresentational State Transfer) architectural style. The communication is always initiated by the studio, which can query the gateway to obtain the available devices, services, resources and associated values. The studio can also register a callback, to be notified by the gateway as soon as a resource value evolves. A complete documentation for the developer environment could be found online on the wiki [50].

Section 11 - **Conclusion**

This document presents the final specifications of the Application Programming Interface (API) that enables development of services on top of the SocIoTal platform. The goal is to bring services closer to citizens through suitable APIs consumable by application and service developers.

The API specification follows the best practices in documenting of such interfaces. A REST style API is approach used in the complete platform, making the utilisation unified and convenient for the end developers. This version of APIs is fully updated and maintained online [1] where the latest update, codes and libraries could be found.

For components only partial description is provided in this document followed with a link to the online resource where the complete specification is given to decrease verbosity of the document.

This environment have a goal to encourage and promote development of new services for the citizens and more wider public in general, giving them the means and the tools to compose partial building blocks or to fully employ potential of the platform.

It is expected that the SocIoTal platform lifecycle will reach beyond the project lifetime, thus it is suggested to follow the latest APIs version provided at the SocIoTal GitHub.

Section 12 - **References**

[1] SocIoTal GIT Hub Wiki, available online: https://github.com/sociotal/SOCIOTAL/wiki
[2] SocIoTal GIT hub Wiki, SocIoTal Context Manager Introdution, available at: https://github.com/sociotal/Context-Manager/blob/master/SocIoTal_CM_Intro.md
[3] SocIoTal GIT hub Wiki, SocIoTal Context Manager NGSI9, available at: https://github.com/sociotal/SOCIOTAL/wiki/SocIoTal-Context-Manager#ngsi9-api
[4] SocIoTal GIT hub Wiki, SocIoTal Context Manager NGSI10, available at: https://github.com/sociotal/SOCIOTAL/wiki/SocIoTal-Context-Manager#ngsi10-api
[5] SocIoTal GIT hub Wiki, SocIoTal Context Manager Extended Methods, available at: https://github.com/sociotal/SOCIOTAL/wiki/SocIoTal-Context-Manager#extended-methods
[6] SocIoTal GIT hub Wiki, SocIoTal Context Manager: Create Context, available at: https://github.com/sociotal/SOCIOTAL/wiki/SocIoTal-Context-Manager#createcontextentity
[7] SocIoTal GIT hub Wiki, SocIoTal Context Manager: Delete Context, available at: https://github.com/sociotal/SOCIOTAL/wiki/SocIoTal-Context-Manager#deletecontextentity
[8] SocIoTal GIT hub Wiki, SocIoTal Context Manager: Query Context, available at: https://github.com/sociotal/SOCIOTAL/wiki/SocIoTal-Context-Manager#querycontext-by-entityid
[9] SocIoTal GIT hub Wiki, SocIoTal Authorization manager, available at: https://github.com/sociotal/SOCIOTAL/wiki/SocIoTal-Authorization-Manager
[10] Constrained application protocol - CoAP, available at: http://coap.technology/impls.html
[11] SocIoTal GIT hub Wiki, SocIoTal Community Manager introduction and concept, available at: https://github.com/sociotal/CommunitiesManager/blob/master/SocIoTal_ComM_Intro.md#sociotal-communities-manager-introduction
[12] SocIoTal GIT hub Wiki, SocIoTal Community Manager: Register user, available at: https://github.com/sociotal/SOCIOTAL/wiki/SocIoTal-Communities-Manager#register-a-user
[13] SocIoTal GIT hub Wiki, SocIoTal Community Manager: Get user by id, available at: https://github.com/sociotal/SOCIOTAL/wiki/SocIoTal-Communities-Manager#get-user-info-by-id
[14] SocIoTal GIT hub Wiki, SocIoTal Community Manager: get user by name, available at: https://github.com/sociotal/SOCIOTAL/wiki/SocIoTal-Communities-Manager#get-user-info-by-name
[15] SocIoTal GIT hub Wiki, SocIoTal Community Manager: Delete user, available at: https://github.com/sociotal/SOCIOTAL/wiki/SocIoTal-Communities-Manager#delete-a-user
[16] SocIoTal GIT hub Wiki, SocIoTal Community Manager: Create a community, available at: https://github.com/sociotal/SOCIOTAL/wiki/SocIoTal-Communities-Manager#create-a-community
[17] SocIoTal GIT hub Wiki, SocIoTal Community Manager: List community, available at: https://github.com/sociotal/SOCIOTAL/wiki/SocIoTal-Communities-Manager#list-communities
[18] SocIoTal GIT hub Wiki, SocIoTal Community Manager: Get community info, available at: https://github.com/sociotal/SOCIOTAL/wiki/SocIoTal-Communities-Manager#get-community-info
[19] SocIoTal GIT hub Wiki, SocIoTal Community Manager: List available roles, available at: https://github.com/sociotal/SOCIOTAL/wiki/SocIoTal-Communities-Manager#list-available-roles

[20] SocIoTal GIT hub Wiki, SocIoTal Community Manager: Assign a role, available at: https://github.com/sociotal/SOCIOTAL/wiki/SocIoTal-Communities-Manager#assign-a-role
[21] SocIoTal GIT hub Wiki, SocIoTal Community Manager: List users, available at: https://github.com/sociotal/SOCIOTAL/wiki/SocIoTal-Communities-Manager#list-users
[22] SocIoTal GIT hub Wiki, SocIoTal Community Manager: Revoke a role, available at: https://github.com/sociotal/SOCIOTAL/wiki/SocIoTal-Communities-Manager#revoke-a-role
[23] SocIoTal GIT hub Wiki, SocIoTal Community Manager: Autoremove from a community, available at: https://github.com/sociotal/SOCIOTAL/wiki/SocIoTal-Communities-Manager#autoremove-from-a-community
[24] SocIoTal GIT hub Wiki, SocIoTal Community Manager: Delete a community, available at: https://github.com/sociotal/SOCIOTAL/wiki/SocIoTal-Communities-Manager#delete-a-community
[25] SocIoTal GIT hub Wiki, SocIoTal Community Manager: List my communities, available at: https://github.com/sociotal/SOCIOTAL/wiki/SocIoTal-Communities-Manager#list-my-communities
[26] SocIoTal GIT hub Wiki, SocIoTal Community Manager: Token request by username, available at: https://github.com/sociotal/SOCIOTAL/wiki/SocIoTal-Communities-Manager#token-request-by-username
[27] SocIoTal GIT hub Wiki, SocIoTal Community Manager: Simple token request by username, available at: https://github.com/sociotal/SOCIOTAL/wiki/SocIoTal-Communities-Manager#simple-token-request-by-username
[28] SocIoTal GIT hub Wiki, SocIoTal Community Manager: Validate a token, available at: https://github.com/sociotal/SOCIOTAL/wiki/SocIoTal-Communities-Manager#validate-a-token
[29] SocIoTal GIT hub Wiki, SocIoTal Identity manager, available at: https://github.com/sociotal/SOCIOTAL/wiki/SocIoTal-Identity-Manager
[30] "Idemix cryptographic library (Idemix), v2.3.0. IBM, available at http://www.zurich.ibm.com/idemix/downloads.html
[31] SocIoTal GIT hub Wiki, SocIoTal Group manager, available at: https://github.com/sociotal/SOCIOTAL/wiki/SocIoTal-Group-Manager
[32] SocIoTal GIT hub Wiki, SocIoTal Trust manager, available at: https://github.com/sociotal/SOCIOTAL/wiki/Trust-Manager-API
[33] SocIoTal GIT hub Wiki, SocIoTal F2F interaction API, available at: https://github.com/sociotal/SOCIOTAL/wiki/F2F-Interactions-API
[34] SocIoTal GIT hub Wiki, SocIoTal User Environment API, available at: https://github.com/sociotal/SOCIOTAL/wiki/User-Environment-API
[35] SocIoTal GIT hub Wiki, SocIoTal User Environment API Channel Management API, available at: https://github.com/sociotal/SOCIOTAL/wiki/User-Environment-API#channel-management-api
[36] SocIoTal GIT hub Wiki, SocIoTal User Environment API Channel Management API: Create a channel, available at: https://github.com/sociotal/SOCIOTAL/wiki/User-Environment-API#createchannel
[37] SocIoTal GIT hub Wiki, SocIoTal User Environment API Channel Management API: Update a channel, available at: https://github.com/sociotal/SOCIOTAL/wiki/User-Environment-API#updatechannel
[38] SocIoTal GIT hub Wiki, SocIoTal User Environment API Channel Management API: Delete a channel, available at: https://github.com/sociotal/SOCIOTAL/wiki/User-Environment-API#deletechannel
[39] SocIoTal GIT hub Wiki, SocIoTal User Environment API Channel Management API: Get the list of the channels, available at: https://github.com/sociotal/SOCIOTAL/wiki/User-Environment-API#getchannellist

[40] SocIoTal GIT hub Wiki, SocIoTal User Environment API Channel Management API: Get information about a channel , available at: https://github.com/sociotal/SOCIOTAL/wiki/User-Environment-API#getchannelinformation
[41] SocIoTal GIT hub Wiki, SocIoTal User Environment API Channel Management API: Get data stored in channels, available at: https://github.com/sociotal/SOCIOTAL/wiki/User-Environment-API#getchanneldata
[42] SocIoTal GIT hub Wiki, SocIoTal User Environment API Composition API, available at: https://github.com/sociotal/SOCIOTAL/wiki/User-Environment-API#composition-api
[43] SocIoTal GIT hub Wiki, SocIoTal User Environment API Composition API: Add composition, available at: https://github.com/sociotal/SOCIOTAL/wiki/User-Environment-API#addcomposition
[44] SocIoTal GIT hub Wiki, SocIoTal User Environment API Composition API: Get composition list, available at: https://github.com/sociotal/SOCIOTAL/wiki/User-Environment-API#getcompositionslist

[45] SocIoTal GIT hub Wiki, SocIoTal User Environment API Composition API: Get composition, available at: https://github.com/sociotal/SOCIOTAL/wiki/User-Environment-API#getcomposition
[46] SocIoTal GIT hub Wiki, SocIoTal User Environment API Composition API: Update a composition, available at: https://github.com/sociotal/SOCIOTAL/wiki/User-Environment-API#updatecomposition
[47] SocIoTal GIT hub Wiki, SocIoTal User Environment API Composition API: Delete a composition, available at: https://github.com/sociotal/SOCIOTAL/wiki/User-Environment-API#deletecomposition
[48] SocIoTal GIT hub Wiki, SocIoTal User Environment API Phone management API, available at: https://github.com/sociotal/SOCIOTAL/wiki/User-Environment-API#phone-management-api
[49] SocIoTal GIT hub Wiki, SocIoTal User Environment API Phone management API: Register a device, available at: https://github.com/sociotal/SOCIOTAL/wiki/User-Environment-API#registerdevice
[50] SocIoTal GIT hub Wiki, SocIoTal Developer Environment API, available at: https://github.com/sociotal/SOCIOTAL/wiki/Sociotal-Developer-Environment-API

Abbreviations and Acronyms

API: Application Programming Interface
CRUD: Set of operations for a resource, Create-Read-Update-Delete.
F2F: Face-to-face, referred to personal interactions that occur in proximity
HTTP: Hypertext Transfer Protocol
REST: Representational State Transfer
IoT: Internet of Things
IoT-A: an architectural reference model for IoT achieved by the IOT-A project
URL: Uniform Resource Locator
UserEnv: The SocIoTal End User Environment
VE: see Virtual Entity
Virtual Entity: Computational or data element representing a Physical Entity in IOT-a model
WS: Web Service
WP: Work Package
XACML: eXtensible Access Control Markup Language
XML: eXtensible Markup Language
JSON: JavaScript Object Notation