SEVENTH FRAMEWORK PROGRAMME

THEME ICT-2009.1.2

"Internet of Services, Software and Virtualization"

D2.3
SocIoS Object Model

Project acronym: SocIoS

Project full title: *Exploiting Social Networks for Building the Future Internet of Services*

Contract no.: 257774

Workpackage:	WP3	Toolsets	
Editor:	S. Porat		IBM
Author(s):	S. Porat		IBM
	N. Ioannou		Google
	K. Tserpes		NTUA
Authorized by	S. Porat		IBM
Doc Ref:	D2.3		
Reviewer	G. Papadakis		NTUA
Reviewer	N. Ioannou		Google
Dissemination Level	PU		

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2011] under grant agreement no. 257774

SOCIOS CONSORTIUM

Beneficiary Number	Beneficiary name	Beneficiary short name	Country	Date enter project	Date exit project
1(coordinator)	Institute of Communication and Computer Systems/National Technical University of Athens	ICCS/NTUA	Greece	Month 1	Month 30
2	IBM Haifa Research Lab	IBM	Istrael	Month 1	Month 30
3	Athens Technology Center	ATC	Greece	Month 1	Month 30
4	Google Ireland Limited	Google	Ireland	Month 1	Month 30
5	Cognium Systems	Cognium	France	Month 1	Month 30
6	Center for the Study of the Information Society, University of Haifa	HU	Israel	Month 1	Month 30
7	Deutsche Welle	DW	Germany	Month 1	Month 30
8	Stefi Productions S.A.	Stefi	Greece	Month 1	Month 30
9	Katholieke Universiteit Leuven (K.U.Leuven) – Interdisciplinary Centre for Law and ICT	KULeuven	Belgium	Month 1	Month 30

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2011] under grant agreement no. 257774

Executive Summary

This document describes the core concepts of SocIoS through an Object Model and the basic operations on these concepts as depicted via the set of Core Services. The Object Model and the Core Services constrain the datatypes and the services that SocIoS Auxiliary Services can use for their operation. SocIoS object model is a close variation of OpenSocial Data Specification, as developed by Google along with MySpace and a number of other social networks. The Object Model defines five core concepts that are directly mapped to entities that live in the underlying social networks. The core concepts are: Person, MediaItem, Activity, Message and Group and are used to represent SN users, SN content, SN activities, short messages posted to SN users and groups of users, respectively.

Section 1 and Section 2 provide overview information and describe the rationale behind the adoption of OpenSocial. Section 3 describes the core entities together with some utility types that are used to specify the core entities. Section 4 describes the Core Services that provide a unique point of interaction with SNS exposing operations that encapsulate the functionality of the underlying SN APIs. Finally, Section 5 shows a scenario of using SocIoS platform to compose a workflow using the Core Services and SocIoS Auxiliary Services.

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2011] under grant agreement no. 257774

1 Table of Contents

EXECUTIVE SUMMARY **3**

2 INTRODUCTION **1**

3 SOCIOS OBJECTS **3**

3.1 SocIoS Person object 4
3.1.1 Person Fields *4*
3.2 SocIoS Activity object 9
3.2.1 Activity Fields *9*
3.3 SocIoS MediaItem object 10
3.3.1 MediaItem Fields *11*
3.4 SocIoS Message object 12
3.4.1 Message Fields *12*
3.5 SocIoS Group object 13
3.5.1 Group Fields *13*
3.6 Other SocIoS objects 13

4 SOCIOS CORE SERVICES **15**

4.1 Auxiliary Types for Find Methods 16
4.2 SocIoS Core Services 17

5 SOCIOS SCENARIOS **18**

5.1 The Journalist Scenario 18
5.1.1 The Journalist Scenario Workflow *18*

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2011] under grant agreement no. 257774

2 Introduction

A fundamental layer is SocIoS architecture, as depicted in Figure 1, is the one that sits between the heterogeneous social networks (SNS) and SocIoS middleware. This layer, is composed by a set of Core Services (referred interchangeably as SocIoS API) acting as the single point of reference for the layers above, namely to SocIoS middleware and to SocIoS Auxiliary Services. On the other hand, this layer is providing a unified way to interact with the underlying SNS. As such, they define a single interface that masks various underlying SNS API methods. In terms of implementation, SocIoS Core Services are proxies to a set of adaptors, each interacts with a specific social network using the underlying API and wrapping the underlying SNS data types into SocIoS ones. The Core Services are used, for example, to extract information about entities that live in the underlying SNS, and to transform the result into entities that live in SocIoS.

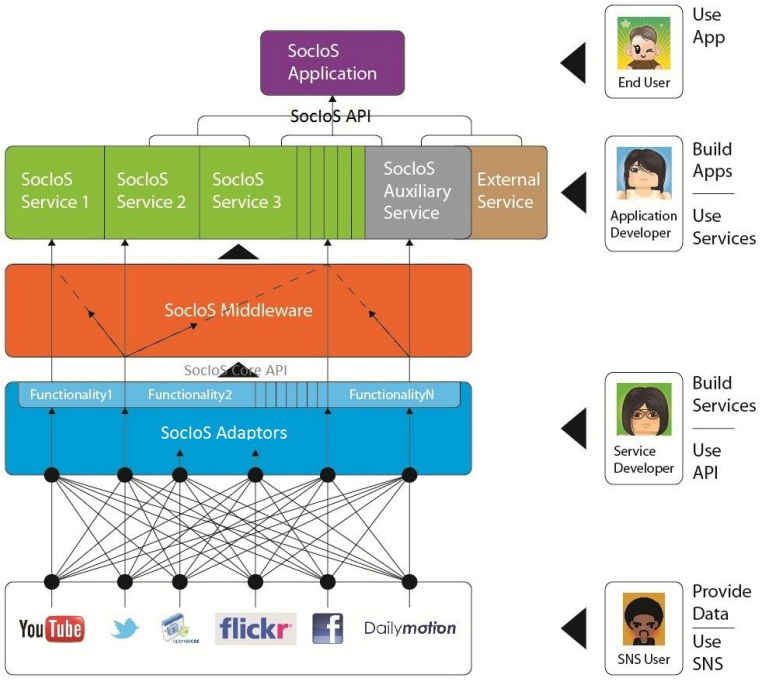


Figure 1: SocIoS Overview

SocIoS Object Model captures the set of datatypes (SocIoS objects) and the core APIs that are acting upon these datatypes (SocIoS Core Services). As explained above, this formal specification will allow the unification of the underlying object models of the SNSs. It must be able to capture the most dominant concepts and their characteristics, and provide mechanisms to manage SNS entities in a unified way.

Defining a clear Object Model has the potential to greatly assist the implementation of and integration effort needed for the layers above, since developers have a clear view of the objects that they need to manipulate as well as ways to manipulate them. Furthermore, the Object Model adds value to the exploitation of the project, as it becomes a point of

reference enabling the extension of its functionality and becoming a compatibility point to other solutions or even new SNS APIs.

In our effort to define SocIoS object models, both the datatypes and the Core Services, we studied the most prominent SNS APIs, according to criteria relevant to the SNS API such as popularity, maturity, openness, relevance and applicability to SocIoS goals and objectives. This analysis led us to a good understanding of what are the conceptually common objects that the SNS APIs are using. To test our conclusions, we implemented a showcase scenario in which the notion of MediaItem was tested under various SNS [1].

Even though the results from this showcase scenario were encouraging, the consortium had to deal with a major issue: the MediaItem object had to be defined from scratch while studying on one hand the SocIoS requirements and on the other hand a multitude of SNS APIs. This infused a rather large overhead (which should be repeated for all the other concepts), and many questions emerged as to which definition is the best. In order to resolve this issue, we turned to existing solutions that followed a similar rationale. Unfortunately, the most prominent specifications that are applied to projects such as GNIP® [2], hootsuite® [3] and Stroodle® [4] were not usable. The reasons are that: a) they are proprietary solutions and do not distribute their source code; b) they focus on a subset of SNS in comparison to what SocIoS wants to do (i.e. be SNS agnostic); c) they only implement a subset of the functionalities required by SocIoS and, as a result, it is practically impossible to implement the SocIoS scenarios using them; d) they are not extensible in terms of SNS they support and functionality; and, e) they act as passive APIs, without extending the API methods so that they can be then used by the developers.

The solution that seemed to be conceptually closer to SocIoS requirements and the best fit for our purpose turned to be OpenSocial. The rationale behind OpenSocial is to specify an API that will apply to a family of SNS, and allow a unified management of the content and functionality of this family. Even though the purpose of OpenSocial is different than that of SocIoS, it is close enough to serve as the base for SocIoS Object Model.

The next step was to select those OpenSocial objects that fit SocIoS domain concepts. We define five top-level, 1st class SocIoS entities, whose definition are based on the corresponding OpenSocial entities:

1. **Person**: Information on a person

[1] SocIoS initial showcase demo is available here: http://www.sociosproject.eu/Showcases/VeryInitialShowcase/tabid/382/language/en-US/Default.aspx

[2] Gnip, The Social Media API, available at: http://gnip.com/

[3] Hootsuit, Social Media Dashboard, available at: http://hootsuite.com/

[4] Stroodle, Your Social Pulse, available at: http://stroodle.it/

2. **Activity**: Anything a user does on a social network
3. **MediaItem**: Represents some content posted on a social network, like an image, movie, or audio
4. **Message**: Represents a short text message sent on a social network
5. **Group**: Used to identify a group of persons

It should be noted that SocIoS Object Model, as defined in this document, captures only those entities that represent corresponding entities in the underlying SNS. We plan to further extend this model to capture entities that are natural to the upper layers of SocIoS. More specifically, aggregated information, which is a result of processing SNS objects, is not captured in the core Object Model. For example, the notion of Event as an intense activity within the SNS-world is not included in SocIoS Object Model. The definition of an Event includes the processing and grouping of a multitude of relevant activities, which is a result of aggregating information about SNS objects. Such a task is left to SocIoS Auxiliary Services, and the associated complex entities are excluded from SocIoS core Object Model.

The next step after the definition of the five core SocIoS Objects was to define the methods that would allow the management of these objects and essentially provide access to content and functionality to underlying SNS APIs. For that purpose, SocIoS borrowed more datatypes from OpenSocial to describe useful components, such as dates, or compound types, such as lists. Finally, in order to enable multi-criteria queries for retrieving information about SNS entities, SocIoS adopts OpenSocial Filter datatypes.

We do allow some flexibility on the interpretation of the semantics of a SocIoS concept by the Auxiliary Services and potentially by the implementation of the adaptors. An example is the SocIoS Group entity which represents different grouping notions in the underlying social networks. Moreover, a SocIoS Group object may be created by an Auxiliary Service without having a mapped entity in some social network.

The next Section describes SocIoS objects. Section 4 then outlines the set of SocIoS Core Services, and finally we show in Section 5 how these objects and services are used, together with some Auxiliary Services to carry a scenario that will be demoed as part of the 1st pilot.

3 SocIoS Objects

This section describes the data types used by SocIoS Services. For clarity, we shall distinguish between the following levels of abstraction:

- **1st class entities**: capture the core concepts in SocIoS
- **2nd class entities**: auxiliary types that capture useful semantics applicable for each 1st class entity (like SocIoS object ID, or SocIoS Filter type that serve for retrieving information on the corresponding entity from the underlying SNs)

- **3rd class entities**: special types that are repeatedly used, mainly to capture Person identifiable information (like DateTime to tell the person's birthday or Address to represent his/her address)

We use common names for the primitive types that are used to define the elements of the above types, e.g. Int, String, Long, etc.

As pointed in Section 1, the Object Model consists of five top-level, 1st class entities, the definition of which is based on OpenSocial data specification. The following sub-sections, 3.1-3.5 describe in detail the five core entities: Person, Activity, MediaItem, Message and Group.

Each 1st class entity has a unique ID which identifies the object in SocIoS platform. Usually, there is a 1-1 mapping between a SocIoS 1st class object and an entity that lives in one of the underlying social networks. The SocIoS object ID is specifying the source social network, e.g., Facebook or Twitter. When passed as an argument to a Core Service, the ID field is used to select the adaptor that is responsible to retrieve the information about the associated SN object using the underlying SN APIs.

3.1 SocIoS Person object

The Person object encapsulates information on a particular SN user. The initial definition follows the OpenSocial Person object, and thus provides access to a user's information on some social network. As per the OpenSocial Specification, this information is stored in the user's profile and, depending on the site/social network, can include anything from "favorite TV shows" to "5 things you'll find in my bedroom."

An important field in a Person object is the unique identification of the person in SocIoS. This is depicted through an ObjectId object. It's worth noticing that in the first implementation of SocIoS infrastructure, each ObjectId represents the user identification in a particular social network. This means that John Miller that is registered in Facebook and Twitter will have two Person objects in SocIoS, one representing his contents in Facebook and the other in Twitter.

3.1.1 Person Fields

OpenSocial defines few dozens of attributes to represent a SN user, most of which stay as part of SocIoS Person object. Following is a partial list of Person fields (in alphabetical order), including those that are mandatory in the context of SocIoS' objectives. The WSDL for SocIoS Core Services provides the full list of Person fields.

Field Name	Type	Description
aboutMe	String	A general statement about the person
accounts	List of Account objects	Each element represents an online account held by this Person (Account is defined as a 3rd class entity in SocIoS)
activities	ActivityList	Person's favorite activities (ActivityList is defined as a 2nd class entity in SocIos; see below)
addresses	List of Address objects	Physical mailing addresses for this Person (Address is defined as a 3rd class entity in SocIoS)
age	Int	The age of this person
anniversary	DateTime	The wedding anniversary (DateTime is defined as a 3rd class entity in SocIoS)
appData	List of AppData objects	A collection of keys and values to represent the Person's social applications (AppData is defined as a 3rd class entity in SocIoS)
birthday	DateTime	The birthday of this Person
bodyType	String	Person's body characteristics
books	List of String values	Person's favorite books
cars	List of String values	Person's favorite cars
children	List of String values	Description of the Person's children
connected	Boolean	Boolean value indicating whether the user and this Person have established a bi-directionally asserted connection of some kind. The value is true if and only if there is at least one value for the relationship field, described below
currentLocation	String	Description of the person's current location
displayName	String	The name of the Person, suitable for display to end-users

Field Name	Type	Description
emails	List of String values	Each String value represents an email address of the Person
ethnicity	String	Person's ethnicity
fashion	String	Person's thoughts on fashion
food	List of String values	Person's favorite food
gender	String	The gender of the Person
happiestWhen	String	Describes when the person is happiest
hasApp	Boolean	Indicating whether the person has application installed
heroes	List of String values	Person's favorite heroes
humor	String	Person's thoughts on humor
id	ObjectId	Unique identifier for the person (ObjectId is defined as a 2nd class entity in SocIoS)
ims	List of String values	Each value in the list represents an instant messaging address for this person
interests	List of String values	Person's interests, hobbies or passions
jobInterests	List of String values	Person's favorite jobs, or job interests and skills
languagesSpoken	List of String values	List of the languages that the person speaks as ISO 639-1 codes
livingArrangement	String	Description of the person's living arrangement
lookingFor	List of String values	Person's statements about who or what they are looking for, or what they are interested in meeting people for
movies	List of String values	Person's favorite movies
music	List of String	Person's favorite music

	values	
Field Name	**Type**	**Description**
name	Name	The broken-out components and fully formatted version of the person's real name (Name is defined as a 3rd class entity in SocIoS)
networkPresence	NetworkPresence	Person's current network status. Specified as one of: AWAY, CHAT, DND, OFFLINE, ONLINE OR XA (NetworkPresence is defined as a 3rd class entity in SocIoS)
nickname	String	The casual way to address this Person 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
organizations	List of Organization objects	Each value describes a current or past organizational affiliation of this Person (Organization is defined as a 3rd class entity in SocIoS)
pets	List of String values	Description of the person's pets
phoneNumbers	List of String values	Each value represents a phone number for this Person
photos	List of String values	URL of a photo of this person. 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. 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.
politicalViews	List of String values	Person's political views
preferredUsername	String	Person's username (e.g. jsmarr or daveman692)
profileSong	String	URL of a person's profile song

Field Name	Type	Description
profileUrl	String	Person's profile URL, specified as a string. This URL must be fully qualified. Relative URLs will not work in gadgets
profileVideo	String	URL of a person's profile video
quotes	List of String values	Person's favorite quotes
relationshipStatus	String	Person's relationship status
religion	String	Person's religion or religious view
romance	String	Person's view about romance
scaredOf	String	What the person is scared of
sexualOrientation	String	Person's sexual orientation
smoker	String	Person's smoking status
sports	List of String values	Person's favorite sports
status	String	Person's status, headline or shoutout
tags	List of String values	A user-defined category label for this person, e.g. "favorite" or "web20". These values SHOULD be case-insensitive, and there SHOULD NOT be multiple tags provided for a given person that differ only in case.
thumbnailUrl	String	Person's photo thumbnail URL, specified as a string. This URL must be fully qualified. Relative URLs will not work in gadgets.
turnoffs	List of String values	Person's turn offs.
turnOns	List of String values	Person's turn ons
tvShows	List of String values	Person's favorite TV shows
updated	DateTime	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 DateTime. If this Person has never been modified since its initial creation, the value MUST be the same as the value of published. (DateTime is a 3rd class entity in SocIoS)
urls	List of String values	URL of a web page relating to this Person
utcOffset	Int	The offset from UTC of this Person's current time zone, as of the time this response was returned. The value MUST conform to the Date-UTC-Offset. 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

3.2 SocIoS Activity object

The Activity object captures anything a user does on a social network, e.g., tweet, update status, post (or comment on) a video, etc. In the future we shall extend this concept to capture additional SocIoS activities, like creating a new SocIoS Service.

3.2.1 Activity Fields

Following is a partial list of Activity fields (in alphabetical order), including those that are mandatory in the context of SocIoS' objectives. The WSDL for SocIoS Core Services provides the full list of Activity fields.

Field Name	Type	Description
appId	ObjectId	Specifying the application that this activity is associated with
body	String	Specifying an optional expanded version of an activity. Bodies may only have the following HTML tags: <b> <i>, <a>, . The container may ignore this formatting when rendering the activity
bodyId	ObjectId	Specifying the body message ID
externalId	ObjectId	An optional ID generated by the posting application
id	ObjectId	An ID that is permanently associated with this activity

Field Name	Type	Description
mediaItems	MediaItemList	Any photos, videos, or images that should be associated with the activity. Higher priority ones are higher in the list.
postedTime	Long	Specifying the time at which this activity took place in milliseconds since the epoch
priority	Double	A number between 0 and 1 representing the relative priority of this activity in relation to other activities from the same source
streamFaviconUrl	String	Specifying the URL for the stream's favicon
streamSourceUrl	String	Specifying the stream's source URL
streamTitle	String	Specifying the title of the stream
streamUrl	String	Specifying the stream's URL
templateParams	ActivityTemplateParams	A map of custom key/value pairs associated with this activity. These are used for evaluation in templates. The data has type ActivityTemplateParams which is a 3rd class entity in SocIoS (See the XSD for a full definition of this type)
title	String	Specifying the primary text of an activity. Titles may only have the following HTML tags: <b> <i>, <a>, . The container may ignore this formatting when rendering the activity
url	String	Specifying the URL that represents this activity
userId	ObjectId	ID of the user that performed this activity

3.3 SocIoS MediaItem object

The MediaItem object represents some content posted on a social network. In the 1st phase of the project, a MediaItem object represents an image, a movie, an audio file or a text update. In the future we may extend this type to represent other content elements, like Word documents or PowerPoint presentations.

3.3.1 MediaItem Fields

Following is a partial list of MediaItem fields (in alphabetical order), including those that are mandatory in the context of SocIoS' objectives. The WSDL for SocIoS Core Services provides the full list of MediaItem fields.

Field Name	Type	Description
albumId	ObjectId	Album to which the media item belongs
created	DateTime	Creation date time associated with the media item
description	String	Description of the media item
duration	Int	For audio/video clips - playtime length in seconds. set to -1/not defined if unknown
fileSize	Long	Number of bytes (set to -1/undefined if unknown)
id	ObjectId	ID associated with the media item
language	String	Language associated with the media item in ISO 639-3 format
location	Address	Location corresponding to the media item
mimeType	String	The MIME type of media, specified as a string
numComments	Int	Number of comments on the media item
numViews	Int	Number of views for the media item
numVotes	Int	Number of votes received for voting
rating	Int	Average rating of the media item on a scale of 0-10 (this value is retrieved from the underlying SN)
startTime	DateTime	For streaming/live content, date time when the content is available
taggedPeople	List of String values	List of string (IDs) of people tagged in the media item
tags	List of String values	Tags associated with this media item
thumbnailUrl	String	URL to a thumbnail image of the media item
title	String	Describing the media item
type	MediaItemType	The type of media, specified as a MediaItemType

		object
url	String	Specifying the URL where the media can be found

3.4 SocIoS Message object

The Message object is used to represent a short message posted on a social network, like a Facebook message or a Twitter tweet. The mechanism for sending a Message to a SN user is specified in the SendMessage Core service

3.4.1 Message Fields

Following is the list of fields (in an alphabetical order), as per OpenSocial data model

Field Name	Type	Description
appUrl	String	Identifies the application that generated this message
body	String	The main text of the message
bodyId	ObjectId	The main text of the message as a message template. Specifies the message ID to use in the gadget xml
collectionIds	ObjectIdList	Identifies the messages collection IDs this message is contained in
id	ObjectId	Unique ID for this message
inReplyTo	ObjectId	Message ID, use for threaded comments/messages. Reference the semantics of the Atom Threading model defined in rfc4685. URLs should be mapped to Atom <link rel="type" .../>
recipients	ObjectIdList	List of person IDs
replies	ObjectIdList	List of message ids. Reference the semantics of the Atom Threading model defined in rfc4685. URLs should be mapped to Atom <link rel="type" .../>
senderId	ObjectId	Id of person who sent the message
status	MessageStatus	Status of the message (NEW, READ, DELETED). The MessageStatus type is defined as a 3rd class entity in SocIoS
timeSent	DateTime	UTC time message was sent
title	String	The title of the message
titleId	ObjectId	The title of the message as a message template. Specifies

		the message ID to use in the gadget xml
updated	DateTime	Last update for this message
urls	List of String values	List of related URLs for this message. Supported URL types include 'alternate', alternate for this mailbox (text/html being the most common)

3.5 SocIoS Group object

Groups are meant to represent ensembles of persons. A SocIoS Group object may represent an ensemble defined in an underlying SN, like a List in Twitter that is used for categorizing users, or a Group in Facebook that is used to define an area where a set of people can communicate together. It's worth noticing that a SocIoS Group object is not necessarily mapped to an ensemble defined on a social network. A Group object may be created and processed by a SocIoS Auxiliary Service, in which case the object ID is granted by the service. An example of such Auxiliary Service is the Group Service that is used to manage groups of users.

3.5.1 Group Fields

Following is the list of fields (in an alphabetical order):

Field Name	Type	Description
description	String	Description of the group
id	ObjectId	Unique ID for this group
title	String	Title of group; used as the display name of the group

Notice that a Group object (though defined per OpenSocial data specification) does not represent an entity in the underlying social networks. A SocIoS group object is created and manipulated by the SocIoS Group Management Service. The operation getGroupMembers returns the list of Person objects associated with a given Group.

3.6 Other SocIoS objects

In addition to the five base entities described in the previous sub-section, SocIoS object model includes additional entities that capture identities of SocioS entities, lists of SocioS entities, filters that can be used to extract information from SN, and other basic datatypes.

We first define an abstraction to capture useful semantics that is applicable to each of the 1st class entities. We refer to these datatypes as SocIoS 2nd class entities:

- **Object Identification**

Each 1st class entity (Person, Activity, MediaItem, Message or Group object) has a unique identification represented through an **ObjectId** type. The ObjectId value is permanently held in the "id" field of the object. In case where the SocIoS object represents an entity that lives in an underlying SN, the ObjectId specifies the source SN, and is thus used by the Core Services to route the call to the proper adaptor to retrieve information about this entity from the proper SNS.

- **List of objects**

  A collection of 1st class objects or their IDs is heavily used by SocIos services. We use the convention "<XXX>List" to name the datatype that represents a collection of XXX objects. For example, a PersonList denotes a sequence of Person objects and a MediaItemList object represents a collection of MediaItem objects.

  Similarly, an ObjectIdList represents a sequence of ObjectId values. Since a collection of identifiers is always used to represent a collection of objects of the same type, we use multiple names "XXXIdList", all equivalent to ObjectIdList, to surface the types of the objects in the list. Thus, PersonIdList or MediaItemIdList are equivalent to ObjectIdList, and are used to specify lists of Person IDs or MediaItem IDs (respectively).

- **Filter objects**

  Filter types are used to represent multi-criteria queries for retrieving information about SNS entities. For each 1st class entity (Person, Activity, MediaItem, Group or Message) we define an associated Filter type (e.g., PersonFilter or MediaItemFilter) that determines possible criteria to search for SN entities (e.g., SN users or content elements, respectively).

  Each field in a 1st class entity determines some searching criteria. There is 1-1 association between fields of a 1st class datatype and fields in the corresponding Filter type. For example, a PersonFilter includes the fields 'age', 'displayName' and 'emails'.

  Filters are recursively defined to determine the queries used to search for SN entities. String values are used to apply string matching, while ranges are used to search for numeric values. For example, we can form a query to look for all SN users of age 30 to 40 (i.e., the 'age' field holds a number from 30 to 40), named "Miller" (i.e., the displayName field includes the String "Miller") and work in IBM (i.e., the emails field includes the String "ibm.com").

Finally, we use a 3rd level of abstraction to define datatypes that are used in 1st and 2nd class entities. We refer to this collection as 3rd class SocIoS entities. Following are few examples of such types:

- **Name**: represents a name of a Person
- **Account**: represents an account held by a Person

- **Address**: represents an address
- **DateTime**: represents a date
- Etc.

The socios-vocabulary.xsd provides a full list of these utility 3rd class SocIoS entities.

4 SocIoS Core Services

This section describes the web services that interact with the underlying SNs. They are used to extract information about SN entities and transform the result into the corresponding SocIoS entities. As such, they define a single interface that masks various underlying SNS API methods. In terms of implementation, the SocIoS Core Services are acting as the single point of reference for the layers above, namely the SocIoS middleware and the SocIoS Auxiliary Services. On the other hand, the Core Services are used as proxies to a set of adaptors, each is interacting with a specific social network and wraps the underlying SN data types into SocIoS ones.

In order for this mechanism to operate, there is the need for the adaptors to have access to SNS users' data. For that purpose, each adaptor executes the underlying SN APIs on behalf of accounts over that SN. The adaptors do things that a registered SN user could do through his account. The SN user has to authenticate the adaptor, which means that an access token is generated from the underlying SN platform. This access token is used as an argument in every call of the adaptor.

SocIoS AAA (Authorization, Authentication and Accounting) service is responsible for user's registration which populates user credentials, privileges allocation, user's validation and user's profile management. It provides keys that determine SocIoS end-users' permissions. An extended operation of the AAA component allows it to store the access tokens for every SN user that authenticates the adaptors, including of course the SocIoS end-users (since they too have SNS accounts). The Core Services use these stored tokens to perform the underlying SN APIs.

In summary, each of the Core Services accepts a parameter that represents an access token. For clarity, we shall omit this repeating parameter in the descriptions below.

We distinguish between four types of Core Services:

1. "Get" services that accept IDs of SN entities and return the associated SocIoS entities with detailed information as retrieved from the underlying SNs
2. "Find" services that accept a request for information on SN entities, apply sophisticated searches to discover all possible SN entities that match the request, and then return the associated SocIoS entities with detailed information as retrieved from the underlying SNs. The requests are defined using either filter objects or related entities.

3. "FindConnected" services that accept an ID of a SN entity and return the set of related SocIoS entities with detailed information as retrieved from the underlying SNs. It's worth noticing that in the 1st phase of the project we define one single interface to capture relationships between Person objects or between Activity objects, while each SN may have different interpretations for such relationships.

4. SendMessage that accepts a Person ID and sends a short message to the corresponding SN user

4.1 Auxiliary Types for Find Methods

We define the following request types to control information retrieval in Find methods:

1. **PersonFindRequest** that determines options to search for Person objects using one of the following arguments:

    - PersonFilter object that determines the criteria to search for Person information. Example filters are the age or the gender of the person, either as absolute values or in ranges.

    - MediaItemIdList object in which case the find method will look for all SN users that are related to the given MediaItem objects. For example, SN users that own the media items, commented on them or have re-shared them

2. **ActivityFindRequest** that determines options to search for Activity objects using one of the following arguments:

    - ActivityFilter object that determines the criteria to search for Activity objects. Example filters are the title of the activities or the date when they were published.

    - PersonFilter object in which case the find method will retrieve the activities of the persons matching the input criteria.

    - MediaItemIdList object in which case the find method will look for the activities associated with corresponding MediaItem objects, e.g. comments made on them.

    - PersonIdList object in which case the find method will retrieve the activities of the corresponding Person objects.

3. **MediaItemFindRequest** that determines options to search for MediaItem objects using one of the following arguments:

    - MediaItemFilter object that determines the criteria to search for MediaItem objects. Example filters are strings describing titles.

- PersonFilter object in which case the find method will retrieve the MediaItem objects that are related to those persons found using the input criteria.
- PersonIdList object in which case the find method will retrieve the media item objects owned by the corresponding Person objects.

4.2 SocIoS Core Services

We define the following Core Services:

1. **FindPersons**: can be used to search and retrieve information about SN users
   - accepts a PersonFindRequest object
   - returns a PersonList object containing Person objects that are found based on the criteria defined via the given PersonFindRequest object
2. **GetPersons**: can be used to retrieve information on SN users, given a list of IDs
   - accepts a PersonIdList object
   - returns a PersonList object with information on all Person objects whose IDs is given as input
3. **FindConnectedPersons**: can be used to get a list of person IDs that are connected/related to a given person
   - accepts an ObjectId that identifies a SN user
   - returns a PersonList object with information on all Person objects that relate to the given Person
4. **FindMediaItems**: can be used to search and retrieve pictures, audio or video from the underlying SNs
   - accepts a MediaItemFindRequest object
   - returns a MediaItemList object containing MediaItem objects that are found based on the criteria defined via the given MediaItemFindRequest object
5. **GetMediaItems**: can be used to retrieve information on media items that live in the underlying SNs, given a list of IDs
   - accepts a MediaItemIdList object
   - returns a MediaItemList object with information on the MediaItem objects whose IDs is given as input

6. **FindActivities**: can be used to search and retrieve information on activities that happened on the underlying SNs
   - accepts a ActivityFindRequest object
   - returns a ActivityList object containing Activity objects that are found based on the criteria defined via the given ActivityFindRequest object
7. **GetActivities**: can be used to retrieve information on activities given a list of IDs
   - accepts a ActivityIdList object
   - returns a ActivityList object with information on all Activity objects whose IDs is given as input
8. **FindConnectedActivities**: can be used to get a list of IDs that identify activities that are connected/related to a given activity (e.g. re-tweets)
   - accepts a ActivityId object
   - returns a ActivityList object that represents the Activity objects that relate to the given Activity
9. **SendMessage**: can be used to post a message to some SN user
   - accepts a PersonId object and a Message object
   - Sends a message to the corresponding SNS user

5 SocIoS Scenarios

This section describes the scenario that will be implemented as part of the project 1st pilot. In context of this document, we show how this scenario is going to be implemented by composing some workflows using the Core Services and the Auxiliary Services.

5.1 The Journalist Scenario

Birgit, a journalist from DW, wants to monitor the activity of a group of SNS users who are related to a certain topic or a location. In detail she wants to get all the information possible regarding the clashes in Syria, taking advantage of the sharing of content from a vast amount of information sources (possibly original user created content) and the viral dissemination effect of the SNS (get access to news in a timely fashion). She is interested into monitoring the content-related activities of a group of people and use the system to effectively sort these activities based on whether they are related to a real-life important event or not.

5.1.1 The Journalist Scenario Workflow

We describe the first three sub-workflows that will implement the scenario.

Sub-Workflow 1: Journalist gets her SNS friends and then searches in their content to find something related to “Syria clashes”. Then she gets the filtered list of SNS users and tags them with a Group name

- GetPersons Core Service is used to create a SocIoS Person object for the journalist
- FindConnectedPersons Core Service is used to get a list of Person objects, each representing a SNS friend of the journalist
- FindActivities Core Service is used to search the contents of these friends to find activities related to “Syria clashes"
- GetPersons Core Service is used to retrieve all information on the selected list of Person objects
- The Auxiliary Group Service is then used to group these Person objects together and tag them with a Group name

Sub-Workflow 2: The journalist searched in the whole SNS user base that SocIoS has access, for users with content that relates to “Syria clashes” and when she finds them she extends the previously created user Group.

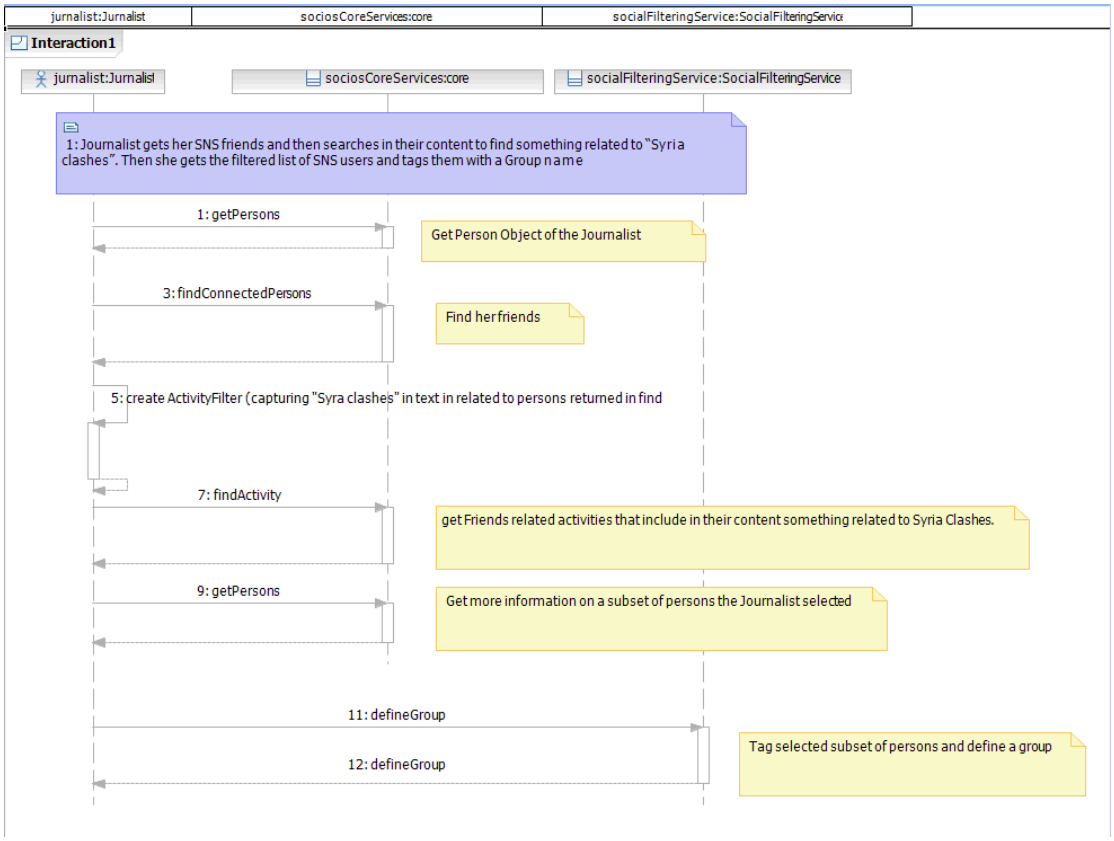
- FindActivities Core Service is used to search the whole SNS user base that SocIoS has access to find activities related to “Syria clashes"
- GetPersons Core Service is used to retrieve all information on the selected list of related Person objects
- The Auxiliary Group Service is used to extend the previously created Group
- FindMediaItems Core Service is used to search the whole SNS user base that SocIoS has access to find media items related to “Syria clashes"
- GetPersons Core Service is used to retrieve all information on the selected list of related Person objects
- The Auxiliary Group Service is used to extend the previously created Group

Sub-Workflow 3: The journalist initiates a search for SNS users that are located in "Syria" or are related in any other way with this country. She gets the returned users and extends the user Group by adding them too.

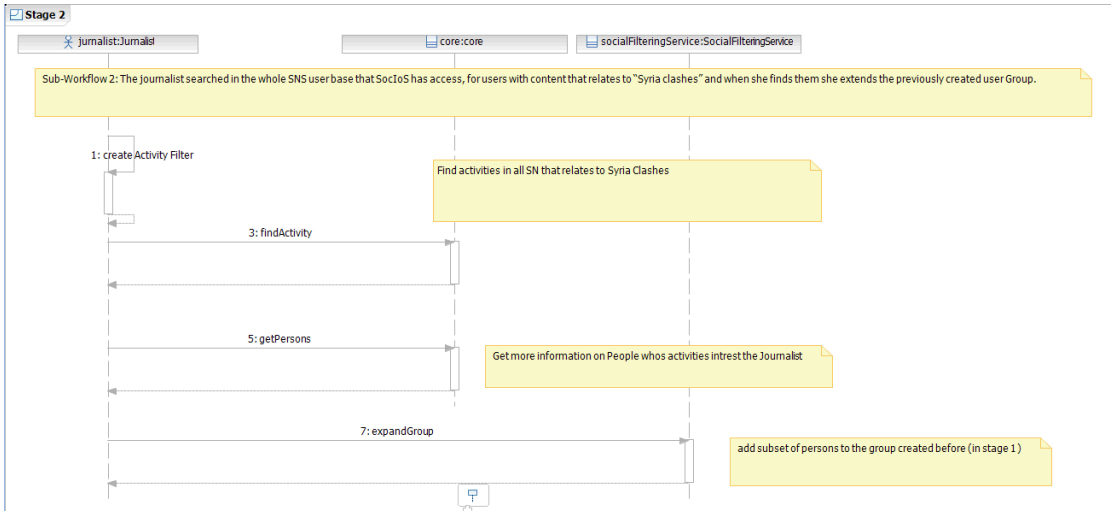
- A PersonFindRequest object is generated to search for Person objects according to their location field or any other field that may include the String "Syria"
- FindPersons Core Service uses the PersonFindRequest object to retrieve information on all SN users that are located or related in any way with “Syria"
- The Auxiliary Group Service is used to extend the previously created Group

Attached are the sequence diagrams for Sub-Workflows 1-3

Sequence diagram for Sub-Workflow #1



Sequence diagram for Sub-Workflow #2



Sequence diagram for Sub-Workflow #3