D33.6

3.6 MOBiNET Identity Principles

Work package: **3.3**
Version number: **1.1**
Dissemination level: **PU**
Date: **26/11/2013**

7th RTD Framework Programme
Directorate General for Communications Networks, Content & Technology
Cooperative Systems for energy efficient and sustainable mobility (FP7-ICT-2011-6.7)
Contract Type: Collaborative project
Grant agreement no.: 318485

Version Control

Version history			
Version	**Date**	**Main author**	**Summary of changes**
0.1	30/09/13	Rieks Joosten	Initial version
0.2	08/10/13	Rieks Joosten	Filled in missing gaps, renamed '*domain*' into '*i-Scope*', published on EmDesk
0.3	10/10/13	Rieks Joosten	Filled in missing gaps, published on EmDesk
0.4	14/10/13	Thierry Jacquin	Merge with Privacy design
0.9	15/10/13	Rieks Joosten	Redaction changes; review ready
1.0	31/10/13	Rieks Joosten	Finalized
1.1	26/11/13	Marco Annoni, Enzo Contini	Review of version 1.0; Added sub-clause on Authorization; additional explanations in section 5
	Name		**Date**
Prepared			
Reviewed	Marco Annoni, Enzo Contini		26/11/2013
Authorised	Dorota Boruc		27/11/2013
Circulation			
Recipient		**Date of submission**	
European Commission			
Project partners			

Authors

Rieks Joosten, Thierry Jacquin, Marco Annoni, Enzo Contini

Table of contents

Abbreviations and definitions

Abbreviation	Description
ADAC	Allgemeiner Deutscher Automobil-Club
BSN	Burger Service Number (Dutch equivalent of social security number)
CO	Car Owner
GBA	Gemeentelijke Basis Administratie (administration of Dutch citizens)
GPRS	General Packet Radio Service
GUI	Graphical User Interface
HW	Hardware
IC	Insurance Company
IMEI	International Mobile Equipment Identity
IP	Internet Protocol
IT	Information Technology
MGA	MOBiNET Governing Authority
MIM	MOBiNET Identity Module, or MOBiNET Identity Management
PAYD	Pay As You Drive
PI	Personal Information
PW	Password
SMS	Short Message System
SP	Service Provider
SSO	Single Sign On
TDSP	Telematics Data Service Provider
U/PW	Username/Password
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator

Executive Summary

This document specifies an ontology, i.e. a set of terms and constraints that (the use of) such terms must satisfy, allowing the precise expression of any concern related to ‘identity’ and ‘identity management’ within the scope of MOBiNET, and particularly the MOBiNET Identity Module (MIM). Also, it gives examples of how the MIM capabilities can be used for interacting with appropriate endpoints at the MOBiNET service provider premises, for the purpose of user authentication and, taking into account the Laws of Identity and the principles of "Privacy by Design".

The focus of this document is on the – often neglected – information level. To this end, both terms that are commonly used (such as *identity*, *user*, *session*) and new terms (e.g. *information scope* or *i-Scope*) are precisely specified, as well as the relation these terms have to one another, and constraints that have to be satisfied. These terms, relations and constraints may be referred to as an ontology, whose purpose (and contribution) is to precisely express not only concerns, but also solutions, in a sense that can be grasped by anyone committed to this ontology. This document does not specify how information is to be represented or processed, what registries should look like, which attributes users should have, nor how or where the information is to be stored.

A main contribution of this document is that it explicitly and consequently addresses the accountability for the information, e.g. with respect to its meaning (semantics), its relation to other information-pieces, information-constraints that must be satisfied, the truthfulness of (the data representing) the information, etc., which we together refer to as ‘information management’. Since also the MOBiNET consortium, or ultimately the commercial party that operates the MOBiNET platform and ecosystem, will be required to bear such accountability, we need a term to refer to such an accountable entity; we called it the MOBiNET Governing Authority or MGA. Like any other business party that has its own information, it will need to not only concern itself with the information management, but also with specifying and operationalizing the associated business processes. This, however, is outside the scope of this document.

This document starts with an introduction explaining the need for its contents. Then it continues with defining identities and attributes, as these are the key components of any information system. In particular, care is taken to associate such definitions with (business) responsibility (accountability). Next, we introduce terminology that allows us to discuss issues that occur in the context of information processing, such as ‘sessions’, ‘login’, etc. A separate chapter specifies information registrations and how they are used. That sets the stage for a specification of the MIM features and capabilities. We conclude by giving various examples of how this can be used in practice. The definitions are summed up in the glossary.

1. Introduction

Organizations have more and more turned to service-oriented architectures (SOA's), making their services available to other organizations that in turn combine them in unforeseen ways, causing the emergence of so-called "service ecosystems"[1]. Projects such as The Service Ecosystem[2] and SEE[3] attempt to contribute to the architecture and engineering of such ecosystems. MOBiNET[4] is a project that aims to create a large-scale ecosystem for mobility services with parties in many countries.

In order for a service ecosystem to be usable, it must allow accounts that users already have with one of the ecosystem partners, to be used for obtaining services of other partners in the ecosystem. For example, if a motor club and a parking lot operator both were members of the MOBiNET ecosystem, a motor club member should be able to park in a parking lot of the parking lot operator, and pay for the use. The electronic trans*actions* involved not only concern services of the motor club and the parking lot operator, but also of financial institutions and - depending on the actual service that is obtained, of other members of the ecosystem. The services provided by such parties must not only interoperate at the technical level, but at the information level as well: it must be ensured that the parking lot operator receives proper payment and that this amount is deducted from the proper bank account, which may be the account of the driver, the account of his employer, or someone else, depending on the arrangements made.

Interoperability at the information level is quite different from interoperability at the technical level, because at the technical level (particularly at the lowest levels), there are already standards that can be agreed upon. A simple example is that networking is done based on IPv4 (and/or any other standardized protocols).

At the information level, things are quite different. While we generally have agreement about the terminology (i.e. the words, the 'raw data'), there is ((much,)much) less agreement on what a word means. For example, attributing the characteristic 'trustworthy' to a person or organization has a different meaning for different individuals. The same is true for many, if not most, other words. The consequence of this is that if different parties in one ecosystem use the same words, that does not imply that they assign it the same information, or that they agree. Any agreement with respect to the meaning of words in an ecosystem has to be 'standardized' in a similar way as e.g. network protocols are standardized.

For us, the fact that different parties/individuals all assign their own meaning to words, is an axiom. It is the basic fundament on top of which we build our ontology for identity, identifiers, and other terms.

Information coherence, e.g. the property that the parking lot operator receives proper payment and that this amount is deducted from the proper bank account, requires that every ecosystem party is capable of

[1] A. P. Barros and M. Dumas. The Rise of Web Service Ecosystems. IT Profesional, Vol. 8, Issue 5, pp 31-37, September/October 2006

[2] http://agentgroup.unimo.it/wiki/index.php/The_Service_Ecosystem

[3] http://www.cs.helsinki.fi/u/thruokol/see/

[4] http://www.mobinet.eu/

- identifying entities (e.g. the driver, his car, his employer) (at a confidence level that is sufficient for the purposes of that part);
- registering entities, i.e. storing information that the party attributes to such entities (e.g. account numbers, license plates);
- communicate about entities, i.e. sending and receiving messages containing data pertaining to entities, in such a way that the meaning that the sending party attributes to this data is consistent and coherent with the meaning that the receiving party attributes to such[5].

Traditionally, parties (e.g. service providers) have been working 'stand-alone' for a large part, which means that as long as entity identification and registration works for them, that's ok. Cooperating with other parties requires that they are able to communicate about entities. As we stated, it is not self-evident that attributes assigned by one party have the same, or even a similar meaning, for all other parties. Transferring information about entities has proven to be difficult. A major cause for these difficulties is that organizations tend to focus on technologies (protocols, structuring the data and such) and have too little awareness that this data means different things for different parties (at the information level). This attitude may have serious consequences, as the Scientific Council for Government Policy showed for the Dutch government[6].

Understanding identities, attributes and their transferral, to the extent that IT-requirements can be drafted and implemented, requires a formal framework that allows concerns of the various stakeholders to be expressed and addressed. For service ecosystems such as MOBiNET, such a framework needs to specify the Identity Management service that is envisaged as part of the ecosystem infrastructure. We have developed the MOBiNET Identity Framework for this.

The **MOBiNET Identity Framework** (which is described in this document) is a conceptual framework at the information level (consider it an ontology). Thus, it defines the meaning of terms related to 'identity' that are relevant for (various) MOBiNET stakeholders. Whatever implementation(s) will be used for the MOBiNET Identity Manager (MIM), it MUST process identities, attributes and the like in accordance with the meaning specified in this framework.

Words that are part of the framework's ontology are *typeset in italic*; when words are defined, they are ***typeset in bold italic***.

[5] In order to be able to address such semantic interoperability issues, it is important to clearly distinguish between 'data' and 'information' (or 'meaning').

[6] Scientific Council for Government Policy, iGovernment, Amsterdam University Press, 2011

2. Identity Basics

Names, serial numbers, social security numbers, student numbers, patient numbers, usernames, etcetera are commonly called identities. Although their purpose is to identify individuals, and therefore are also referred to as 'identifiers', we will use the word *'identity'* because in our experience this gives the least amount of terminological debate.

2.1. Identities and Identifiers

An *identity* as we use it, has the property that it identifies a person (or another *entity*[7]) within a specific *i-Scope*[8] and as such uniquely represents that entity. For example, '6123.45.671' is an *identity* within the *i-Scope* of the Dutch register of citizens (the 'GBA'), within which it identifies and represents (a record for) a person. In another context, e.g. a bank, it might also be an *identity*, identifying and representing a bank account. As another example, 'mama' identifies a single person for many people (= many different *i-Scope*s). Thus, apart from being a sequence of bits or (alphanumeric) characters, every *identity* has the property of identifying a single *entity* within a specific *i-Scope*. Character strings that either do not identify an *entity* or are not associated with an *i-Scope*, are not considered *identities*.

Identities may be typed, meaning that every *identity* of a specific type identifies an entity of that type. For example, *identities* in the GBA-*i-Scope*, i.e. the character sequences used to identify (registered) Dutch citizens, are all of the same kind, and are called 'Burgerservicenummer's or BSN's. Typing allows *identities* to identify multiple *entities* in a single *i-Scope*. For example, within the *i-Scope* of an airport, 'A3' may not only identify a gate (it is then a 'gate number') but also a parking lot (in which case it is a 'parking lot number').

Typed *identities* are subject to syntactical constraints that distinguish valid representations from (syntactically) invalid ones. For example, credit card numbers, IMEI numbers and the like must satisfy a specific checksum formula in order to be valid[9]. IPv4-addresses must consist of four decimal numbers in the range [0..255] separated by dots. Syntactical constraints are imposed for a variety of reasons that are beyond the scope of this article.

The above properties, that are readily observed in practice, lead to the following specification of *identity*:

with the following definitions:

[7] An entity is basically something that exists. Merriam-Webster has a more precise definition.

[8] Words that may have the same meaning include: scope, information scope, context, organization, person.

[9] See e.g. http://en.wikipedia.org/wiki/Luhn_algorithm.

Term	Meaning
Identity	A symbol (bitstring, alphanumeric characterstring, picture, set of attributes, etc.) that (1) identifies and uniquely represents *Entity* within its *i-Scope* and (2) satisfies the syntactic criterion as specified in *Type*.
i-Scope[10]	A scope, within which an information model of (usually a small part of) the real world exists, expressed in terms of *symbols*, relations (between them) and constraints (expressed in terms of these *symbols* and relations), and for the coherence, consistency, correctness, currentness etc., a single *party* is accountable.
party	An organization or (a group of) *person(s)* that is capable of being held accountable.
Type	A composite of 1) a criterion that distinguishes between *entities* that are of a specific class and *entities* that are not and 2) a criterion that distinguishes between valid and invalid *identity* representations (of this *Type*), where both criteria are valid within the *identity's i-Scope*, and are governed by the *party* that is accountable for the *i-Scope*.
Entity	Something that exists - see also the definition of Merriam-Webster).

NOTES

1. An *i-Scope* (or: *information scope*) may be seen as the expression in terms of data elements of how (part of) the 'real world' is perceived by the *party* that is accountable for the *i-Scope*. Any claim, any truth, any assertion that is done by, or under accountability of, this *party*, originates from one of its *i-Scopes*.

2. When we say that something is, or falls under, the responsibility of an *i-Scope*', this implies that the accountability for this is born by the *party* that is accountable for the *i-Scope*. Note that since every *i-Scope* only has one such *party*, this accountability is established unambiguously.

3. It is the responsibility of the *i-Scope* to guarantee that each *identity* is unique, i.e. that every *identity* of a certain *type* identifies (represents) at most one *entity*, AND that every *entity* that must be identifiable within the *i-Scope* has at least one *identity* (representation).

4. The lifetime of an *identity* within an *i-Scope* needs not be the same as that of the *entity* it identifies. We use the term ***persistent identity*** for any *identity* for which the *i-Scope* guarantees that it will live at least as long as the *entity* it identifies within that *i-Scope*, AND it will also live at least as long as this *entity* needs to be referable within that *i-Scope*[11]. The BSN is an example of a *persistent identity*. Usernames are generally not *persistent identities* for people (but may be *persistent identities* for *user accounts*).

5. An *identity* that is defined in an *i-Scope* can be used within other *i-Scope*s as well. For example, the BSN, the *i-Scope* of which is limited to the Dutch government, is also used in healthcare, and may well be used by banks in the near future.

[10] An *i-Scope* may also be referred to as '*information scope*' or, for our purposes, as '*identity scope*'.

[11] A person that has deceased must still be referable, e.g. for handling its will.

6. The practical use of our definition of *identity* has been demonstrated in various discussions. One example is the discussion of whether a BSN can be used as (the sole) 'global *identity*' for patients in the healthcare *i-Scope*. If it could, then all healthcare organizations could refer to their patients using this single *identity*, relieving them of the task of translating patient numbers in the health records that they receive from other organizations to patient numbers that are known in their own patient registrations. Using our model, it becomes quickly apparent that the set of people having a BSN is neither the same nor a superset of the set of people that apply for healthcare: people that are illegally in the country do not have a BSN but would need an *identity* for receiving healthcare. Therefore, the *Type* of BSNs differs from what is required in healthcare implying that BSNs cannot be used as the sole 'global *identity*' in healthcare.

Within an *i-Scope*, entities can be attributed properties. For example, a person can be attributed with an age or a birthdate. The set of attributed properties (which we call ***attribute***s) is also called a ***record*** or 'profile'. Every *attribute* in a *record* is a property[12] of one and the same *entity*. It is important that records can be found within an *i-Scope*, because they contain information that is needed to fulfill trans*action*s in which both the *i-Scope* and the *entity* participate.

Finding a *record* is easy if an *identity* for the *record* is provided as search key, and the *identity* is stored in the *record* itself as this would be a simple lookup. For example, if you phone your insurance company and you specify the policy number, they can quickly find the information related to the insurance policy.

However, if you do not know an *identity*, you can also specify multiple *attribute*s of that policy, such as the name and address of its holder, and its birthdate (in case a married couple lives at that address). Any *attribute* may contribute to the finding of the policy *record*, but it will only be found if the set of *attribute*s are not shared with any other policy *record*. We say that any set of *attribute*s that identifies a *record* is a ***complex identifier***.

Note that an *identity* that identifies a *record* that contains *attributes* of a (real-world) *entity* identifies this *entity* (indirectly and implicitly). The converse, however, is not necessarily true.

2.2. Identity (and Information) Management

The previous section nicely illustrate our axiom that every *i-Scope* (person or organization) assigns its own meaning to words (data, symbols, *identities*). In order to end up with a consistent and coherent 'language' (which here includes the meaning or semantics), every *i-Scope* must manage this language, as it may change to accommodate for new situations. There are different kinds of changes:

- *identities* (names) of *entities* may change, e.g. a computer called 'Asterix' may be renamed into 'C-3PO'. Names of *Types* may change, e.g. 'Bills' may be renamed to 'Invoices'. Note that the name of a *Type*, is also an *identity*, but it is the *identity* of an *entity* at a higher abstr*action* level.

- *entities* whose existence becomes known, e.g. a new customer, must be assigned a name (*identity*) within the *i-Scope*; the same thing holds for new *Types* of entities.

Identity Management (and for that matter: Information Management) is an activity that is performed within a single *i-Scope*, and includes the issuing and modification of *identities* to *entities*, at any

[12] not: the property of the *entity*. The (responsibility for the) decision to attribute something to an *entity* is made within the *domain*.

abstraction level (thus including the issuing and modification of *Types*). Also, Identity Management must ensure that the relations between *entities* that are known in the *i-Scope* (i.e. they have an *identity* assigned), which are used for reasoning within that *i-Scope*, are properly defined (both in terms of syntax and semantics – the latter is necessary for proper reasoning).

Thus, Identity Management is a task for every *i-Scope*, and limits itself to the *entities*, *Types*, and relations between them, insofar as they are known within that *i-Scope*.

2.3. I-Scopes, Assertions, Claims, Credentials and Authentication

An ***i-Scope*** is a scope, within which an information model of (usually a small part of) the real world exists, expressed in terms of *symbols*, relations (between them) and constraints (expressed in terms of these *symbols* and relations), and for the coherence, consistency, correctness, currentness etc., a single *party* is accountable. In a sense, an *i-Scope* contains an 'image of the real-world', as perceived by a specific *party*. Thus, within an *i-Scope*, it is possible to reason with data (that represents specific information within the *i-Scope*) and produce more data/information when doing so. Consequently, the truths within a given *i-Scope* may be quite different from those in another *i-Scope*. Therefore, it is meaningful to know from which *i-Scope* data originates (and hence where to go to find out what it means), in which *i-Scope* additional data is created (and finding out what that means), etc.

While acquiring, storing, reasoning with data within a single *i-Scope* is rather straightforward, it becomes more difficult when data is transferred between *i-Scopes*. For example, in one *i-Scope* a 'customer' might someone that pays for a service, whereas in another *i-Scope* a 'customer' may be the one actually using a service. Thus, if the first *i-Scope* reasons using its own meaning of 'customer' using 'customers' from the other *i-Scope*, it may cause problems because those customers may not expect to be paying for the service.

In order to facilitate the exchange of information, we introduce the following terms:

Term	Meaning
Assertion	(A sequence of *symbols* that represent) a statement uttered by (or on behalf of) a *party*.
Claim	An *assertion* by some *party*, that this *party* assures is true.
Credentials	A set of *claims* of a single *party*, the information in which a) may be used by another *party* for some business purpose and b) can be ascertained as being authentic by this other party.
Authentication (of credentials)	The assessment (by some actor) of the authenticity (truthfulness according to the *issuer*) of (the *claims* in) *credentials* that the *actor* has received within the context of a *session*.

NOTES

1. While *assertions* are uttered by a *party*, we do not make assumptions with respect to the truth thereof. After all, liars are known to exist…

2. *Claims* are basically *assertions* that are committed to by the uttering *party*. One way that this commitment can be shown is by a (digital) signature, so that a *claim* is a (digitally) signed *assertion*.

3. Examples of *credentials* could be a username-password pair, a (digitally signed) certificate that identifies a user or a service, and/or contains *attributes*, etc.

4. There is also 'message authentication', which is an assessment that a received message actually was sent by the alleged sender.

5. Note that it is up to the 'business' to decide what criteria the *actor* must apply to decide that the *credentials* are authentic. Theoretically, this business decision should be underpinned by a risk analysis that takes into account the kinds of *actions* that the server may be executing as a consequence of this decision.

2.4. Identities and the MOBiNET Governing Authority and Eco-system

According to this framework, whenever we talk about a 'MOBiNET identity', we must assume the existence of an associated *i-Scope* (including its accountable *party*) and *Type*. To describe this more specifically, we introduce the following terms:

Term	Meaning
MGA	See: *MOBiNET Governing Authority*
MOBiNET Governing Authority	A *party*[13] that, amongst other things, governs syntax and semantics of types (including entity-types) that are to be used throughout the MOBiNET ecosystem.
MOBiNET Ecosystem	The scope within which *partners* cooperate.
partner	A *party* that takes part in the *MOBiNET ecosystem*, and as such abides by the ecosystem laws/rules/policies[13] as set forth by the *MOBiNET Governing Authority*.

Using these terms, we stipulate that:

1. the *i-Scope* of a '*MOBiNET identity*' is the *MOBiNET ecosystem*, which means that all MOBiNET identities are meaningful within this ecosystem.

2. the *party* that is accountable for the *i-Scope* of a '*MOBiNET identity*' is *MOBiNET Governing Authority* (or ***MGA***), which is a *party* that, amongst other things, governs syntax and semantics of *Types* that are to be used throughout the MOBiNET ecosystem.

Identity Types that are (already) defined by some *party* may be used within the *MOBiNET ecosystem* provided their *Type* criteria are properly disclosed by this *party*. This allows for (semantic) standards to be used. Note that this is the case regardless of whether the *MOBiNET ecosystem* exists or not.

[13] This must be realized by SP6.

Restricting the use of *Identity Types* that are (already) defined by a *partner* to other *partners* within the *MOBiNET ecosystem*, can be achieved by disclosing the *Type* criteria through a registration service within the *MOBiNET ecosystem*.

Identity Types may also be defined for use throughout, and perhaps also outside, the *MOBiNET ecosystem*. The specifications of such *Types* is then decided on / governed by the *MGA*[13].

2.5. Personal Information and Privacy

Because personal information (PI) and privacy are closely related to *identities* and the management thereof, we introduce these terms here lightly. There are many flavors of personal information. One might distinguish the following types of *personal information*:

- information about a person (e.g. the color of its hair);
- information about how a person wants an application (or more generally: a web-service) to preferably behave (personal preferences);
- information about the context of a person, which could be seen as any information related to a person that is not about the person itself or a preference.

For quite some information systems, their specifications state what information is to be stored. Often, such specifications are not related to actual business requirements, but stem from a 'hallucination' of what businesses might want/need. This may result in the capability of storing a large number of possible data types (attributes). For example, Active Directory allows you to store over 1000 attributes for users. Consequently, nobody really knows which attribute is to be used for what purpose.

Within MOBiNET, we do this in a different manner. If there is a need for a 'global' definition of personal information, this is to be decided by the *MGA*, thus guaranteeing commitment of all MOBiNET *partners*. All other (and this will be most) information will be related to specific applications or services and therefore must be specified, accumulated, stored and disseminated under the accountability of the *parties* that own/govern such applications or services.

MOBiNET allows you to search for information that is available from the various *partners* in a similar way as that it allows you to search for services that are available from the various *partners*. This allows for optimal flexibility while limiting the burden put on the MOBiNET community and platform.

Privacy is a term with many definitions and surrounded with many opinions. Within MOBiNET, we will use the relative simple paradigm that any (personal) information that is processed (used, transferred, …) at run-time by a web-service must be necessary for the function and relevant for the purpose of that web-service (purpose binding), or for a web-service that is actually called (run-time).

We introduce the following terms for use in privacy-related discussions:

Term	Meaning
Personal Information	Information, pertaining to a specific person, its preferences or its context.
PI-attribute	An *attribute* that contains *personal information*.

Term	Meaning
Subject (of a *PI-attribute*)	1. the person to which *personal information* pertains. 2. the person to which the information in the *PI-attribute* pertains.
Complex attribute	A coherent set of attributes, pertaining to the same *entity*.
Complex PI-attribute	A *complex attribute*, pertaining to a *person*, having two distinct representations: 1. the 'full representation', i.e. the *complex attribute* itself, which may contain sufficient information to identify its *subject*, and 2. the 'minimized representation', i.e. a *complex attribute* that is derived from the full representation, and does not contain sufficient information to identify its *subject*.

NOTES:

- *Personal Information* (PI) may or may not identify a person (within some *i-Scope*);
- Like all other *attributes*, every *PI-attribute* is part of some *i-Scope*.
- *Complex attributes* can be seen as a(n) (un)structured set of more detailed *attributes*. For example, an 'address' may consist of attributes 'street', 'number', 'zipcode', 'city', etc.
- A full representation of a *complex PI-attribute* can be reduced (into a minimized representation) by several means. One way is to remove part of the (detailed) *attributes*; the 'address' of the previous bullet may be reduced to only the 'zipcode' or 'city'. Another way is to obfuscate part of the *attribute*, e.g. a bank-account number could be obfuscated by only showing the last four digits.
- The (design) decision of how a full representation of a *complex PI-attribute* is reduced to its minimized representation is the responsibility of the *party* that is accountable for the *i-Scope* of which the *complex PI-attribute* belongs. This party is also accountable for the property that this minimized representation does not identify its *subject*.

3. MIM Context

This chapter specifies terminology and concepts that are necessary for describing contexts within which the MIM is to operate.

3.1. Parties, Actors and Actions

In the real world, there is a clear distinction between being responsible (accountable) for an *action*, and actually executing the *action*. For example, filling in an order and sending it to a manufacturer is executing some *action*, which can be done by a secretary. However, the managing director, who has had no part in filling it in or sending it away, is accountable for this *action*.

Someone that is accountable for *actions* is accountable without regard as to who executes them. For example, the Minister of Justice will be held accountable for *action*s that lead to the escape of convicts, regardless of the *actors* involved.

Conversely, different *actions* that are executed by a single *actor* may fall under the accountability of different *parties*. For example, if you may make a phone call with your mobile phone, it depends on the context who will be accountable (and hence picks up the bill). If (the SIM-card in) the phone is yours (personally), you will be paying for it. If you have a company phone, your employer will pick up the bill. If you make a collect call, the person that you are calling (or someone related to that person) will pay the charges. And if you have a pre-paid phone, the person that filled your pre-paid account (which theoretically could be anyone) pays for it.

These examples illustrate the importance of being able to distinguish between who is performing an *action* and who is accountable for this. To this end, we introduce the following terms:

Term	Meaning
Party	A person (or a limited group of persons) that is capable of being held accountable.
Action	Something that is done (see also Merriam-Webster online) by precisely one *actor*, for which precisely one *party* is accountable.
Actor	An *entity* (typically a person or a machine) that is capable of executing an *action*.

NOTES

1. A *party* may be referred to by another *identifier* than the name of the person. For example, role names such as ‘the Minister of Justice’ or ‘the manager of X’ refer to a person that can be held accountable, and hence identify a *party*. Also, the name of an organization (or department, or …) may be used to identify a *party*, as this implies the person(s) that are in charge.

2. An *actor* need not be a living person. It can also be a machine, in particular: a computer, as they, too, can execute *actions*. All actors will need to be identifiable at some point, e.g. using a username (for humans) or using server/service *identities*.

3. The *party* that is accountable for an *action* may, or may not, also be the *actor* that executes the *action*. This is in particular noteworthy in use-cases that focus on 'the end-user', where assumptions regarding the accountability of *action*s of such 'end-users' are not always made explicit (or verified).

4. People are usually thought of as being both an *actor* and a *party* for their *action*s. Be aware that this is not always the case. For example, small children are *actors*, but they are not a *party* as not they, but their parents are held accountable. Similarly, people that are put under guardianship remain *actors*, but their guardian is accountable for their *action*s. An example of the converse is when people die: then the are no longer *actors*, but they remain a *party* until all their obligations have been settled, the last will is executed and the heritage is handled. This, of course, requires (an) actor(s) that actually do this.

3.2. Sessions

A nice, short definition of the term '***Session***' is: a timespan during which *actors* interact with one another (Wikipedia has a more elaborate definition), during which they execute *action*s that have consequences for the participating *actors*. However, since an *action* is not only associated with one *actor* (who executes the *action*) but also with one *party* (that is accountable for (the consequences) of that *action*), executing an *action* within a *session* may have ramifications for all *parties* associated with that *session*.

To make this a bit more concrete, consider one *actor* that is a web-service that runs on some machine of a service provider (SP), and another *actor* that we will not further specify but refer to as 'the end-user'. Usually, a *session* starts when an end-user sends some request to the web-service, requiring the web-service to perform some *action*. Depending on the possible consequences the execution of such an *action* may have for the SP (the *party* accountable for such *action*s), the web-service may request the end-user to identify itself and provide proof of such identity (e.g. send a userid and password). The end-user evaluates the consequences of doing so, and decides whether to comply with the request or to stop communicating, thereby terminating the *session*. If the credentials are sent, the web-server executes the *action* of verifying the credentials (authentication). Both *actors* continue to send requests and responses to one another, and execute *action*s that affect each other (and the associated *parties*), until one of them decides to terminate the *session*. The important thing here is that an *action* will (or should) only be executed if the *party* that is accountable for this *action* has a sufficient level of confidence that the consequences of executing the *action* are acceptable for itself; if this *party* values the relationship with the *party* that is accountable for the *actions* 'at the other end' of the *session*, then of course he will take the consequences of the other *party* into account as well, but this is a business decision of the first *party*.

Note that the different *actors* that participate in a *session* each have their own perspective on this *session*. This is reflected by the fact that each *actor* assigns different attributes the (same) *session*, and usually is not aware of whether, or which attributes the other *actor* may have assigned to the *session*. This allows each *actor* to also (subjectively) view a ***session*** as a description of the context within which it communicates with some other *actor*, in terms of a set of so-called ***session-variables***[14] that represent properties that the *actor* has attributed to the *session*.

[14] In this document, we do not make any assumptions as to how session-variables are stored or managed. They can be stored in cookies, in local session variables (as in PHP), or by any other means.

We use the term *session* both for the objective perspective (where *session* is defined as 'a timespan during which *actors* interact with one another …' and the subjective perspective (where *session* is defined as 'a description of the context within which it communicates with some other *actor*') as they are merely two different perspectives of the same thing.

3.2.1. End-User Web-Services, Session Variables and Environment Variables

MOBiNET end-user web-services are web-services that communicate directly with an end-user (through its browser, or browser-like app). They do so through some kind of communication channel (e.g. using http(s) or something similar). An individual end-user web-service may communicate with multiple end-users at the same time. In order to separate the inter*action*s that it has with individual end-users, web-services establish *session*s, i.e. sets of meta-data pertaining to the communications channel.

An end-user web-service will use *session*s e.g. to store data about 'the other end' of the communications channel. Individual pieces of data pertaining to an individual session, are called ***session variables***. Such *session variables* exist for a purpose, and depending on what purposes are to be pursued, different session variables will be needed. Therefore, there is no standard specification of *session variables*. *Session variables* cannot outlive the session to which they belong. Here are some examples:

- 'sessionAccount', i.e. the *account* that was identified when *logging in*. The purpose for having such a variable is that every transaction in the session can be attributed to an *account*, which implies that the *party* that is accountable for the *actions* that the *actor* 'at the other end' of the *session* executes, e.g. sending messages with specific content, is unambiguously known (and can, theoretically, be sued). From a sessionAccount, other variables such as 'sessionGroup' (or 'sessionUser') and/or 'sessionParty' may be derived because every *account* pertains to a pair of (*Group*, *Party*) or (*User*, *Party*).

- 'sessionPermissions' and/or 'sessionRoles', i.e. the set of permissions and/or roles that has been activated in a *session*, based on the permissions and/or roles assigned to the *account* of the session user and the roles that the web-service can deal with[15]. The purpose for having such variables is to contribute to realize *access control*.

- 'sessionUserLocation' which hold the location (or area) of the end-user equipment of the *actor* 'at the other end' of the *session*.

- 'sessionServicePurpose' might be the purpose to which the communications and transactions in the *session* contribute. Such a *session variable* might be helpful for checking whether or not privacy constraints are satisfied.

Where *session variables* are defined to contain information about an individual *session* of a service, most often such information only pertains to 'the other end' of that *session*. The reason for this is that any information that pertains to 'this end' of the *session* would be duplicated for every *session* that a service has. However, it is also necessary to dispose of information about 'this side' of the *session*. This information is stored in what we define to be ***environment variables*** (of a service). An *environment variable* is just like a *session variable*, except that it belongs to a specific service (rather than a single session), and the information it contains thus pertains to 'this end' of all *sessions* of that service.

[15] Role-based access control (RBAC) is standardized by NIST.

Environment variables cannot outlive the (runtime) service to which they belong. Here are some examples:

- 'serviceID', i.e. an *identity* of the service, allowing it to inform services that it will be calling, as well as services that it is called from, of its properties.
- 'serviceCertificate', i.e. a PKI-certificate that a service may use to prove its *identity* to other services.
- 'serviceAccount', i.e. the *account* under which the service is executed. The purpose for having such a variable is that it is unambiguously clear who the *party* is that is accountable for that every action executed by that service takes part. An environment variable called 'serviceParty' may be derived from this.
- 'serviceLocation' might hold the location (machine, room, coordinates, area) of the processor or machine that executes the service.
- 'servicePurpose' might be the purpose that the service seeks to fulfill. Such an *environment variable* might be helpful for checking whether or not privacy constraints are satisfied.

3.2.2. Passing Information Between Web-Services

The whole idea of Service Orientation is that (end-user) web-services invoke other web-services, e.g. a payment service, so that they do not have to implement that functionality themselves. Calling another web-service is: setting up a *session* between the calling service and the called service, each of which will maintain a set of *session variables* that serve some purpose of the service using this *session variable*. This is (conceptually) the same as what is described in the previous section.

When an end-user web-service calls another web-service, this other web-service may not only be interested to know about what (end-user) web-service it was called from, but (depending on the purpose it pursues) also in the values of 'sessionUser' or 'sessionResponsible' of the end-user web-service. The values for 'sessionUser' or 'sessionResponsible' of the called web-service should then be 'inherited' from the calling (end-user) web-service.

It is a crucial ability for web-services is to propagate information. This ability has two parts:

1. Any calling web-service needs to obtain the information that is required by the called web-service, prior to passing it on. Such information may already be available, e.g. in the *environment variables* of the calling web-service, or in the *session variables* of the *session* that has led the calling web-service to call the called web-service. Otherwise, the calling web-service will need to access other sources of information, which could be other services (run by other *parties*), or the *actor* at 'the other end' of the *session*. What a service is capable of doing in this respect (or not), is part of the design of the calling web-service.
2. In order to obtain this information, any calling web-service needs to know what information the called web-service actually needs in its *session* with the calling web-service. Since web-service compositions may be very dynamic in MOBiNET, such information needs (of the called web-service) may not always be (completely) specifiable at design-time (of the called web-service), and hence may also be unknown at design-time of the calling web-service. Therefore, a dynamic means for specifying the information needs of the called web-service and communicating such

requirements to the calling web-service, is called for. We assume that this topic is addressed by the MOBiNET Service Directory component (WP3.2).

3.3. Accounts, Login and Identification

Within a *session*, messages are being exchanged between, and *actions* are being executed by the *actor* at every 'end' of the *session*. Executing an *action* may often only occur if a set of preconditions (e.g. access control rules) has been satisfied. Often, such conditions pertain to 'the other end' of the *session* and in order to evaluate such rules information about the *actor* and *party* at 'the other side' is required. Such information can originate from 1) 'the other side' (e.g. a postal address), 2) 'our own side' (e.g. the permission to access a service) or 3) a third party (e.g. a statement saying that 'the other side' has sufficient money in its bank account).

3.3.1. Accounts

We use the term ***account*** to refer to a set of *attributes* pertaining to an (*group*, *party*) pair, where a *group* is defined as a (non-empty) set of *actors*. Put a bit more elaborately: an *account* is a set of attributes that apply to either one *group* or one *party* or to the combination of both. Such *attributes* could be a permission (e.g. for accessing a service), a phone number (that may be used for sending SMS messages to one or more *actors* in the *group*, and/or to the *party*), an e-mail address, etc.

The term ***group account*** is commonly used to refer to an *account* for a *group* for which it is expected that multiple *actors* are in the *group*. *Group accounts* are sometimes considered bad practice because any *action* that is executed under a *group account* cannot be linked to an individual *actor*. On the other hand, they are advantageous because attributing a permission to a *group account* allows every *actor* in the group to do whatever the permission allows for.

The term ***user account*** is commonly used to refer to an *account* for a *group* that consists of precisely one *actor*, so that whenever an *action* is executed in a *session* under some *account*, the *actor* that executed the *action* is known. For the sake of completeness and consistency, we define the term ***user*** as a set of precisely one *actor* (note that in this definition, every *user* is a (special kind of) *group*).

We use the term ***session account*** to refer to the *account* (set of *attributes*) that represents properties of the *group* and/or *party* that is at the 'other end' of the session. In order to allow for unambiguous accountability, it must be ensured that within one *session*, no more than one *account* may be 'activated'. In other words, every *session* may have at most one *session account* throughout its lifetime.

3.3.2. Login

We use the term ***login*** to refer to the *action*, within (the context of) a *session*, where one *actor* sends a message to the other *actor* in that *session*, where this message contains (at least) a) an *account identifier* and b) some data that ensures that the message that is sent originates from the (*group*, *party*) pair that the *account* pertains to. We use the term ***login credentials*** to refer to credentials containing an *account identifier*. A common kind of *login credential* consists of a username (or userid)[16] and password, where the username typically identifies an *account* in the application that receives the message, and the password provides assurance that the end-user is whom he claims to be.

[16] Of course, a groupname or groupid may be used instead.

Note that while the term ‘authentication’ is commonly taken to mean ‘*authentication* of *login credentials*’ we have (already) defined it in a broader sense. Still, authentication in this more common sense is important, as when this is successful, this results in an *account* being identified (within the *account registration* that is used by the service that is logged into), and this *account* is subsequently used as the *session account* for the *session* into which the login took place.

Also note that it is a common, but understandable misconception that usernames (and the like) should always identify the a user (or a *person*, or an *actor*). A username identifies an *account* (within the context of a (set of) service(s)); it only also identifies an *actor* if the account is a *user account*. But even then, a username only indirectly identifies an *actor*, as the identification is a two-step process: the first step is to identify the account, and the second step is to identify the (*group, party*) pair, which, only in case the *group* is a *user*, identifies the *actor*.

3.3.3. Identification

Every web-service, regardless of whether it is an end-user web-service or not, must allow the calling web-service (or browser/end-user) to identify itself and must authenticate this identity. End-users may typically use usernames (userid’s) and passwords (U/PWs) or other means. Web-services (or web-servers) may use so-called service/server certificates of some public-key infrastructure. Other means are possible, too. *Logins* may be used for transferring the messages containing such *login credentials*.

In the ‘Identity Basics’ chapter we have already generalized the idea of *identifiers* and *identities*, so that they can also be used for other kinds of *entities* such as cars, on-board units, road-side equipment, IT-servers, IT-services, organizations, bills, insurance policies, etc.

3.4. Access

Access is short for Access Control and Access Management. We describe it here shortly as it is part of the context, but the MIM does not provide any functionality for this.

3.4.1. Access Control

Access control (of a web-service) is the ability (of that web-service) to decide whether or not to allow or deny access to its functionality whenever it is called from an end-user (browser) or (other) web-service. Allowing access implies that a session is created for this communications channel which is subsequently populated with appropriate session variables.

For end-user web-services (which to the end-user appear to be Applications), its 'owner' must set up and maintain an administration that contains all necessary information for its web-services to decide about allowing or denying access, in accordance to the business rules of this owner. Since such administrations depend on the business rules of the owner, a generic MOBiNET access control administration cannot be supplied.

A similar reasoning holds for web-services that are (exclusively) called by other web-services. Note that in order for them to decide whether or not to allow or deny access, it may be necessary to know the identity of the calling web-service as well as the Party that owns the web-service; of course, this depends on the business rules of the owner of the web-service that is to make the access control decision.

3.4.2. Access Management

Access Management for a web-service is the capability of web-service owners to decide what business rules their web-services must employ for making access control decisions. Deciding upon the ontology to be used for stating the business rules, as well as deciding which business rules should be used for making access control decisions, is really a business problem. Because of this business dependence, a generic MOBiNET access management capability cannot be readily supplied.

3.4.3. Authorization

Authorization and Access Control are terms often mistakenly interchanged. Authorization is the act of checking to see if a user has the proper permission to access a particular file or perform a particular action, assuming that user has successfully authenticated himself. Authorization is very much credential focused and dependent on specific rules and access control lists preset by the web application administrator(s) or data owners. Typical authorization checks involve querying for membership in a particular user group, possession of a particular clearance, or looking for that user on a resource's approved access control list, akin to a bouncer at an exclusive nightclub. Any access control mechanism is clearly dependent on effective and forge-resistant authentication controls used for authorization.

Access Control refers to the much more general way of controlling access to web resources, including restrictions based on things like the time of day, the IP address of the HTTP client browser, the domain of the HTTP client browser, the type of encryption the HTTP client can support, number of times the user has authenticated that day, the possession of any number of types of hardware/software tokens, or any other derived variables that can be extracted or calculated easily.

4. Entity Registrations

Within an *i-Scope*, *entities* can be attributed *properties*. For example, a person can be attributed with an age or a birthdate. The set of attributed *properties* (which we call ***attribute***s) is also called a ***record*** or 'profile' of the *entity*. Conversely, every *attribute* in a *record* is a property of one and the same *entity*. Within an *i-Scope*, a set of *records* pertaining to *entities* of a single *Type* make up a *registration*. It is logical that the *party* that is accountable for this *i-Scope* is also accountable for such *registrations*, and therefore also for any assignment of *attributes* to *entities*, i.e. the registered *attributes*. In short: a ***registration*** is a container for data that represents information attributed to *entities* that is valid within a specific *i-Scope*, and one specific *party* is accountable for the (in)correctness, (mis)use, (in)consistency, (in)coherence, and (non) up-to-date-ness of this information.

The information in a registration is used within the *i-Scope* for doing 'business reasoning'. For example, in a registration of customers, information about the time that somebody is a customer may be used to assign him a 'Gold' status. The business rules or constraints that - implicitly or explicitly - apply to the registered *entities* and *attributes*, are also part of the *i-Scope* that uses them. Within the *i-Scope,* it must be ensured that the contents of its registrations satisfy such constraints, because this helps to guarantee the registrations integrity and consistency, as well as coherence of information and reasoning. All this is not new knowledge: in practice, organizations are more or less familiar with this way of working[17].

An organization that 1) uses information of other organizations and 2) does not realize that these other organizations have other business rules and constraints, will find itself applying its own business logic to data that does not satisfy its own constraints, resulting in invalid business outcomes with possibly disastrous impact.

The MIM module helps *partner*s of the MOBiNET ecosystem to overcome such difficulties by providing an identity framework that takes these differences explicitly into account. In this chapter, we will focus on

- *registering entities*, i.e. storing information that the *party* attributes to such *entities* (e.g. account numbers, license plates);
- searching data within such *registries*, i.e. specifying the requirements that need to be satisfied in order to find such data;
- communicate about *entities*, i.e. sending and receiving messages containing data pertaining to *entities*, in such a way that the meaning that the sending *party* attributes to this data is consistent and coherent with the meaning that the receiving *party* attributes to such[18].

[17] a 'contaminated' or 'polluted' registrations (often: databases) are registrations that contain data that violate (i.e. do not satisfy) such constraints. Businesses want to keep their registrations 'clean' so that the business reasoning that is done within business processes, is valid. Polluted registrations may lead to invalid reasoning and unwanted consequences that may be detrimental to the business.

[18] In order to be able to address such semantic interoperability issues, it is important to clearly distinguish between 'data' and 'information' (or 'meaning').

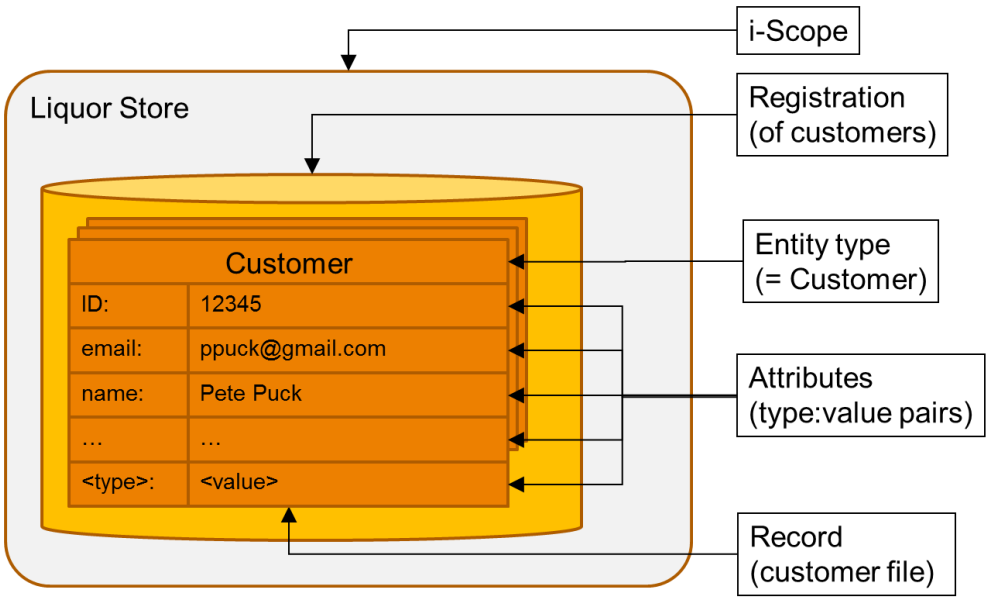
4.1. Entity Registrations

An *entity registration* is a *registration* of *entities* of some *type*. We will first consider the regular case, and continue with the privacy aware one.

4.1.1. Regular Entity Registrations

A (nominal) Entity Registration usually contains *records*, each of which contains information (in terms of *attributes*) about a single *entity*. A database may contain multiple entity registrations. A single *party* is accountable for whatever happens with the database. Consequently, it is also accountable for every entity registration within that database. Finally, any data in the database must be represent information in an *i-Scopes* for which that *party* is accountable.

An example for a customer registration of a *party* contains *records* of *entities* that the organization has classified as being a customer (this can also be organized in another way, e.g. in terms of database tables, etc.). This is visualized in the figure below:



There are some important considerations to take into account, in particular if multiple *i-Scopes* are involved.

First, ***The validity of business reasoning within an i-Scope with respect to an entity type (e.g. 'Customer'), depends on the criteria that distinguish entities of that type from entities that are not***.

For example, consider a country that has a law stating that anyone that is over 18 years old can be brought to justice and anyone under this age cannot. Also, assume that there is a liquor store in that country that requires its customers to be 20+ years old. In this case, the liquor store can correctly infer from this that 'Every customer can be brought to justice', and define its business processes based on this (inferred) business rule. However, if the liquor store did not have this age constraint for its customers, the business reasoning would fall apart and the liquor store can come into the position where it wants to sue a customer, but cannot do that because (s)he is under 18.

This consideration is of particular relevance when multiple *i-Scopes* use the same name to identify an *entity type*. For example, many organizations use 'customer' to identify an *entity type*. However, it is common practice that each *i-Scope* uses its own criteria to distinguish between entities that are customers and entities that are not. This allows one *party* to attribute me the property of 'customer', while another *party* would not do so, which in practice is nice to have. But difficulties may arise if one *i-Scope* were to apply its own business reasoning (using its own criteria for customers) on people that have been attributed the 'customer' property by another organization' (i.e. entities that satisfy the constraints for 'customer' of another *i-Scope*). Such reasoning may be invalid and lead to (serious) business incidents.

In order not to prevent such incidents from happening, one should be alert to the fact that ***while the name (identity) of an entity type (e.g. 'Customer') does identify such an entity type, it does NOT specify (define) it***. In other words: customers of organization 'A' must always be treated distinct from customers of organization 'B', unless 'A' and 'B' have some agreement stating the requirements that both of them will apply (and enforce) with respect to customers.

Of course, this holds for every other *entity type* as well.

4.1.2. Privacy Aware Entity Registrations

A ***privacy aware entity registration*** is an *entity registration* about *persons*, containing *(complex) PI-attributes* for such persons, and that is split into two parts:

1. a ***minimized registration*****,** contains the 'minimized representations' of the *(complex) PI-attributes*;
2. a ***full registration***, contains the 'full representations' of the *(complex) PI-attributes*;

where every *complex PI-attribute* in the *minimized registration* has a counterpart in the *full registration*, and vice versa. Note that since both registrations contain information of a single *i-Scope,* the *party* that is accountable for the *i-Scope* is responsible for both registrations.

In the liquor store example, the content of the records about people would be minimized, meaning that when such information is registered, the actual *(complex) attributes* are placed in the *full registration*, a 'minimized *complex attribute*' is derived from that and subsequently sored in the *minimized registration*. The minimization strategy, algorithms etc. for each of the *complex PI attributes* is under the responsibility of the liquor store itself; such a strategy could be to store 'Peter P' in order to preserve the alphabetical ordering at a minimal capacity. A very cautious strategy would be to return 'anonymous' indifferently for the name attribute of any entities of this type, but this could lead to a clear degradation of the service when users don't provide their consent. This balance has to be tuned and assumed by the individual *parties*.

4.2. Obtaining Information About Entities

There is a difference between searching for data or searching for information. Searching for data is a purely technological matter and is concerned with matching characters, numbers, bits and bytes. An example is: searching for text within a string. Searching for information is quite different, because meaning must now also be taken into account. For example, comparing the value of two amounts, e.g. one expressed in US$ and another expressed in Euro's is not simply a matter of comparing bits and bytes. And if one day, the value of an amount expressed in US$ is less than some (other) amount

expressed in Euro's, a few days later this may be the other way around. Also, a single data element may be assigned different meanings by different parties. Previous chapters have examples of this.

Searching for information requires one to thoroughly think about the registration(s) in which to search. First, you must be sure that the data in such registrations actually represent the required information, or that such information can be derived from this data. When searching your own registrations, this should be no problem. However, when querying registrations of other *parties*, it may help if the other *party* has made a proper (semantic) description of such data available. Such descriptions are basically the *Type* criteria for *identities* and *attributes*. For example, if a person claims to be 'trustworthy', you may want to know who says so, i.e. in which *i-Scope* this is a property that is attributed to an *identity* for that person. From the criteria that this *i-Scope* uses for 'trustworthy' and the criteria is uses for the used *identity*, you may judge what this means for you, e.g. you may decide whether or not to trust this person based on the (valid) claim from this *i-Scope*.

However, you need a bit more than just the *Type* criteria; you also need to ascertain the trustworthiness of the *party* with respect to its truthfully providing information about the *Type criteria, identities* and *attributes* that he provides. Thus, if a *party* receives data that allegedly pertains to some *entity*, references to the *i-Scope* (or *party*) within which this data is held to be true, may help the receiving *party* to establish the information associated with this data and ascertain its value (trustworthiness).

4.3. Communicating About Entities

Whenever a *party* wants to obtain information about an *entity* to from another *party*, it needs to ascertain that the *identity* that it uses to identify the *entity*, matches the *identity* that the other *party* uses, i.e. that both *identities* refer to the same *entity*. Conversely, the *party* that conveys such information may also need to ascertain that the *identities* match. Ascertaining that two (or more) identities that belong to different *i-Scopes* actually refer to the same *entity* is called the matching problem.

The matching problem occurs quite often. Consider the case where someone wants to hire a parking lot for an hour, and wants to reserve and pay (with a credit card) for this place online. Also, he wants to get a discount because he is a member of some mobility club. The various *parties* that participate in this service (the parking lot owner, the credit card company, and the mobility club) will need to use *identities* for the car, the person, the credit card and perhaps others. Whenever one party sends an *identity* to another party, both must be sufficiently sure that the *entity* that it identifies with one *party* is the same *entity* as is identified by the other *party*.

Not properly solving this matching problem can result in undesired consequences: the credit card account of the wrong person may be charged, discounts may be wrongly given or denied, etc. The bigger the unwanted consequences can be, the more effort parties must place in making really sure that this problem is solved.

While the matching problem is in principle the same for every pair of *parties* (they must ascertain that their communications refer to a single *entity*), it can differ as well, because there are so many ways in which the assurance can be obtained.

An often suggested solution for this problem are the so called 'global' *identities*, i.e. *identities* that are known by multiple *parties*. The term 'MOBiNET identity' that has been coined seems to be an example of this. The perceived advantage is that service providers in the *MOBiNET ecosystem*, could use this *identity* without translation on their part, making it easy to communicate to other *partners*. However, as

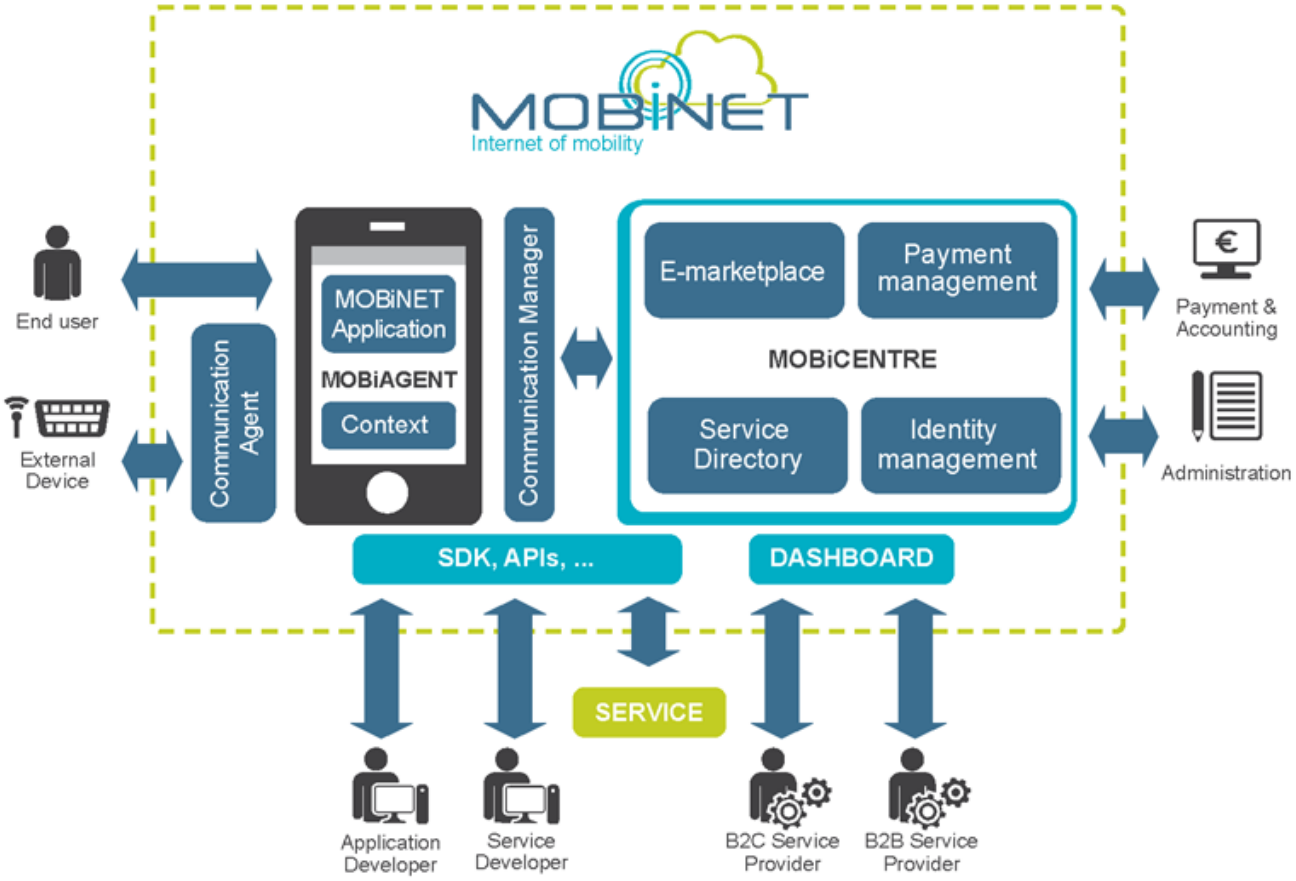
with all ‘global’ *identities*, the problem may be diminished, but it will not be solved. The reason for this can be found when we consider what happens when a *party* joins the *MOBiNET ecosystem*: there is no guarantee that the users of that *party* can be assigned a *MOBiNET identity* because the *Type* criteria for entities that may be assigned such an identity may not overlap all users of that party. At the time of writing, we cannot be sure of this because the *MGA* has not yet established such criteria. But we have seen this happening in many cases ‘in the wild’, e.g. with the BSN, which identifies Dutch citizens, and when it was attempted to use within the realm of education, it became clear that also non-citizens sometimes have a right to education, rendering the BSN useless as a ‘global’ identifier.

Another drawback for global identities is that it may jeopardize privacy. If all sorts of *parties* may identify the same person (or other *entity*) with a single *identity*, all sorts of information can be linked between the various *parties*, without the users knowledge or consent.

Another solution for the matching problem can be found in what we call a ‘translation service’. This service, that will be defined further on, enables *partners* to obtain a (plaintext or encrypted) *identity* that is useable in an *i-Scope* of another *partner*, for an *entity* that it has its own *identity* for. Thus, it can use its own *identity* when referring to *entities* within their own *i-Scopes*, and use the translation service to convert it into *identities* for other *partners*.

5. MIM Features and Capabilities

The MOBiNET Identity Module (MIM) is one of the core components of the MOBiCENTRE, as illustrated by the figure below.



A core principle of MOBiNET is to provide users a common identity that will be used ubiquitous within the MOBiNET. On the one end, this enables end-users to have single user subscription that is valid for all services in the realm of MOBiNET; and on the other hand, it enables service providers to do business with users that have no prior affiliation but are accredited through MOBiNET.

In general a number of different user profiles have to be considered in the MOBiNET framework, namely:

- Business *actors* (e.g. service/content providers: B2B and B2C);
- Developers (applications/services SW)
- MOBiNET framework administrators
- Private end-users

A suitable methodology for dealing with these different identities must be implemented in the framework. The module responsible for these task is referered to as the MIM (MOBiNET identity Module).

MIM enables any MOBiNET *actor* to have authorized access to any MOBiNET services allowed by its own identity profile. MIM might consist of the following components ensuring that all privacy and security best practices are addressed and providing:

- Authentication services for every MOBiNET user to every MOBiNET service (i.e. SSO within the MOBiNET realm), based on existing subscriptions that users (already) have with members of the MOBiNET Supplier Community (e.g. service providers).
- Identity attributes of authenticated MOBiNET actors for using MOBiNET services.
- Identity attributes management and update from authorized *actors*.
- Standards-based interface with MOBiNET partners who aim to serve as Identity Provider (e.g. OpenID).
- Authorization services to grant access to MOBiNET services (e.g. OAuth).
- Single sign-on for end-users (e.g. based on federated identity).

The authentication and authorization features provided by the MIM are an enabler to the access to the MOBiNET Service Directory which is developed by WP3.2. As a matter of fact, any access to Service Directory has to be subjected to proper identity management by the MIM. Whenever applicable, any needed identity conversion or association among identity attributes has to be carried out by the MIM.

One of the concepts of MOBiNET is the establishment of a business to business e-Marketplace that brings together different service providers and lets them benefit mutually through sharing their services. This requires a *reliable, secure, and controlled mechanism for authentication and authorization* (carried out by the MIM) in order to access to the discovery of existing services.

MIM also deals with Privacy issues by maintaining an identity repository for any MOBiNET actor and overning the privacy policy and access policies.

see comment

The MIM may be called by MOBiNET services, i.e. services that are part of the *MOBiNET ecosystem*, for various purposes. The MIM has several features and capabilities for the concrete, technical support of such purposes, including:

1. A facility that allows a *partner*[19] to publish an URI that points to a service of that *partner*, that is capable of disclosing *attributes* of an *i*-Scope of that *partner* for *entities* registered by this *partner*. The *partner* that has published such an URI must control the service identified by this URI, and ensure that the disclosed information is correct (both syntactically and semantically), consistent (within its own *i-Scope*), valid, up-to-date, etc. Access control constraints for this service include constraints required by the disclosing *partner* and/or the *MGA* and perhaps other stakeholders (to be specified).

   The purpose of this capability is to facilitate *partners* in using (e.g. reasoning with) information that other *partners* have.

[19] Note that the *MGA* is also a *partner* that hence may use this facility.

2. A facility that allows a *partner*[19] to register (*identifier*(s) for) *entities* of any *entity-type*. Registration of an *entity* by a *partner* requires the *partner* to register (at least):

    a) (A reference to) the *entity-type* of the *entity* that is being registered[20].

    b) A *persistent identifier* for that *entity* in an *i-Scope* of that *partner*.

3. A facility that allows a *partner*[19] to register *attributes* for any *entity* that the *partner* has registered an *identity* for (and hence is in an *i-Scope* of that *partner*). The MIM will record the *i-Scope* of that *partner* in which the attribute has meaning. The *partner* that registers such *attributes* is responsible for keeping them correct (both syntactically and semantically), consistent (within its own *i-Scope*), valid, up-to-date, etc. Such attributes become available to all *partners* under access control constraints decided upon by the *MGA*.

4. A facility that allows a *partner*[19] to translate an *identity* in one of its *i-Scopes* to an *identity* for one of the *i-Scopes* of another *partner*, with the property that both *identities* refer to one and the same *entity*. Characteristics of this facility are that:

    a) It supports privacy, in the sense that an *identity* in an *i-Scope* of a *partner* is (by default) only made available to another *party* when it is (salted and) encrypted. The effect of this is that the MIM does not violate confidentiality of (the plaintext) *identities* from an *i-Scope*.

    b) A *partner* that wants to refer to an *entity* using MIM functionality must use an *identifier* that it has registered itself with the MIM.

    c) A *partner* that wants to communicate with another *partner* about an *entity*, using MIM functionality, can only do so if the other *partner* is willing to register, or already has registered, that *entity*.

    The purpose of this capability is to facilitate the communication of messages between two *partner*s, in which a distinct *entity* is referred to, by solving the matching problem and retaining privacy.

5. A facility that allows a *partner*[19] to register person information into a Privacy Aware Entity Registration (under its own accountability), allowing (other) *partners* to query the registration, obtaining either obfuscated (privatized) information from the minimized registration or, if they need it for a purpose that the *subject* has approved, obtaining the full information.

We emphasize that the MIM does NOT take any responsibility for the meaning of the data (URI's, *identities*, or *attributes*) that are stored – the *partner* that has registered such data is accountable for that. The MIM also does not destroy any data that has been registered; this should be done by the *partner* that has registered the data. There is one exception: when a *partner* leaves the *MOBiNET ecosystem*, all registered data for which it was accountable, will be deleted.

This property of neutrality with respect to the content of registrations is particularly relevant with respect to privacy protection. Since *partners* are equipped with the Mobinet Privacy Framework, they are in a position to manage privacy in their own *i-Scopes*, there is no need for the *MGA* to be(come) accountable or liable for storing or transporting private data except from its own *i-Scopes*. Further expansion

[20] This may be a location within the realm of the published URI (see capability 1).

mechanisms may be established point to point by the concerned parties to enforce a accountable processing of the exchanged data. Supported by the MOBINET Privacy Framework, such enforcement restricts the MOBINET responsibility to:

1. Manage private information in the MIM module
    - Implement the privacy by default for personal information registration
    - Implement the privacy by consent strategy which consist in the following sequential steps:
        1- Model roles and purposes of the MIM module
        2- Provide a consent editor for each registered *i-Scope*
        3- Optional – register the user deployed consent as an entity field for the registered *party*
        4- Enforces the consent at MIM *i-Scope* private fields processing time with respect to the right consent.
2. Deliver data and keep track of their originating *i-Scope*. These ones are responsible for their own privacy enforcement, possibly supported by the MOBINET Privacy Framework.

6. Examples

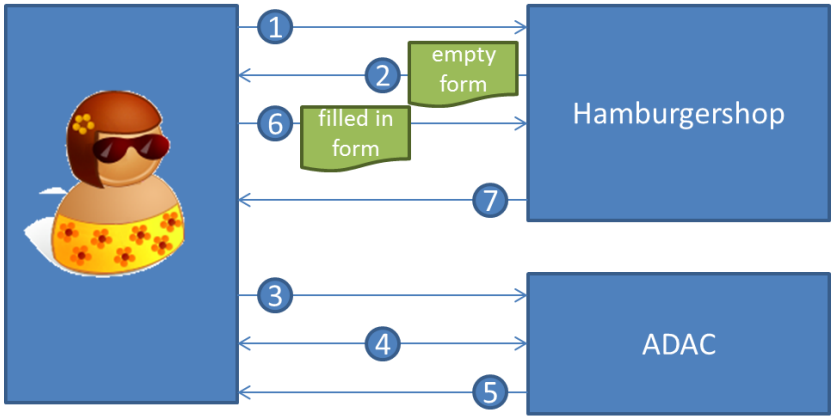
This chapter contains some examples of how issues may be resolved using the MOBiNET Identity Framework.

6.1. The Hamburger Use Case

A company called 'Burgerstore' sells burgers, with a 5% discount for ADAC members. How does the Burgerstore know that you are an ADAC member and can decide for the discount? A similar use case is for VOLVO services that become cheaper if a user is member of ADAC (or some other party).

6.1.1. How it works

The task of the Burgerstore is to sell burgers upon the request of a user (customer) and receive payment for that. In this use-case, we are interested in finding out how the Burgerstore becomes capable of computing the payment due if he decides to provide a 5% discount if the user is an ADAC member. This means that Burgerstore must be able to ascertain whether or not this is the case. In the real world, Burgerstore employees would ask the customer to show their ADAC member card. The below figure shows how we can do pretty much the same thing in the electronic world.



Whenever the user starts a *session* by visiting the Burgerstore website (1), the Burgerstore sends a form (2) in which the customer can fill in what kind of burger(s) he wants. The (electronic) form also contains the logo of the ADAC and clicking on this logo starts another *session* (3), this time between the user and the ADAC Identity Provider (IdP) service site, requesting the ADAC to ascertain that this user is in fact an ADAC member[21].

When the ADAC IdP service receives the request, it will want to *identify* and *authenticate* the *user*, which it can conveniently do by requesting the user to send its ADAC *credentials* (4, 5), or any other way that the ADAC IdP service can handle to *identify* the users ADAC *account*. For example, the ADAC IdP service may allow the user to provide Facebook, Twitter, Google, or other credentials – perhaps even *MOBiNET credentials*. After successful *login* (*authentication* of the user-supplied *credentials*) the ADAC IdP service can determine from the *session account* whether or not the user is an ADAC member (and

[21] For several reasons, Burgerstore will send the session id of this session with the request to ADAC.

also: which member it is). If the user is a member, the ADAC IdP service issues a *claim* asserting that this user is an ADAC member[22], and sends this claim back to the user's browser, storing it in the form that the user is preparing for the Burgershop.

The user completes the form and sends it to the Burgershop (6). The burgershop processes the order, and computes the total amount to be paid. If the form also contains a valid claim[23], 5% is deducted off the total amount, and the bill is issued to the user to be paid (7).

6.2. Using Identities of Other Partners (Allianz Use Case)

In this use case, a car owner (CO) has a contract with an insurance company (IC) for a so-called 'pay-as-you-drive (PAYD) insurance', which allows everyone - including bad drivers - to be appropriately insured. There is a separate hardware device in the car, that measures how the car is driven, transmitting telematics data (e.g. through GPRS) to a collection service of a telematics data service provider (TDSP). Once every month, IC contacts TDSP to get the telematics information of the car in order to compute the fees that the car owner will have to pay the next month(which may be settled by the MOBiNET payment module, but that is not (yet) described here).

6.2.1. How it works - the Insurance Company (IC) Point of View

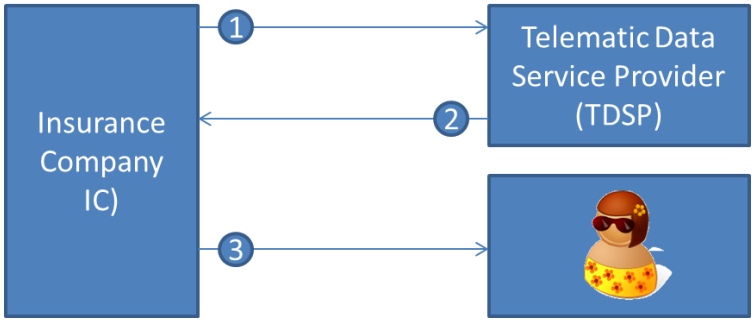
Consider an arbitrary PAYD contract between IC and a contract party. Every month, IC must compute a 'PAYD fee due' for this (and every other) PAYD contract, and submit a payment request to the contract party. In order to do this, the contract must identify the telematics HW box that is within the car, but not the car itself. After all, if there is an accident, the telematics data of the HW box will corroborate that it was indeed part of the accident (location, acceleration and other data). If the box was not in the accident, IC won't cover the costs of the accident.

Every month, the IC must send a request to the TDSP that owns the HW box referred to in the PAYD contract, asking for the telematics data of that HW box, providing any additional information that the TDSP has specified that it requires in order to handle the request. Please refer to the section ***How it works: the Telematics Data Service Provider (TDSP) Point of View*** to find out what this information is and why it is needed.

IC may get the telemetric data from TDSP, or not. In the latter case, IC may decide to set the fee to the maximum, because there is no evidence that it has been driven carefully. If IC gets the telemetric data, it can decide what the fee can be based on the information and any other information it has.

[22] part of this claim is the session id that Burgerstore has sent, so that Burgerstore may verify that this claim in fact belongs to this session and was not obtained from another session, as this would allow the re-use of this claim in arbitrary other sessions and provide the opportunity for non-ADAC members to obtain a discount as well.

[23] this means: the claim contains the session-id, a statement that the user is an ADAC member, and a digital signature by the ADAC (IDp Service).



The protocol is as follows:

1. IC sends service request to TDSP for obtaining telematics data of a specific HW box. (IC obtains the various identifiers from the PAYD contract)
2. TDSP verifies whether it owns the HW box identified in the request, and whether its user has registered a permission indicating that IC may use the telematics data. Only if there is such a permission, TDSP sends the requested data.
3. With or without the telematics data, IC can compute the fee for the next month's insurance, and sends a payment request to the PAYD contract party

Thus, the only information that IC needs to send a payment request for the 'fee due' of a contract, is:

- The party to the PAYD contract and the address at which the request has to be sent (post, email, fax, anything else…)
- An identifier for the hardware box, which is part of (and hence stored in) the PAYD contract
- The TDSP that owns the hardware box (may also be part of the contract)
- A service endpoint owned by the TDSP where requests for telematics data can be sent

6.2.2. How it works: the Telematics Data Service Provider (TDSP) Point of View

The TDSP owns a set of HW boxes that it issues to various parties. For every HW box that the TDSP owns, it has registered the party that uses this box (max. one party for every box).

In order to add value to the box, it will process requests by other parties, such as insurance companies, for telematics data of HW boxes. However, such data is only provided if the party that uses the box has permitted that party that requested the data to use such data. Otherwise, a request for telematics data will be declined.

Any payment that either the user of the HW box or the party requesting data should make to the TDSP is outside the scope of this use-case. Also note that it is not relevant for this use-case how the HW box sends its data to the TDSP.

Thus, the only information that TDSP needs to process requests for telematics data are:

- The parties that use its HW boxes

- A registration of permissions that such parties provide to still other parties for using the telematics data that its HW box produces.

6.2.3. How it works: Information view

Based on the above, we can distill information requirements for the various stakeholders:

Requirements for TDSP registrations:

1. Every HW box owned by the TDSP shall be used by at most one party (called the HW box user).
2. For every HW box owned by the TDSP that is used by a HW box user, the TDSP shall register this HW box user.
3. The TDSP shall provide HW box users to register and unregister one or more permissions, each of which allows a specific third party to obtain telematics data for a specific HW box that the HW box user actually uses.

Requirements for IC registrations (for PAYD contracts):

1. Every PAYD contract is for a specific, single car (that is insured).
2. Every PAYD contract refers to (at most) one HW box, using an identifier that is processable by the TDSP that owns the box.
3. Every PAYD contract that refers to a HW box must contain an identifier of the (service of the) TDSP owning the HW box (that services requests for telematics data)
4. Every PAYD contract refers to the contract party, i.e. the party that has (1) insured the car that is mentioned in the PAYD contract and (2) has specified the HW box identifier to be used in the PAYD contract and (3) will be paying the insurance fees.

6.3. User Registration

In principle, users are registered with *partners*. The Identity Provider (IdP) services of such *partners* are used to *authenticate* users in *sessions*, examples of which have been given earlier. However, it is conceivable that an IdP is operated under the accountability of the *MGA*. Then, the *MGA* would need to define processes for registering users, querying such registries, keeping the contents of such registries up-to-date, consistent and coherent, and cleaning it up when necessary. In this section, we give examples of ways in which users may be registered with such a *MGA* IdP.

6.3.1. Using Existing Credentials

Purpose
In order for users to be able to use MOBiNET services, they must be registered. The advantage of being registered with the MGA IdP is that they can use MOBiNET facilities independent from being registered with a (commercial) *partner.*

Prerequisites
The partner must have a link to the *MGA* IdP on its website (e.g. in the form of the MOBiNET logo)

Protocol
1. User goes to the website of the partner, and clicks on the MOBiNET link (logo) to obtain a MOBiNET service (or a partner service that uses a MOBiNET service, that will after some time result in a call to the MOBiNET service).

2. MOBiNET sends a request to the partner service for a user identity as issued by the partner.

3. The partner service either sends the requested user identity to MOBiNET (after having authenticated the user in the way that is usual for that partner), or it refers the user to the MOBiNET login page where it can enter its MOBiNET U/PW, effectively obtaining the user identity as issued by MOBiNET.

4. If MOBiNET has obtained a user identity, it checks whether the user identity is already registered with the MIM. If not (it then must be an identity that is issued with the partner), it creates a new MOBiNET identity, and associates it with the identity that the partner has issued.

6.3.2. Using *MGA* IdP

Purpose
In order for users to be able to use MOBiNET services, they must be registered. The advantage of being registered with the MGA IdP is that they can use MOBiNET facilities independent from being registered with a (commercial) *partner.*

Protocol
1. User goes to *MGA* IdP website and requests to be registered

2. *MGA* IdP sends a form where the user needs to fill in:

   a. Username

   b. Password

   c. Other information (to be decided by the *MGA*)

3. User sends requested information

4. *MGA* IdP processes this information in order to ascertain the truthfulness (validity) of the provided information. This processing may involve additional protocol steps with the user or other parties (e.g. sending a message to an email address as a means to ascertain the validity of a supplied email address).

5. *MGA* IdP sends either a ‘Welcome’ message (registration successful) or ‘Error’ message (registration failed)

6.4. Privacy Framework

The Privacy Framework is a set of tools providing any party to reach, at a very low cost, the following situation.

- Any record attribute conveying personal data is minimized at the time of its registration in the managed *i-Scope*. In this example, the private value storage is delegated to a third-party service (named Minexp).

- When a processing need occurs within the *i-Scope* itself, this processing need must be characterized in terms of purposes (coming from the execution context) and roles (based on session management). This characterization may then be interpreted in front of the data subject consent. The first step of this interpretation is to know which level of visibility is granted in such a context for the considered private data.
    - If no expansion is granted, then the application will have to process the minimized value.
    - Else, the third-party service is called for expanding the value to the granted level of visibility.

When using the MOBINET Privacy Framework on the MIM module itself, the clear separation of private fields from public ones must be managed by the MGA for MIM entities. When such a privacy by default enforcement has been conducted, any subscribed user would be in position to express a consent for the MIM i-Scope. Be this consent stored in the MIM itself is still an open question.

The following sequence diagram considers the positive case where the field processing is granted for the applicative context (purposes / roles) encountered at access time.

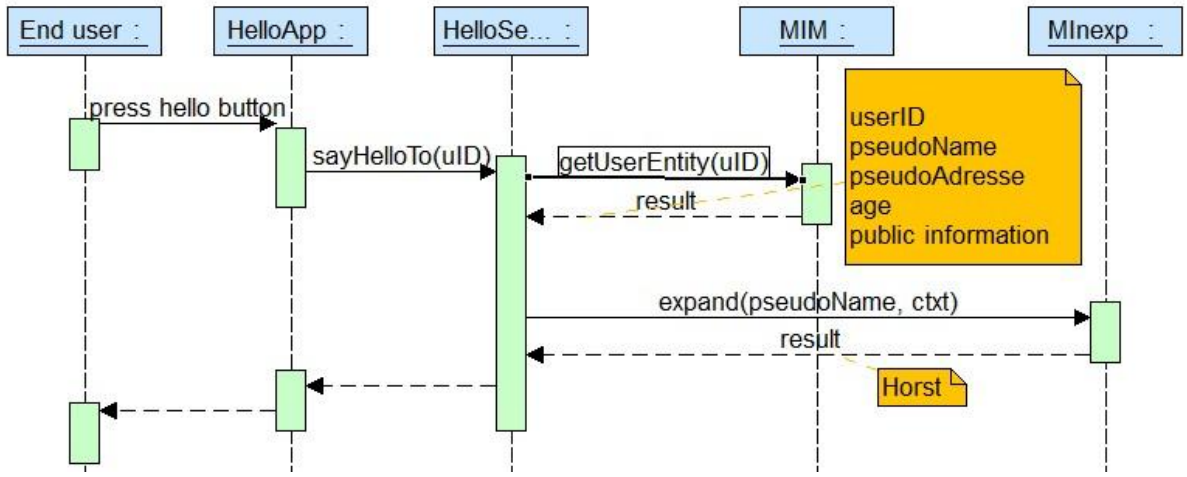


Figure 1 - the sequence of a private field expansion at processing time

When looking into more details the expansion operation, it appears clearly that the decision of expansion must occur prior to the expansion per-se. The decision service responsibility is to load the up-to-date consent provided by the data subject, and then to offer the service of deciding if an applicative context deserves a private field expansion or not. In case of positive answer, this could lead to the following sequence:

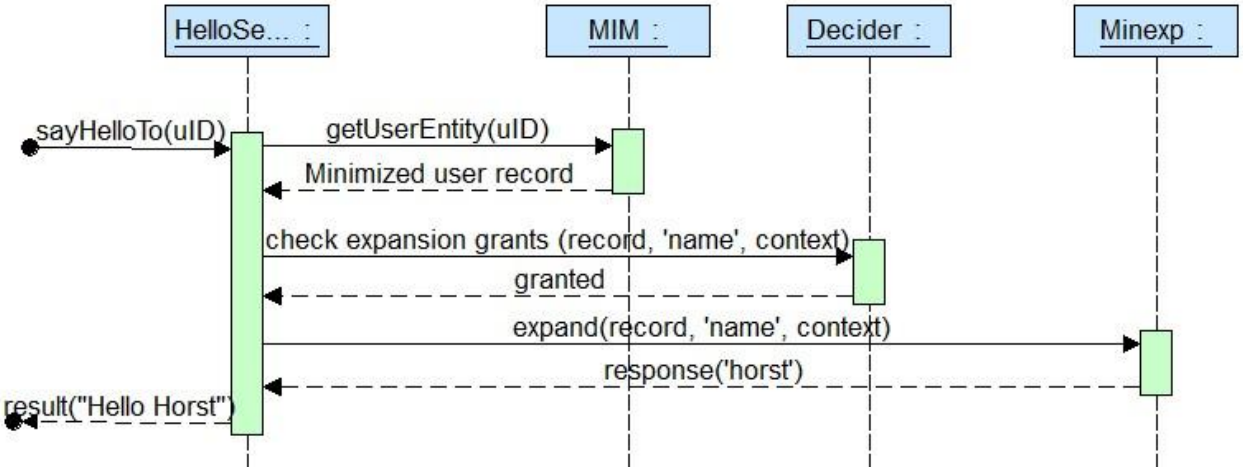


Figure 2 – The expansion decision is based on the user consent interpretation at processing time.

Annex 1 – MIM Glossary

The following table contains a description of the terminology used within the context of the MIM. In this table,

- any *italic term* has the meaning as defined in this table
- any term between angle-brackets '<' and '>' is a symbolic placeholder for whatever is between these brackets. For example, <*Party*> is a placeholder for some *party*

Term	Meaning
<Party> Identifier	An *identifier* consisting of attributes that have been assigned by <*Party*>.
Account	A set of *attributes* pertaining to an (*group, party*) pair within a single *i-Scope*.
Account registry	A set of *accounts* within a single *i-Scope*.
Action	Something that is done (see also Merriam-Webster online) by precisely one *actor*, for which precisely one *party* is accountable.
Actor	An *entity* (typically a person or a machine) that is capable of executing an *action*.
Assertion	(a sequence of *symbols* that represent) a statement uttered by (or on behalf of) a *party*.
Attribute (of an entity)	A type-value pair that some *party* has assigned to an *entity*. See also: *Issuer* (of an *attribute*).
Attribute type	The type of an *attribute*.
Authentication	The assessment (by some actor) of the authenticity (truthfulness according to the *issuer*) of (the *claims* in) *credentials* that the *actor* has received within the context of a *session*.
Claim	An *assertion* by some *party*, that this *party* assures is true.
Complex attribute	A coherent set of *attributes*, within some *i-Scope*, pertaining to a single *entity*.
Complex PI-attribute	A *complex attribute*, pertaining to a *person*, having two distinct representations: 3. the 'full representation', i.e. the *complex attribute* itself, which may contain sufficient information to identify its *subject*, and 4. the 'minimized representation', i.e. a *complex attribute* that is derived from the full representation, and does not contain sufficient information to identify its *subject*.
Credentials	A set of *claims* of a single *party*, the information in which (a) may be used by another *party* for some business purpose and (b) can be ascertained as being authentic by this other party.
Entity	(A record of) something that exists - see also the definition of Merriam-Webster).
Entity Attribute	A type-value pair that has been assigned (within a specific *i-Scope*) to the *entity*. See also: *Issuer* (of an *attribute*).
Entity-type	A class of *entities* that is defined by a criterion (that distinguishes members of this class from non-members) that is governed by a specific *party*.
Group	A (non-empty) set of *actors* associated with a single *party*.

Term	Meaning
Group account	An *account* pertaining to a (*group*, *party*) pair, where it is explicitly expected that the *group* may consist of more than one *actor*.
Identifier (for an entity)	A (non-empty) set of *attributes* each of which is attributed by a single (and same) *party* to a single *entity* of a specific *Type*, with the property that this *party* has not attributed the same set of *attributes* to a different *entity* of that same *Type*. Note that under this definition, every set of *attributes* for a specific *entity* is an *identifier* provided there is no other *entity* (of the same *Type*) to which all of these *attributes* apply simultaneously. Thus, identifiability is a property of a set of *attributes* rather than of an *entity* in its own right. (See also: '*Persistent identifier*').
Identify (an entity)	1. The *action* where an *actor* specifies an *identifier* (for some *entity*). 2. The *action* where an *actor* claims that a *symbol* is an *identity* for some *entity*.
Identity (of an Entity)	1. A *symbol* that (1) identifies and uniquely represents an *entity* within an *i-Scope* and (2) satisfies the syntactic criterion as specified in type; in this meaning, an *identity* is also an *identifier* (for that *entity*). 2. A *symbol* of which an *actor* claims that it has the property as stated under (1). 3. The union of all *partial identities* of that *entity*.
Identity scope	See: *i-Scope.*
Information scope	See: *i-Scope.*
i-Scope	A scope, within which an information model of (usually a small part of) the real world exists, expressed in terms of *symbols*, relations (between them) and constraints (expressed in terms of these *symbols* and relations), and for the coherence, consistency, correctness, currentness etc., a single *party* is accountable.
Issuer (of an attribute of an entity)	The *party* that has assigned the *attribute* to the *entity*.
Issuer (of an identifier)	The *party* that has assigned every *attribute* that the *identifier* consists of.
Login	an *action*, within (the context of) a *session*, where one *actor* sends a message to the other *actor* in that *session*, and this message contains (at least) (a) an *account identifier* and (b) some data that ensures that the message originates from the (*group, party*) pair that the *account* pertains to.
MIM	MOBiNET Identity Management or MOBiNET Identity Module.
MOBiNET Ecosystem	The scope within which *partners* cooperate.
MOBiNET Governing Authority	A *party* that, amongst other things, governs syntax and semantics of types (including entity-types) that are to be used throughout the MOBiNET ecosystem.
Organization	(Name of) a *party*, usually consisting of multiple people, where one (or more) people (usually referred to as the board, or the director(s)) execute all *actions* that are related to bearing the *party's* accountability.
Partial Identity (of an entity)	The set of all *attributes* that a single party has assigned to a single entity.

Term	Meaning
Partner	A *party* that takes part in the MOBiNET ecosystem, and as such abides by the ecosystem laws/rules/policies as set forth by the *MOBiNET Governing Authority*.
Party	An organization or (a group of) *person(s)* that is capable of being held accountable.
Persistent identifier (of an entity)	An *identifier* whose *issuer* guarantees that the set of *attributes* that make up the *identifier* will retain its identifying property throughout at least the lifetime of the *entity* that it identifies AND at least the lifetime that this *entity* needs to be referable.
Personal Information	Information, pertaining to a specific person, its preferences or its context.
PI-attribute	An *attribute* that contains *personal information*.
Record	The set of *attributes* that have been attributed to a single *entity* in a *registration*.
Registration	A container for data that represents information attributed to *entities* that is valid within a specific *i-Scope*, and one specific *party* is accountable for the (in)correctness, (mis)use, (in)consistency, (in)coherence, and (non) up-to-date-ness of this information.
Scope	Synonym of '*i-Scope*'.
Session	1. A timespan during which (two) *actors* interact (communicate) with one another; each *actor* may identify any *session* it has and keep. 2. A description (in terms of a set of *session variables*) of the context within which an *actor* communicates with some other *actor*.
Session account	An *account* that is activated within a *session*, of which the *account attributes* represent properties of the *group/user* and/or *party* that the account pertains to.
Session variable	A type-value pair that represents a property of the *session* to which it is attributed.
SessionID	An *identity* of a *session*.
Subject (of a *PI-attribute*)	1. the person to which *personal information* pertains. 2. the person to which the information in the *PI-attribute* pertains.
Symbol	A bit-string, alphanumeric character-string, picture, set of attributes, etc.
Type	A composite of (1) a criterion that distinguishes between *entities* that are of a specific class and *entities* that are not and (2) a criterion that distinguishes between valid and invalid *identity* representations (of this *Type*), where both criteria are valid within the *identity's i-Scope*, and are governed by the *party* that is accountable for the *i-Scope*.
Type (of a type-value pair)	A (data) type whose syntax and semantic constraints are valid within a single *i-Scope*, and are governed by the *party* that is accountable for the *i-Scope*.
Type-Value pair	A pair of data (d1,d2) where d1 is a type (or *Type*) and d2 is a value of that type (or *Type*) that satisfies the *Type* criteria.
User	A *group* that consists of precisely one *actor*.
User account	An *account* pertaining to a (*user, party*) pair.
User authentication	*Authentication*, where the *credentials* identify a *user account*.
Value (of a type-value pair)	A data element (bit string) whose syntax and semantics are defined by the type (or *Type*) in the type-value pair.